axiom™

The 30 Year Horizon

Manuel Bronstein *William Burge* *Timothy Daly*
James Davenport *Michael Dewar* *Martin Dunstan*
Albrecht Fortenbacher *Patrizia Gianni* *Johannes Grabmeier*
Jocelyn Guidry *Richard Jenks* *Larry Lambe*
Michael Monagan *Scott Morrison* *William Sit*
Jonathan Steinbach *Robert Sutor* *Barry Trager*
Stephen Watt *Jim Wen* *Clifton Williamson*

Volume Bibliography: Axiom Literature Citations

January 3, 2021

7beff147070c2fd433caddc60932eecd30ca28c3

Inclusion of names in the list of credits is based on historical information and is as accurate as possible. Inclusion of names does not in any way imply an endorsement but represents historical influence on Axiom development.

Roy Adler
Christian Aistleitner
Michael Albaugh
Cyril Alberga
Jason Allen
Richard Anderson
George Andrews
Jerry Archibald
S.J. Atkins
Jeremy Avigad
Brent Baccala
Knut Bahr
Henry Baker
Martin Baker
Stephen Balzac
Yurij Baransky
David R. Barton
Thomas Baruchel
Gerald Baumgartner
Gilbert Baumslag
Michael Becker
Nelson H. F. Beebe
Jay Belanger
Siddharth Bhat
David Bindel
Fred Blair
Vladimir Bondarenko
Ed Borasky
Mark Botch
Raoul Bourquin
Alexandre Bouyer
Karen Braman
Wolfgang Brehm
Peter A. Broadbery
Martin Brock
Manuel Bronstein
Christopher Brown
Stephen Buchwald
Florian Bundschuh
Luanne Burns
William Burge
Ralph Byers
Quentin Carpent
Jacques Carette
Pierre Casteran
Robert Cavines
Pablo Cayuela
Bruce Char
Ondrej Certik
Tzu-Yi Chen
Bobby Cheng
Cheekai Chin
David V. Chudnovsky
Gregory V. Chudnovsky
Mark Clements
Roland Coeurjoly
Emil Cohen
Hirsh Cohen
Josh Cohen
James Cloos
Jia Zhao Cong
Christophe Conil
Don Coppersmith
George Corliss
Robert Corless
Gary Cornell
Frank Costa
Meino Cramer
Karl Crary
Jeremy Du Croz
David Cyganski
Nathaniel Daly
Timothy Daly Sr.
Timothy Daly Jr.
James H. Davenport
David Day
James Demmel
Didier Deshommes
Michael Dewar
Inderjit Dhillon
Jack Dongarra
Jean Della Dora
Gabriel Dos Reis
Claire DiCrescendo
Sam Dooley
Pierre Doucy
Nicolas James Doye
Zlatko Drmac
Lionel Ducos
Iain Duff
Lee Duhem
Martin Dunstan
Brian Dupee
Dominique Duval
Robert Edwards
Hans-Dieter Ehrich
Heow Eide-Goodman
Alexandra Elbakyan
Carl Engelman
Lars Erickson
Mark Fahey
William Farmer
Richard Fateman
Bertfried Fauser
Stuart Feldman
John Fletcher
Brian Ford
Albrecht Fortenbacher
George Frances
Constantine Frangos
Timothy Freeman
Korrinn Fu
Marc Gaetano
Rudiger Gebauer
Van de Geijn
Kathy Gerber
Patricia Gianni
Eitan Gurari
Gustavo Goertkin
Samantha Goldrich
Max Goldstein
Holger Gollan
Teresa Gomez-Diaz
Ralph Gomory
Laureano Gonzalez-Vega
Stephen Gortler
Johannes Grabmeier
Matt Grayson
Martin Griss
Andrey G. Grozin
Klaus Ebbe Grue
James Griesmer
Vladimir Grinberg
Oswald Gschnitzer
Ming Gu
Fred Gustavson
Jocelyn Guidry
Gaetan Hache
Steve Hague
Satoshi Hamaguchi
Sven Hammarling
Mike Hansen
Richard Hanson
Richard Harke
Joseph Harry
Bill Hart
Vilya Harvey
Martin Hassner
Arthur S. Hathaway
Dan Hatton
Waldek Hebisch
Karl Hegbloom
Ralf Hemmecke
Tony Hearn
Henderson
Antoine Hersen
Nicholas J. Higham
Lou Hodes
Alan Hoffman
Hoon Hong
Roger House
Joris van der Hoeven
Gernot Hueber
Pietro Iglio
Joan Jaffe
Alejandro Jakubi
Richard Jenks
Bo Kagstrom
William Kahan
Kyriakos Kalorkoti
Kai Kaminski
Matt Kaufmann
Grant Keady
Tom Kelsey
Wilfrid Kendall
Tony Kennedy
David Kincaid
Keshav Kini
Knut Korsvold
Ted Kosan

Charles Lawson
Frederic Lehobey
J. Lewis
John Lipson
Michael Lucks
Dave Mainey
Ursula Martin
Alasdair McAndrew
Roland McGrath
Edi Meier
Simon Michael
Gerard Milmeister
H. Michael Moeller
Scott Morrison
William Naylor
John Nelder
Jinzhong Niu
Kostas Oikonomou
Bill Page
Norm Pass
Didier Pinchon
Frank Pfenning
E. Quintana-Orti
A. Petitet
Greg Puhak
Norman Ramsey
Huan Ren
Jason Riedy
Wilken Rivera
Simon Robinson
Martin Rubey
Philip Santas
Aleksej Saushev
Gerhard Schneider
Frithjof Schulze
Steven Segletes
Nick Simicich
Elena Smirnova
Richard Stallman
Jonathan Steinbach
Fabio Stumbo
Klaus Sutner
Eugene Surowitz
T. Doug Telford
Balbir Thomas
Simon Thompson
Steve Toleque
Barry Trager
Gregory Vanuxem
E.G. Wagner
Paul Wang
Jaap Weel
M. Weller
Thorsten Werther
James T. Wheeler
Clifton J. Williamson
Robert Wisbauer
Knut Wolf
Liu Xiaojun
Qian Yun

George L. Legendre
Michel Levaud
Ren-Cang Li
Rudiger Loos
Richard Luczak
Francois Maltey
Dan Martins
Bob McElrath
Paul McJones
Ian Meikle
Jonathan Millen
William Miranker
Michael Monagan
Joel Moses
Patrice Naudin
Godfrey Nolan
Michael O'Connor
Humberto Ortiz-Zuazaga
David Parnas
Susan Pelzel
Ayal Pinkus
Erik Poll
Gregorio Quintana-Orti
Andre Platzer
Claude Quitte
Anatoly Raportirenko
Albert D. Rich
Renaud Rioboo
Jean Rivlin
Raymond Rogers
Jeff Rutter
Grigory Sarnitskiy
Alfred Scheerhorn
Martin Schoenert
Fritz Schwartz
Srinivasan Seshan
Peter Simons
Jacob Nyffeler Smith
Ken Stanley
Alexander Stepanov
Christine Sundaresan
Robert Sutor
Yong Kiam Tan
James Thatcher
Mike Thomas
Dylan Thurston
Dick Toupin
Hale Trotter
Kresimir Veselic
Bernhard Wall
Stephen Watt
Al Weis
Mark Wegman
Michael Wester
John M. Wiley
Stephen Wilson
Sandra Wityak
Hans Peter Wuermli
Clifford Yapp
Vadim Zhytnikov

Franz Lehner
Howard Levy
Xin Li
Craig Lucas
Camm Maguire
William Martin
Osni Marques
Michael McGettrick
Bob McNeill
David Mentre
Victor S. Miller
Mohammed Mobarak
Marc Moreno-Maza
Mark Murray
C. Andrew Neff
Arthur Norman
Summat Oemrawsingh
Julian A. Padget
Igor Pashev
Michel Petitot
Frederick H. Pitts
Jose Alfredo Portes
Beresford Parlett
Peter Poromaas
Arthur C. Ralfs
Guilherme Reis
Michael Richardson
Robert Risch
Nicolas Robidoux
Michael Rothstein
R.W Ryniker II
David Saunders
William Schelter
Marshall Schor
Jens Axel Segaard
V. Sima
William Sit
Matthieu Sozeau
William Stein
Doug Stewart
Ben Collins-Sussman
Moss E. Sweedler
Max Tegmark
Laurent Thery
Carol Thompson
Francoise Tisseur
Raymond Toy
Themos T. Tsikas
Christof Voemel
Justin Walker
Andreas Weber
Juergen Weiss
James Wen
R. Clint Whaley
Berhard Will
Shmuel Winograd
Waldemar Wiwianka
Yanyang Xiao
David Yun
Paul Zimmermann

Contents

New Foreword

On October 1, 2001 Axiom was withdrawn from the market and ended life as a commercial product. On September 3, 2002 Axiom was released under the Modified BSD license, including this document. On August 27, 2003 Axiom was released as free and open source software available for download from the Free Software Foundation's website, Savannah.

Work on Axiom has had the generous support of the Center for Algorithms and Interactive Scientific Computation (CAISS) at City College of New York. Special thanks go to Dr. Gilbert Baumslag for his support of the long term goal.

The online version of this documentation is roughly 1000 pages. In order to make printed versions we've broken it up into three volumes. The first volume is tutorial in nature. The second volume is for programmers. The third volume is reference material. We've also added a fourth volume for developers. All of these changes represent an experiment in print-on-demand delivery of documentation. Time will tell whether the experiment succeeded.

Axiom has been in existence for over thirty years. It is estimated to contain about three hundred man-years of research and has, as of September 3, 2003, 143 people listed in the credits. All of these people have contributed directly or indirectly to making Axiom available. Axiom is being passed to the next generation. I'm looking forward to future milestones.

With that in mind I've introduced the theme of the "30 year horizon". We must invent the tools that support the Computational Mathematician working 30 years from now. How will research be done when every bit of mathematical knowledge is online and instantly available? What happens when we scale Axiom by a factor of 100, giving us 1.1 million domains? How can we integrate theory with code? How will we integrate theorems and proofs of the mathematics with space-time complexity proofs and running code? What visualization tools are needed? How do we support the conceptual structures and semantics of mathematics in effective ways? How do we support results from the sciences? How do we teach the next generation to be effective Computational Mathematicians?

The "30 year horizon" is much nearer than it appears.

Tim Daly
CAISS, City College of New York
November 10, 2003 ((iHy))

Chapter 1

The Axiom Bibliography

This bibliography serves two purposes. It generates output for the ACM bibliography (beebe.bib) and Axiom (axiom.bib).

This bibliography covers areas of computational mathematics. Papers which mention Axiom have a "keyword=" entry of "axiomref". Papers we have on site have a "paper=" entry. Papers which are referenced in the algebra have the "algebra" tag.

The authors are listed in the index. The topic keywords are listed in the index. Algorithms are mentioned in the index.

The **TO** index entry tries to say that the first named algorithm or author has been updated or improved by the second named algorithm or author.

Introduction of special terms (e.g. Toeplitz matrix) may include a paragraph for those unfamiliar with the terms.

1.1 Axiom Literate Sources

— **axiom.bib** —

```
@misc{Daly17,
  author = "Daly, Timothy and Botch, Mark",
  title = {{Axiom Developer Website}},
  link = "\url{http://axiom-developer.org}",
  year = "2017"
}
```

———-

— **axiom.bib** —

```
@book{Book00,
  author = "Axiom Authors",
  title = {{Volume 0: Axiom Jenks and Sutor}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol0.pdf}",
```

```
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Jenks:2003:AVS"
}
```

———-

— **Jenks:2003:AVS** —

```
@Book{Jenks:2003:AVS,
  author = "Richard D. Jenks and Robert S. Sutor and Tim Daly",
  title = {{Axiom Volume 0: The Scientific Computation System}},
  pages = "xviii + 1187",
  year = "2003",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =  "\url{http://www.axiom-developer.org/axiom-website/bookvol0.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@book{Book01,
  author = "Axiom Authors",
  title = {{Volume 1: Axiom Tutorial}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol1.pdf}",
  publisher = "Axiom Project",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@book{Book02,
  author = "Axiom Authors",
  title = {{Volume 2: Axiom Users Guide}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol2.pdf}",
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Daly:2005:AVAb"
}
```

———-

— **Daly:2005:AVAb** —

```
@Book{Daly:2005:AVAb,
  author = "Tim Daly and Martin Dunstan",
  title = {{Axiom Volume 2: Axiom Users Guide}},
  pages = "iv + 7",
  year = "2005",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.axiom-developer.org/axiom-website/bookvol2.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— axiom.bib —

```
@book{Book03,
  author = "Axiom Authors",
  title = {{Volume 3: Axiom Programmers Guide}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol3.pdf}",
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Daly:2005:AVAc"
}
```

———-

— Daly:2005:AVAc —

```
@Book{Daly:2005:AVAc,
  author = "Tim Daly and Martin Dunstan",
  title = {{Axiom Volume 3: Axiom Programmers Guide}},
  pages = "iv + 3",
  year = "2005",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.axiom-developer.org/axiom-website/bookvol3.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— axiom.bib —

```
@book{Book04,
  author = "Axiom Authors",
  title = {{Volume 4: Axiom Developers Guide}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol4.pdf}",
  publisher = "Axiom Project",
  year = "2016",
```

```
  keywords = "axiomref",
  beebe = "Daly:2005:AVAd"
}
```

———-

— **Daly:2005:AVAd** —

```
@Book{Daly:2005:AVAd,
  author = "Tim Daly and Martin Dunstan",
  title = {{Axiom Volume 4: Axiom Developers Guide}},
  pages = "v + 91",
  year = "2005",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.axiom-developer.org/axiom-website/bookvol4.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@book{Book05,
  author = "Axiom Authors",
  title = {{Volume 5: Axiom Interpreter}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol5.pdf}",
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Daly:2003:AVA",
}
```

———-

— **Daly:2003:AVA** —

```
@Book{Daly:2003:AVA,
  author = "Tim Daly and Martin Dunstan",
  title = {{Axiom Volume 5: Axiom Interpreter}},
  pages = "xlvi + 1387",
  year = "2003",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.axiom-developer.org/axiom-website/bookvol5.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@book{Book06,
  author = "Axiom Authors",
  title = {{Volume 6: Axiom Command}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol6.pdf}",
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Daly:2005:AVAe"
}
```

———-

— **Daly:2005:AVAe** —

```
@Book{Daly:2005:AVAe,
  author = "Tim Daly and Martin Dunstan",
  title = {{Axiom Volume 6: Axiom Command}},
  pages = "vi + 187",
  year = "2005",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.axiom-developer.org/axiom-website/bookvol6.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@book{Book07,
  author = "Axiom Authors",
  title = {{Volume 7: Axiom Hyperdoc}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol7.pdf}",
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Daly:2005:AVAj"
}
```

———-

— **Daly:2005:AVAj** —

```
@Book{Daly:2005:AVAj,
  author = "Tim Daly and Martin Dunstan",
  title = {{Axiom Volume 7: Axiom Hyperdoc}},
  pages = "xvi + 632",
  year = "2005",
  LCCN = "????",
```

```
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.axiom-developer.org/axiom-website/bookvol7.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@book{Book71,
  author = "Axiom Authors",
  title = {{Volume 7.1: Axiom Hyperdoc Pages}},
  publisher = "Axiom Project",
  link = "\url{http://axiom-developer.org/axiom-website/bookvol7.1.pdf}",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@book{Book08,
  author = "Axiom Authors",
  title = {{Volume 8: Axiom Graphics}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol8.pdf}",
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Daly:2005:AVAf"
}
```

———-

— **Daly:2005:AVAf** —

```
@Book{Daly:2005:AVAf,
  author = "Tim Daly and Martin Dunstan",
  title = {{Axiom Volume 8: Axiom Graphics}},
  pages = "xi + 538",
  year = "2005",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.axiom-developer.org/axiom-website/bookvol8.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@book{Book81,
  author = "Axiom Authors",
  title = {{Volume 8.1: Axiom Gallery}},
  publisher = "Axiom Project",
  link = "\url{http://axiom-developer.org/axiom-website/bookvol8.1.pdf}",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@book{Book09,
  author = "Axiom Authors",
  title = {{Volume 9: Axiom Compiler}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol9.pdf}",
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Daly:2005:AVAg"
}
```

———-

— **Daly:2005:AVAg** —

```
@Book{Daly:2005:AVAg,
  author = "Tim Daly and Martin Dunstan",
  title = {{Axiom Volume 9: Axiom Compiler}},
  pages = "iv + 30",
  year = "2005",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =  "\url{http://www.axiom-developer.org/axiom-website/bookvol9.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@book{Book91,
  author = "Axiom Authors",
  title = {{Volume 9.1: Axiom Compiler Details}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol9.1.pdf}",
  publisher = "Axiom Project",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@book{Book10,
  author = "Axiom Authors",
  title = {{Volume 10: Axiom Algebra: Implementation}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol10.pdf}",
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Daly:2005:AVAh"
}
```

———-

— **Daly:2005:AVAh** —

```
@Book{Daly:2005:AVAh,
  author = "Tim Daly and Martin Dunstan",
  title = {{Axiom Volume 10: Axiom Algebra: Implementation}},
  pages = "iv + 5",
  year = "2005",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.axiom-developer.org/axiom-website/bookvol10.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@book{Book101,
  author = "Axiom Authors",
  title = {{Volume 10.1: Axiom Algebra: Theory}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol10.1.pdf}",
  publisher = "Axiom Project",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@book{Book102,
  author = "Axiom Authors",
  title = {{Volume 10.2: Axiom Algebra: Categories}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol10.2.pdf}",
  publisher = "Axiom Project",
```

```
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@book{Book103,
  author = "Axiom Authors",
  title = {{Volume 10.3: Axiom Algebra: Domains}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol10.3.pdf}",
  publisher = "Axiom Project",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@book{Book104,
  author = "Axiom Authors",
  title = {{Volume 10.4: Axiom Algebra: Packages}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol10.4.pdf}",
  publisher = "Axiom Project",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@book{Book105,
  author = "Axiom Authors",
  title = {{Volume 10.5: Axiom Algebra: Numerics}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol10.5.pdf}",
  publisher = "Axiom Project",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@book{Book11,
  author = "Axiom Authors",
  title = {{Volume 11: Axiom Browser}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol11.pdf}",
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Portes:2007:AVA"
```

```
}
```

———-

— **Portes:2007:AVA** —

```
@Book{Portes:2007:AVA,
  author = "Alfredo Portes and Arthur Ralfs and Timothy Daly and
                Martin Dunstan",
  title = {{Axiom Volume 11: Axiom Browser}},
  pages = "xix + 1193",
  year = "2007",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.axiom-developer.org/axiom-website/bookvol11.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@book{Book12,
  author = "Axiom Authors",
  title = {{Volume 12: Axiom Crystal}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol12.pdf}",
  publisher = "Axiom Project",
  year = "2016",
  keywords = "axiomref",
  beebe = "Daly:2005:AVAi"
}
```

———-

— **Daly:2005:AVAi** —

```
@Book{Daly:2005:AVAi,
  author = "Tim Daly and Martin Dunstan",
  title = {{Axiom Volume 12: Axiom Crystal}},
  pages = "iv + 9",
  year = "2005",
  LCCN = "????",
  bibdate = "Tue Mar 30 08:43:19 2010",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.axiom-developer.org/axiom-website/bookvol12.pdf}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@book{Book13,
  author = "Axiom Authors",
  title = {{Volume 13: Proving Axiom Correct}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol13.pdf}",
  publisher = "Axiom Project",
  year = "2016"
}
```

————-

— **axiom.bib** —

```
@book{Book14,
  author = "Axiom Authors",
  title = {{Volume 14: Algorithms}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvol14.pdf}",
  publisher = "Axiom Project",
  year = "2016"
}
```

————-

— **axiom.bib** —

```
@book{Bookbib,
  author = "Axiom Authors",
  title = {{Volume Bibliography: Axiom Literature Citations}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvolbib.pdf}",
  publisher = "Axiom Project",
  year = "2016"
}
```

————-

— **axiom.bib** —

```
@book{Bookbug,
  author = "Axiom Authors",
  title = {{Volume BugList: Axiom Bugs}},
  link = "\url{http://axiom-developer.org/axiom-website/bookvolbug.pdf}",
  publisher = "Axiom Project",
  year = "2016"
}
```

————-

— **axiom.bib** —

```
@misc{list20,
```

```
  author = "Axiom Authors",
  title = {{axiom-developer Archives}},
  year = "2020",
  link = "\url{https://lists.nongnu.org/archive/html/axiom-developer/}"
}
```

———-

1.2 Algebra Documentation References

1.2.1 A

— **axiom.bib** —

```
@inproceedings{Adad89,
  author = "Abadi, Martin and Cardelli, Luca and Pierce, Benjamin
            and Plotkin, Gordon",
  title = {{Dynamic Typing in a Statically Typed Language}},
  booktitle = "16th Principles of Programming Languages",
  publisher = "ACM",
  pages = "213-227",
  year = "1989",
  abstract =
    "Statically typed programming languages allow earlier error
    checking, better enforcement of disciplined programming styles,
    and generation of more efficient object code than languages where
    all type consistency checks are performed at run time. However,
    even in statically typed languages, there is often the need to
    deal with data whose type cannot be determined at compile time. To
    handle such situations safely, we propose to add a type Dynamic
    whose values are pairs of a value $v$ and a type tag T where $v$
    has the type denoted by T. Instances of Dynamic are built with an
    explicit tagging construct and inspected with a type safe typecase
    construct.

    This paper explores the syntax, operational semantics, and
    denotational semantics of a simple language including the type
    Dynamic. We give examples of how dynamically typed values can be
    used in programming. Then we discuss an operational semantics for
    our language and obtain a soundness theorem. We present two
    formulations of the denotational semantics of this language and
    relate them to the operational semantics. Finally, we consider the
    implications of polymorphism and some implementation issues.",
  paper = "Abad89.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Abad94,
  author = "Abadi, Martin and Cardelli, Luca",
  title = {{A Semantics of Object Types}},
  booktitle = "Symp. on Logic in Computer Science '94",
  publisher = "IEEE",
  year = "1994",
  abstract =
    "We give a semantics for a typed object calculus, an extension of
    System {\bf F} with object subsumption and method override.  We interpret
    the calculus in a per model, proving the soundness of both typing and
    equational rules.  This semantics suggests a syntactic translation
    from our calculus into a simpler calculus with neither subtyping nor
    objects",
  paper = "Abad94.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Abad94a,
  author = "Abadi, Martin and Cardelli, Luca",
  title = {{A Theory of Primitive Objects: Untyped and First-Order Systems}},
  booktitle = "Proc. European Symposium on Programming",
  year = "1994",
  abstract =
    "We introduce simple object calculi that support method override and
    object subsumption.  We give an untyped calculus, typing rules, and
    equational rules. We illustrate the expressiveness of our calculi and
    the pitfalls that we avoid.",
  paper = "Abad94a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Abda86,
  author = "Abdali, S. Kamal and Cherry, Guy W. and Soiffer, Neil",
  title = {{A Smalltalk System for Algebraic Manipulation}},
  booktitle = "OOPSLA 86",
  pages = "277-293",
  year = "1986",
  abstract =
    "This paper describes the design of an algebra system Views
    implemented in Smalltalk. Views contains facilities for dynamic
    creation and manipulation of computational domains, for viewing these
    domains as various categories such as groups, rings, or fields, and
    for expressing algorithms generically at the level of categories. The
    design of Views has resulted in the addition of some new abstractions
    to Smalltalk that are quite useful in their own right. Parameterized
    classes provide a means for run-time creation of new classes that
```

```
    exhibit generally very similar behavior, differing only in minor ways
    that can be described by different instantiations of certain
    parameters. Categories allow the abstraction of the common behavior of
    classes that derives from the class objects and operations satisfying
    certain laws independently of the implementation of those objects and
    operations. Views allow the run-time association of classes with
    categories (and of categories with other categories), facilitating the
    use of code written for categories with quite different
    interpretations of operations. Together, categories and views provide
    an additional mechanism for code sharing that is richer than both
    single and multiple inheritance. The paper gives algebraic as well as
    non-algebraic examples of the above-mentioned features.",
  paper = "Abda86.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Abel86,
  author = "Abelson, Harold and Sussman, Gerald Jay and Sussman, Julie",
  title = {{Structure and Interpretation of Computer Programs}},
  year = "1986",
  publisher = "The MIT Press",
  isbn = "0-262-01077-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Abra16b,
  author = "Abraham, Erika and Abbott, John and Becker, Bernd and
            Bigatti, Anna M. and Brain, Martin and Buchberger, Bruno
    and Cimatti, Alessandro and Davenport, James H. and
            England, Matthew and Fontaine, Pascal and Forrest, Stephen
            and Griggio, Alberto and Droening, Daniel and
            Seller, Werner M. and Sturm, Thomas",
  title = {{$SC^2$: Satisfiability Checking Meets Symbolic Computation}},
  journal = "Lecture Notes in Computer Science",
  volume = "9791",
  year = "2016",
  abstract =
    "Symbolic Computation and Satisfiability Checking are two research
    areas, both having their individual scientific focus but sharing
    also common interests in the development, implementation and
    application of decision procedures for arithmetic
    theories. Despite their commonalities, the two communities are
    weakly connected. The aim of our newly accepted $SC^2$ project
    (H2020-FETOPEN-CSA) is to strengthen the connection between these
```

```
    communities by creating common platforms, initiating interaction
    and exchange, identifying common challenges, and developing a
    common roadmap from theory along the way to tools and (industrial)
    applications. In this paper we report on the aims and on the first
    activities of this project, and formalise some relevant challenges
    for the unified $SC^2$ community.",
  paper = "Abra16b.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Abra66,
  author = "Abrahams, Paul W. and Barnett, Jeffrey A. and Book, Erwin
            and Firth, Donna and Kameny, Stanley L. and Weissman, Clark
            and Hawkinson, Lowell and Levin, Michael I. and
            Saunders, Robert A.",
  title = {{The Lisp 2 Programming Language and System}},
  link =
  "\url{http://www.softwarepreservation.org/projects/LISP/lisp2/AbrahamsEtAl-LISP2.pdf}",
  year = "1966",
  paper = "Abra66.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Adam04,
  author = "Adams, Andrew A. and Davenport, James H.",
  title = {{Copyright Issues for MKM}},
  journal = "LNCS",
  volume = "3119",
  year = "2004",
  abstract =
    "We present an overview of the current situation and recent and
    expected future developments in areas of copyright law and
    economics relevant to Mathematical Knowledge Management.",
  paper = "Adam04.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Adax12,
  author = "ISO/IEC 8652:2012(E)",
  title = {{Ada Reference Manual}},
  publisher = "U.S. Government",
```

```
  year = "2012",
  link = "\url{http://www.ada-auth.org/standards/12rm/RM-Final.pdf}"
}
```

———-

— **axiom.bib** —

```
@article{Ahar12,
  author = "Aharonovich, I. and Horwitz, L.P.",
  title = {{Radiation-reaction in classical off-shell electrodynamics,
            I: The above mass-shell case}},
  journal = "J. Math. Phys.",
  volume = "53",
  number = "3",
  year = "2012",
  abstract =
    "Offshell electrodynamics based on a manifestly covarieant
    off-shell relativeistic dynamics of Stueckelberg, Horwitz, and
    Piron, is five-dimensional. In this paper, we study the problem of
    radiation reaction of a particle in motion in this framework. In
    particular, the case of the above-mass-shell is studied in detail,
    where the renormalization of the Lorentz force leads to a system
    of non-linear differential equations for 3 Lorentz scalars. The
    system is then solved numerically, where it is shown that the
    mass-shell deviation scalar either smoothly falls down to 0 (this
    result provides a mechanism for the mass stability of the
    off-shell theory), or strongly diverges under more extremem
    conditions. In both cases, no runaway motion is
    observed. Stability analysis indicates that the system seems to
    have chaotic behavior. It is also shown that, although a motion
    under which the mass-shell deviation $\epsilon$ is constant but
    not-zero, is indeed possible, but, it is unstable, and eventually
    it either decays to 0 or diverges.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Ahox86,
  author = "Aho, Alfred V. and Sethi, Ravi and Ullman, Jeffrey D.",
  title = {{Compilers: Principles, Techniques, and Tools}},
  year = "1986",
  publisher = "Addison-Wesley",
  isbn = "978-0201100884"
}
```

———-

— **axiom.bib** —

```
@book{Aitk99,
  author = "Ait-Kaci, Hassan",
  title = {{Warren's Abstract Machine: A Tutorial Reconstruction}},
  publisher = "MIT Press",
  isbn = "0-262-51058-8",
  year = "1999",
  link = "\url{http://wambook.sourceforge.net/wambook.pdf}",
  paper = "Aitk99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Alam12,
  author = "Alama, Jesse and Mamane, Lionel and Urban, Josef",
  title = {{Dependencies in Formal Mathematics: Applications and
            Extration for Coq and Mizar}},
  journal = "LNCS",
  volume = "7362",
  year = "2012",
  abstract =
    "Two methods for extracting detailed formal dependencies from the
    Coq and Mizar system are presented and compared. The methods are
    used for dependency extraction from two large mathematical
    repositories: the Coq Repository at Nijmegen and the Mizar
    Mathematical Library. Several applications of the detailed
    dependency analysis are described and proposed. Motivated by the
    different applications, we discuss the various kinds of
    dependencies that we are interested in, and the suitability of
    various dependency extraction methods.",
  paper = "Alam12.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Aldr13,
  author = "Aldrich, Jonathan",
  title = {{The Power of Interoperability: Why Objects are Inevitable}},
  booktitle = "Onward!",
  publisher = "ACM",
  year = "2013",
  link = "\url{https://www.cs.cmu.edu/~aldrich/papers/objects-essay.pdf}",
  abstract =
    "Three years ago, in this venue, Cook argued that in their
    essence, objects are what Reynolds called {\sl procedural data
    structures}. His observations raises a natural question: if
    procedural data structures are the essence of objects, has this
```

```
    contributed to the empirical success of objects, and if so, how?

    This essay attempts to answer that question. After reviewing
    Cook's definition, I propose the term {\sl service abstractions}
    to capture the essential nature of objects. This terminology
    emphasizes, following Kay, that objects are not primarily about
    representing and manipulating data, but are more about providing
    services in support of higher-level goals. Using examples taken
    from object oriented frameworks, I illustrate the unique design
    leverage that service abstractions provide: the ability to define
    abstractions that can be extended, and whose extensions are
    interoperable in a first-class way. The essay argues that the form
    of interoperable extension supported by service abstractions is
    essential to modern software: many modern frameworks and
    ecosystems could not have been built without service
    abstractions. In this sense, the success of objects was not a
    coincidence: it was an inevitable consequence of their service
    abstraction nature.",
  paper = "Aldr13.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Alha19,
  author = "Al-hassy, Musa and Carette, Jacques and Kahl, Wolfram",
  title = {{A Language Feature to Unbundle Data at Will}},
  year = "2019",
  link = "\url{http://www.cas.mcmaster.ca/~carette/publications/gpce19main-p33.pdf}",
  abstract =
    "Programming languages with sufficiently expressive type systems
    provide users with different means of data 'bundling'.
    Specifically, in dependently-typed languages such as Agda, Coq,
    Lean and Idris, one can choose to encode information in a record
    either as a parameter or a field. For example, we can speak of
    graphs over a particular vertex set, or speak of arbitrary graphs
    where the vertex set is a component. These create isomorphic
    types, but differ with respect to intended use. Traditionally, a
    library designer would make this choice (between parameters and
    fields); if a user wants a different variant, they are forced to
    build cnversion utilities, as well as duplicate functionality. For
    a graph data type, if a library only provides a Haskell-like
    typeclass view of graphs over a vertex set, yet a user wishes to
    work with the category of graphs, they must now package a vertex
    set as a component in a record along with a graph over that set.

    We design and implement a language feature that allows both the
    library designer and the user to make the choice of information
    exposure only when necessary, and otherwise leave the
    distinguishing line between parameters and fields unspecified.
    Our language feature is currently implemented as a prototype
```

```
    meta-program incorporated into Agda's Emacs ecosystem, in a what
    that is unobtrusive to Agda users.",
  paper = "Alha19.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Alme11,
  author = "Almeida, Jose Bacelar and Frade, Maria Joao and
            Pinto, Jorge Sousa and de Sousa, Simao Melo",
  title = {{Rigorous Software Development}},
  year = "2011",
  publisher = "Springer",
  isbn = "978-0-85729-017-5",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Alpe86,
  author = "Alpern, Bowen L.",
  title = {{Proving Temporal Properties of Concurrent Programs: A
            Non-Temporal Approach}},
  school = "Cornell University",
  year = "1986",
  abstract =
    "This thesis develops a new method for proving properties of
    concurrent programs and gives formal definitions for safety and
    liveness. A property is specified by a property recognizer -- a
    finite-state machine that accepts the sequences of program states
    in the property it specifies. A property recognizer can be
    constructed for any temporal logic formula.

    To prove that a program satisfies a property specified by a
    deterministic program recognizer, one must show that any history
    of the program will be accepted by the recognizer. This is done by
    demonstrating that proof obligations derived from the recognizer
    are met. These obligations require the program prover to exhibit
    certain invariant assertions and variant functions and to prove
    the validty of certain predicates and Hoare triples. Thus, the
    same techniques used to prove local correctness of a while loop
    can be used to prove temporal properties of concurrent
    programs. No temporal inference is required.

    The invariant assertions required by the proof obligations
    establish a correspondence between the states of the program and
    those of the recogniser. Such correspondences can be denoted by
    property outlines, a generalization of proof outlines.
```

```
    Some non-deterministic property recognizers have no deterministic
    equivalents To prove that a program satisfies a non-deterministic
    property, a deterministic sub-property that the program satisfies
    must be found. This is shown possible, provided the program state
    space is finite.

    Finally, safety properties are formalized as the closed sets of a
    topological space and liveness properties as its dense sets. Every
    property is shown to be the intersection of a safety property and
    a liveness property. A technique for separating a property
    specified by a deterministic property recognizer into its safety
    and liveness aspects is also presented.",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Alte07,
  author = "Altenkirch, Thorsten and McBride, Conor and Swierstra, Wouter",
  title = {{Observational Equality, Now!}},
  booktitle = "ACM Workshop Programming Languages meets Program
               Verification",
  publisher = "ACM",
  pages = "57-68",
  year = "2007"
}
```

———-

— **axiom.bib** —

```
@misc{Alte18,
  author = "Altenkirch, Thorsten",
  title = {{Naive Type Theory}},
  year = "2018",
  link = "\url{https://www.youtube.com/watch?v=bNG53SA4n48p}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Altu19,
  author = "Alturki, Musab A. and Moore, Brandon",
  title = {{K vs Coq as Language Verification Frameworks}},
  year = "2019",
  link = "\url{https://runtimeverification.com/blog/k-vs-coq-as-language-verification-frameworks-part-1-of-3/}"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Amad93,
  author = "Amadio, Roberto M. and Cardelli, Luca",
  title = {{Subtyping Recursive Types}},
  journal = "TOPLAS '93",
  volume = "15",
  number = "4",
  year = "1993",
  pages = "575-631",
  abstract =
    "We investigate the interactions of subtyping and recursive types, in
    a simply typed $\lambda$-calculus. The two fundamental questions here are
    whether two (recursive)types are in the subtype relation and whether a
    term has a type. To address the first question, we relate various
    definitions of type equivalence and subtyping that are induced by a
    model, an ordering on infinite trees, an algorithm, and a set of type
    rules. We show soundness and completeness among the rules, the
    algorithm, and the tree semantics. We also prove soundness and a
    restricted form of completeness for the model. To address the second
    question, we show that to every pair of types in the subtype relation
    we can associate a term whose denotation is the uniquely determined
    coercion map between the two types. Moreover, we derive an algorithm
    that, when given a term with implicit coercions, can infer its least
    type whenever possible.",
  paper = "Amad93.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Amin18,
  author = "Amin, Nada and Rompf, Tiark",
  title = {{Collapsing Towers of Interpreters}},
  booktitle = "Principles of Programming Languages",
  year = "2018",
  publisher = "ACM",
  abstract =
    "Given a tower of interpreters, i.e., a sequence of multiple
    interpreters interpreting one another as input programs, we aim to
    collapse this tower into a compiler that removes all interpretive
    overhead and runs in a single pass. In the real world, a use case
    might be Python code executed by an x86 runtime, on a CPU emulated
    in a JavaScript VM, running on an ARM CPU. Collapsing such a tower
    can not only exponentially improve runtime performance, but also
    enable the use of base language tools for interpreted programs,
    e.g. for analysis and verification. In this paper, we lay the
```

```
    foundations in an idealized but realistic setting.

    We present a multi-level lambda calculus that features staging
    constructs and stage polymorphism: based on runtime parameters, an
    evaluator either executes source code (thereby acting as an
    interpreter) or generates code (thereby acting as a compiler). We
    identify stage polymorphism, a programming model from the domain
    of high-performance program generators, as the key mechanism to
    make such interpreters compose in a collapsible way.

    We present Pink, a meta-circular Lisp-like evaluator on top of
    this calculus, and demonstrate that we can collapse arbitrarily
    many levels of self-interpretation, including levels with
    semantics modifications. We discuss several examples: compiling
    regular expressions through an interpreter to base code, building
    program transformers from modified interpreters, and others. We
    develop these ideas further to include reflection and reification,
    culminating in Purple, a reflective language inspired by Brown,
    Blond, and Black, which realizes a conceptually infinite tower,
    where every aspect of the semantics can change
    dynamically. Addressing an open challenge, we show how user
    programs can be compiled and recompiled under user-modified
    semantics.",
  paper = "Amin18.pdf",
  keywords = "printed"
}
```

— axiom.bib —

```
@article{Andr01,
  author = "Andreoli, Jean-Marc",
  title = {{Focussing and Proof Construction}},
  journal = "Annals of Pure and Applied Logic",
  volume = "107",
  pages = "131-163",
  year = "2001",
  abstract =
    "This paper proposoes a synthetic presentation of the proof
    construction paradigm, which underlies most of the research and
    development in the so-called ``logic programming'' area. Two
    essential aspects of this paradigm are discussed here: true
    non-determinism and partial information. A new formulation of
    Focussing, the basic property used to deal with non-determinism in
    proof construction, is presented. This formulation is then used to
    introduce a general constraint-based technique capable of dealing
    with partial information in proof construction. One of the
    baselines of the paper is to avoid to rely on syntax to describe
    the key mechanisms of the paradigm. In fact, the bipolar
    decomposition of formulas captures their main structure, which can
    then be directly mapped into a sequent system that uses only
    atoms. This system thus completely ``dissolves'' the syntax of the
```

```
    formulas and retains only their behavioural content as far as
    proof construction is concerned. One step further is taken with
    the so-called ‘‘abstract’’ proofs, which dissolves in a similar
    way the specific tree-like syntax of the proofs themselves and
    retains only what is relevant to proof construction.",
  paper = "Andr01.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Andr07,
  author = "Andres, Mirian and Lamban, Laureano and Rubio, Julio",
  title = {{Executing in Common Lisp, Proving in ACL2}},
  journal = "LNCS",
  volume = "4573",
  year = "2007",
  abstract =
    "In this paper, an approach to integrate an already-written Common
    Lisp program for algebraic manipulation with ACL2 proofs of
    properties of that program is presented. We report on a particular
    property called ‘‘cancellation theorem’’, which has been proved in
    ACL2, and could be applied to several problems in the field of
    Computational Algebraic Topology.",
  paper = "Andr07.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Andr02,
  author = "Andrews, Peter B.",
  title = {{An Introduction to Mathematical Logic and Type Theory: To
            Truth Through Proof}},
  comment = "Applied Logic Series 27",
  publisher = "Springer",
  year = "2002",
  isbn = "978-94-015-9934-4",
  paper = "Andr02.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Anto12,
  author = "Antoy, Sergio and Peters, Arthur",
  title = {{Compiling a Functional Logic Language}},
```

```
  booktitle = "11th Int. Symp. on Functional and Logic Programming",
  journal = "LNCS",
  volume = "7294",
  pages = "17-31",
  year = "2012",
  abstract =
    "We present the design of a compiler for a functional logic
    programming language and discuss the compiler's implementation.
    The source program is abstracted by a constructor based graph
    rewriting system obtained from a functional logic program after
    syntax desugaring, lambda lifting and similar transformations
    provided by a compiler's front-end. This system is
    non-deterministic and requires a specialized normalization
    strategy. The target program consists of 3 procedures that execute
    graph replacements originating from either rewrite or pull-tab
    seps. These procedures are deterministic and easy to encode in an
    ordinary programming language. We describe the generation of the 3
    procedures, discuss the correctness of our approach, highlight
    some key elements of an implementation, and benchmark the
    performance of a proof of concept. Our compilation scheme is
    elegant and simple enough to be presented in one page.",
  paper = "Anto12.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Appe98a,
  author = "Appel, Andrew W.",
  title = {{SSA is Functional Programming}},
  journal = "SIGPLAN Notices",
  year = "1998",
  abstract =
    "Static Single-Assignment (SSA) form is an intermediate language
    designed to make optimization clean and efficient for imperative
    language (Fortran, C) compilers. Lambda-Calculus is an
    intermediate language that makes optimization clean and efficient
    for functional language (Scheme, ML, Haskell) compilers. The SSA
    community draws pictures of graphs with basic blocks and flow
    edges, and the functional language community writes lexically
    nested functions, but (as Richard Kelsey recently pointed out)
    they're both doing exactly the same thing in different notation.",
  paper = "Appe98a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Appe14,
  author = "Appel, Andrew W. and Dockins, Robert and Hobor, Aquinas
            and Beringer, Lennart and Dodds, Josiah and
            Stewart, Gordon and Blazy, Sandrine and Leroy, Xavier",
  title = {{Program Logics for Certified Compilers}},
  publisher = "Cambridge University Press",
  isbn = "978-1-107-04801-0",
  link = "\url{https://www.cs.princeton.edu/~appel/papers/plcc.pdf}",
  year = "2014",
  paper = "Appe14.pdf"
}
```

———-

— axiom.bib —

```
@article{Appe17a,
  author = "Appel, Andrew W. and Beringer, Lennart and Chlipala, Adam
            and Pierce, Benjamin C. and Shao, Zhong and Weirich, Stephanie
            and Zdancewic, Steve",
  title = {{Position Paper: The Science of Deep Specification}},
  journal = "Philosophical Transactions of the Royal Society",
  volume = "375",
  year = "2017",
  link = "\url{https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2016.0331}",
  abstract =
    "We introduce our efforts within the project ``The Science of Deep
    Specification'' to work out the key formal underpinnings of
    inductrial-scale formal specifications of software and hardware
    components, anticipating a world where large verified systems are
    routinely built out of smaller verified components that are also
    used by many other projects. We identify an important class of
    specification that has already been used in a few experiments that
    connect strong component-correctness theorems across the work of
    different teams. To help popularize the unique advantages of that
    style, we dub it {\sl deep specification}, and we say that it
    encompasses specifications that are {\sl rich}, {\sl two-sided},
    {\sl formal}, and {\sl live} (terms that we define in the
    article). Our core team is developing a proof-of-concept system
    (based on the Coq proof assistant) whose specification and
    verification work is divided across largely decoupled subteams at
    our four institutions, encompassing hardware microarchitecture,
    compilers, operating systems and applications, along with
    cross-cutting principles and tools for effective specification. We
    also aim to catalyse interest in the approach, not just by basic
    researchers but also by users in industry.

    This article is part of the themed issue ``Verified trustworthy
    software systems''",
  paper = "Appe17a.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Appe18,
  author = "Appel, Andrew W.",
  title = {{Verified Functional Algorithms}},
  year = "2018",
  publisher = "University of Pennsylvania",
  link =
    "\url{https://softwarefoundations.cis.upenn.edu/vfa-current/index.html}",
  abstract =
    "Here's a good way to build formally verified correct software:
     \begin{itemize}
     \item Write your program in an expressive language with a good
     proof theory (the Gallina language embedded in Coq's logi)
     \item Prove it correct in Coq
     \item Compile it with an optimizing ML compiler
     \end{itemize}

    Since you want your programs to be {\sl efficient}, you'll want to
    implement sophisticated data structures and algorithms. Since
    Gallina is a {\sl purely functional} language, it helps to have
    purely functional algorithms. In this volume you'll learn how to
    specify and verify (prove the correctness of) sorting algorithms,
    binary search trees, balanced binary search trees, and priority
    queues. Before using this book, you should have some understanding
    of these algorithms and data structures, available in any standard
    undergraduate algorithms textbook. This electronic book is Volume
    3 of the {\sl Software Foundations} series, which presents the
    mathematical underpingging of reliable software. It builds on
    {\sl Software Foundations Volume 1} (Logical Foundations), but
    does not depend on Volume 2. The exposition here is intended for a
    broad range of readers, from advanced undergraduates to PhD
    students and researchers. The principal novelty of
    {\sl Software Foundations} is that it is one hundred percent
    formalized and machine-checked: the entire text is literally a
    script for Coq. It is intended to be read alongside an interactive
    session with Coq. All the details in the text are fully formalized
    in Coq, and the exercises are designed to be worked using Coq.",
  paper = "Appe18.tgz"
}
```

———-

— **axiom.bib** —

```
@article{Ardi09,
  author = "Ardizzoni, Alessandro and Stumbo, Fabio",
  title = {{Quadratic Lie Algebras}},
  journal = "Commun. Algebra",
  volume = "39",
  number = "8",
```

```
  pages = "2723-2751",
  year = "2011",
  link = "\url{https://arxiv.org/pdf/0906.4617.pdf}",
  abstract =
    "In this paper, the notion of universal envoloping algebra
    introduced in [A. Ardizzoni, A First Sight Towards Primitively
    Generated Connected Braided Bialgebras] is specialized to the case
    of braided vector spaces whole Nichols algebra is quadratic as an
    algebra. In this setting a classification of universal enveloping
    algebras for braided vectors spaces of dimension not greater than
    2 is handled. As an application, we investigate the structure of
    primitively generated connected braided bialgebras whose braided
    vector space of primitive elements forms a Nichols algebra which
    is a quadratic algebra.",
  paper = "Ardi09.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Arko17,
  author = "Arkoudas, Konstantine and Musser, David",
  title = {{Fundamental Proof Methods in Computer Science}},
  publisher = "MIT Press",
  year = "2017",
  isbn = "978-0262035538",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Arms19,
  author = "Armstrong, Alasdair and Bauereiss, Thomas and Campbell, Brian
            and Reid, Alastair and gray, Kathryn E. and Norton, Robert M.
            and Mundkur, Prashanth and Wassell, Mark and French, Jon
            and Pulte, Christopher and Flus, Shaked and Krishnaswami, Neel
            and Sewell, Peter",
  title = {{ISA Semantics for ARMv8-A, RISC-V, and CHERI-MIPS}},
  journal = "Proc. ACM Programming Lang.",
  volume = "3",
  year = "2019",
  abstract =
    "Architecture specifications notionally define the fundamental
    interface between hardware and software: the envelope of allowed
    behaviour for processor implementations, and the basic assumptions for
    software development and verification. But in practice, they are
    typically prose and pseudocode documents, not rigorous or executable
    artifacts, leaving software and verification on shaky ground.
```

```
    In this paper, we present rigorous semantic models for the sequential
    behaviour of large parts of the mainstream ARMv8-A, RISC-V, and MIPS
    architectures, and the research CHERI-MIPS architecture, that are
    complete enough to boot operating systems, variously Linux, FreeBSD,
    or seL4. Our ARMv8-A models are automatically translated from
    authoritative ARM-internal definitions, and (in one variant) tested
    against the ARM Architecture Validation Suite.

    We do this using a custom language for ISA semantics, Sail, with a
    lightweight dependent type system, that supports automatic generation
    of emulator code in C and OCaml, and automatic generation of
    proof-assistant definitions for Isabelle, HOL4, and (currently only
    for MIPS) Coq. We use the former for validation, and to assess
    specification coverage. To demonstrate the usability of the latter, we
    prove (in Isabelle) correctness of a purely functional
    characterisation of ARMv8-A address translation. We moreover integrate
    the RISC-V model into the RMEM tool for (user-mode) relaxed-memory
    concurrency exploration. We prove (on paper) the soundness of the core
    Sail type system.

    We thereby take a big step towards making the architectural
    abstraction actually well-defined, establishing foundations for
    verification and reasoning.",
  paper = "Arms19.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inbook{Arms92,
  author = "Armstrong, J.L. and Birding, S.R. and Williams, M.C.",
  title = {{Use of Prolog for Developing a New Programming Language}},
  booktitle = "The Practical Application of Prolog",
  year = "1992",
  chapter = "unknown",
  pages = "unknown",
  publisher = "Institute of Electrical Engineers, London",
  abstract =
    "This paper describes how Prolog was used for the development of a
    new concurrent real-time symbolic programming language called
    Erlang.

    Erlang was developed by first building a prototype in Prolog --
    the prototype was used by a user group to test their reactions to
    the language. As time passed many features were added (and
    removed) from the interpreter and eventually the language reached
    a level of maturity where it was decided to try it out on a
    significant problem.

    About 3 years and some 20,000 lines of Erlang later, performance
```

```
    became an issue -- we wrote Prolog cross compilers from Erlang to
    various concurrent logic programming languages followed by a
    direct implementation of Erlang itself. The direct implementation
    of Erlang was loosely based on the WAM and made by writing a
    Prolog compiler from Erlang to a new abstractmachine and an
    emulator for the abstract machine in 'C'. The instruction set for
    the abstract machine was first prototyped in Prolog -- finally the
    compiler was re-written in Erlang, thus totally removing any
    dependency on Prolog.

    This paper describes some of the key events which lay between the
    simple prototype and the current version of the language.",
  paper = "Arms92.pdf",
  keywords = "printed,DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Arms14,
  author = "Armstrong, Joe",
  title = {{The Mess We're In}},
  link = "\url{https://www.youtube.com/watch?v=lKXe3HUG214}",
  year = "2014",
  keywords = "axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Arno98,
  author = "Arnold,  Vladimir Igorevich",
  title = {{On Teaching Mathematics}},
  journal = "Russian Math. Surveys",
  volume = "53",
  number = "1",
  pages = "229-236",
  year = "1998",
  link = "\url{https://www.uni-muenster.de/Physik_TP/~munsteg/arnold.html}"
}
```

———-

— **axiom.bib** —

```
@article{Aspe04,
  author = "Asperti, Andrea and Guidi, Ferruccio and Coen, Claudio Sacerdoti
            and Tassi, Enrico and Zacchiroli, Stefano",
  title = {{A Content Based Mathematical Search Engine: Whelp}},
```

```
  journal = "LNCS",
  volume = "3839",
  year = "2004",
  pages = "17-32",
  isbn = "3-540-31428-8",
  abstract =
    "The prototype of a content based search engine for mathematical
    knowledge supporting a small set of queries requiring matching and/or
    typing operations is described. The prototype, called Whelp, exploits
    a metadata approach for indexing the information that looks far more
    flexible than traditional indexing techniques for structured
    expressions like substitution, discrimination, or context trees. The
    prototype has been instantiated to the standard library of the Coq
    proof assistant extended with many user contributions.",
  paper = "Aspe04.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Aspe06,
  author = "Asperti, Andrea and Geuvers, Herman and Loeb, Iris and
            Mamane, Lionel Elie and Coen, Claudio Sacerdoti",
  title = {{An Interactive Algebra Course with Formalised Proofs and
            Definitions}},
  journal = "LNCS",
  volume = "4108",
  year = "2006",
  abstract =
    "We describe a case-study of the application of web technology to
    create web-based didactic material out of a repository of formal
    mathematics, using the structure of an existing course. The paper
    discusses the difficulties related to associating notation to a
    formula, the embedding of formal notions into a document (the
    ‘‘view’’), and the rednering of proofs.",
  paper = "Aspe06.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Aspe06a,
  author = "Asperti, Andrea and Coen, Claudio Sacerdoti and
            Tassi, Enrico and Zacchiroli, Stefano",
  title = {{Crafting a Proof Assistant}},
  booktitle = "Proc. Types 2006: Conf. of the Types Project",
  year = "2006",
  abstract =
    "Proof assistants are complex applications whose development has
```

```
    never been properly systematized or documented. This work is a
    contribution in this direction, based on our experience with the
    development of Matita: a new interactive theorem prover basedas
    Coqon the Calculus of Inductive Constructions (CIC). In particular,
    we analyze its architecture focusing on the dependencies of its
    components, how they implement the main functionalities, and their
    degree of reusability. The work is a first attempt to provide a ground
    for a more direct comparison between different systems and to
    highlight the common functionalities, not only in view of
    reusability but also to encourage a more systematic comparison of
    different softwares and architectural solutions.",
  paper = "Aspe06a.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Aspe07,
  author = "Asperti, Andrea and Tassi, Enrico",
  title = {{Higher Order Proof Reconstruction from
            Paramodulation-based Refutations: The Unit Equality Case}},
  booktitle = "MKM 2007",
  year = "2007",
  abstract =
    "In this paper we address the problem of reconstructing a higher
    order, checkable proof object starting from a proof trace left by a
    first order automatic proof searching procedure, in a restricted
    equational framework. The automatic procedure is based on
    superposition rules for the unit equality case. Proof transformation
    techniques aimed to improve the readability of the final proof are
    discussed.",
  paper = "Aspe07.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Aspe07a,
  author = "Asperti, Andrea and Coen, Claudio Sacerdoti and Tassi, Enrico
            and Zacchiroli, Stefano",
  title = {{User Interaction with the Matita Proof Assistant}},
  journal = "J. of Automated Reasoning",
  volume = "39",
  number = "2",
  pages = "109-139",
  year = "2007",
  abstract =
    "Matita is a new, document-centric, tactic-based interactive theorem
    prover. This paper focuses on some of the distinctive features of the
```

```
    user interaction with Matita, characterized mostly by the organization
    of the library as a searchable knowledge base, the emphasis on a
    high-quality notational rendering, and the complex interplay between
    syntax, presentation, and semantics.",
  paper = "Aspe07a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Aspe09,
  author = "Asperti, Andrea and Ricciotti, Wilmer and Coer, Claudio
            Sacerdoti and Tassi, Enrico",
  title = {{A Compact Kernel for the Calculus of Inductive Constructions}},
  journal = "Sadhana",
  volume = "34",
  number = "1",
  year = "2009",
  pages = "71-104",
  abstract =
    "The paper describes the new kernel for the Calculus of Inductive
    Constructions (CIC) implemented inside the Matita Interactive
    Theorem Prover.  The design of the new kernel has been completely
    revisited since the first release, resulting in a remarkably compact
    implementation of about 2300 lines of OCaml code. The work is meant
    for people interested in implementation aspects of Interactive
    Provers, and is not self contained . In particular, it requires good
    acquaintance with Type Theory and functional programming
    languages.",
  paper = "Aspe09.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Aspe09a,
  author = "Asperti, Andrea and Ricciotti, Wilmer and Coer, Claudio
            Sacerdoti and Tassi, Enrico",
  title = {{Hints in Unification}},
  journal = "LNCS",
  volume = "5674",
  pages = "84-98",
  year = "2009",
  isbn = "978-3-642-03358-2",
  abstract =
    "Several mechanisms such as Canonical Structures, Type Classes, or
    Pullbacks have been recently introduced with the aim to improve the
    power and flexibility of the type inference algorithm for interactive
    theorem provers. We claim that all these mechanisms are particular
```

```
    instances of a simpler and more general technique, just consisting in
    providing suitable hints to the unification procedure underlying type
    inference. This allows a simple, modular and not intrusive
    implementation of all the above mentioned techniques, opening at the
    same time innovative and unexpected perspectives on its possible
    applications.",
  paper = "Aspe09a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Aspe09b,
  author = "Asperti, Andrea and Ricciotti, Wilmer and Coer, Claudio
            Sacerdoti and Tassi, Enrico",
  title = {{A New Type for Tactics}},
  booktitle = "SIGSAM PLMMS 2009",
  publisher = "ACM",
  year = "2009",
  isbn = "978-1-60558-735-6",
  abstract =
    "The type of tactics in all (procedural) proof assistants is (a
    variant of) the one introduced in LCF. We discuss why this is
    inconvenient and we propose a new type for tactics that 1) allows the
    implementation of more clever tactics; 2) improves the implementation
    of declarative languages on top of procedural ones; 3) allows for
    better proof structuring; 4) improves proof automation; 5) allows
    tactics to rearrange and delay the goals to be proved (e.g. in case of
    side conditions or PVS subtyping judgements).",
  paper = "Aspe09b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Aspe10,
  author = "Asperti, Andrea and Tassi, Enrico",
  title = {{Smart Matching}},
  journal = "LNCS",
  volume = "6167",
  pages = "263-277",
  year = "2010",
  isbn = "978-3-642-14128-7",
  abstract =
    "One of the most annoying aspects in the formalization of mathematics
    is the need of transforming notions to match a given, existing
    result. This kind of transformations, often based on a conspicuous
    background knowledge in the given scientific domain (mostly expressed
    in the form of equalities or isomorphisms), are usually implicit in
    the mathematical discourse, and it would be highly desirable to obtain
```

```
    a similar behavior in interactive provers. The paper describes the
    superposition-based implementation of this feature inside the Matita
    interactive theorem prover, focusing in particular on the so called
    smart application tactic, supporting smart matching between a goal and
    a given result.",
  paper = "Aspe10.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Aspe10a,
  author = "Asperti, Andrea and Coen, Claudio Sacerdoti",
  title = {{Some Considerations on the Usability of Interactive Provers}},
  journal = "LNCS",
  volume = "6167",
  year = "2010",
  abstract =
    "In spite of the remarkable achievements recently obtained in the
    field of mechanization of formal reasoning, the overall usability
    of interactive provers does not seem to be sensibly improved since
    the advent of the ‘‘second generation’’ of systems, in the mid of
    the eighties. We try to analyze the reasons of such a slow
    progress, pointing out the main problems and suggesting some
    possible research directions.",
  paper = "Aspe10a.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Aspe11,
  author = "Asperti, Andrea and Ricciotti, Wilmer and Coer, Claudio
            Sacerdoti and Tassi, Enrico",
  title = {{The Matita Interactive Theorem Prover}},
  booktitle = "CADE-23 Automated Deduction",
  year = "2011",
  pages = "64-69",
  abstract =
    "Matita is an interactive theorem prover being developed by the Helm
    team at the University of Bologna. Its stable version 0.5.x may be
    downloaded at http://matita.cs.unibo.it . The tool originated in the
    European project MoWGLI as a set of XML-based tools aimed to provide a
    mathematician-friendly web-interface to repositories of formal
    mathematical knoweldge, supporting advanced content-based
    functionalities for querying, searching and browsing the library. It
    has since then evolved into a fully fledged ITP, specifically designed
    as a light-weight, but competitive system, particularly suited for the
    assessment of innovative ideas, both at foundational and logical
```

```
    level. In this paper, we give an account of the whole system, its
    peculiarities and its main applications.",
  paper = "Aspe11.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Aspe12,
  author = "Asperti, Andrea and Ricciotti, Wilmer and Coer, Claudio
            Sacerdoti and Tassi, Enrico",
  title = {{A Bi-directional Refinement Algorithm for the Calculus
            of (Co)Inductive Constructions}},
  journal = "Logical Methods in Computer Science",
  year = "2012",
  volume = "8",
  pages = "1-49",
  abstract =
    "The paper describes the refinement algorithm for the Calculus of
    (Co)Inductive Constructions (CIC) implemented in the interactive
    theorem prover Matita. The refinement algorithm is in charge of giving
    a meaning to the terms, types and proof terms directly written by the
    user or generated by using tactics, decision procedures or general
    automation. The terms are written in an 'external syntax' meant to be
    user friendly that allows omission of information, untyped binders and
    a certain liberal use of user defined sub-typing. The refiner modifies
    the terms to obtain related well typed terms in the internal syntax
    understood by the kernel of the ITP. In particular, it acts as a type
    inference algorithm when all the binders are untyped. The proposed
    algorithm is bi-directional: given a term in external syntax and a
    type expected for the term, it propagates as much typing information
    as possible towards the leaves of the term. Traditional
    mono-directional algorithms, instead, proceed in a bottom-up way by
    inferring the type of a sub-term and comparing (unifying) it with the
    type expected by its context only at the end. We propose some novel
    bi-directional rules for CIC that are particularly effective. Among
    the benefits of bi-directionality we have better error message
    reporting and better inference of dependent types. Moreover, thanks to
    bi-directionality, the coercion system for sub-typing is more
    effective and type inference generates simpler unification problems
    that are more likely to be solved by the inherently incomplete higher
    order unification algorithms implemented. Finally we introduce in the
    external syntax the notion of vector of placeholders that enables to
    omit at once an arbitrary number of arguments. Vectors of placeholders
    allow a trivial implementation of implicit arguments and greatly
    simplify the implementation of primitive and simple tactics.",
  paper = "Aspe12.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Aspe12a,
  author = "Asperti, Andrea and Ricciotti, Wilmer",
  title = {{A Web Interface for Matita}},
  journal = "LNCS",
  volume = "7362",
  year = "2012",
  paper = "Aspe12a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Aspe12b,
  author = "Asperti, Andrea and Ricciotti, Wilmer and
            Coen, Claudio Sacerdoti and Tassi, Enrico",
  title = {{Formal Metatheory of Programming Languages in the Matita
            Interactive Theorem Prover}},
  journal = "Journal of Automated Reasoning",
  volume = "49",
  number = "3",
  pages = "427-451",
  year = "2012",
  abstract =
    "This paper is a report about the use of Matita, an interactive
    theorem prover under development at the University of Bologna, for
    the solution of the POPLmark Challenges, part 1a. We provide three
    different formalizations, including two direct solutions using
    pure de Bruijn and locally nameless encodings of bound variables,
    and a formalization using named variables, obtained by means of a
    sound translation to the locally nameless encoding. According to
    this experience, we also discuss some of the proof principles used
    in our solutions, which have led to the development of a
    generalized inversion tactic for Matita.",
  paper = "Aspe12b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Aspe85,
  author = "Aspetsberger, K.",
  title = {{Substitution Expressions: Extracting Solutions of non-Horn
            Clause Proofs}},
  booktitle = "European COnference on Computer Algebra",
  publisher = "Springer",
  pages = "78-86",
  year = "1985",
```

```
  comment = "LNCS 204",
  paper = "Aspe85.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Aspi01,
  author = "Aspinall, David and Compagnoni, Adriana",
  title = {{Subtyping Dependent Types}},
  journal = "Theoretical Computer Science",
  volume = "266",
  number = "1-2",
  pages = "273-309",
  year = "2001",
  abstract =
    "The need for subtyping in type-systems with dependent
    types has been realized for some years. But it is hard to
    prove that systems combining the two features have
    fundamental properties such as subject reduction. Here we in-
    vestigate a subtyping extension of the system $\lambda$P
    which is an abstract version of the type system of the
    Edinburgh Logical Framework LF. By using an equivalent
    formulation, we establish some important properties of
    the new system $\lambda{\rm P}_{\le}$ including subject
    reduction. Our analysis culminates in a complete and
    terminating algorithm which establishes the decidability
    of type-checking",
  paper = "Aspi01.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Aspi08,
  author = "Aspinall, David and Denney, Ewen and Luth, Christoph",
  title = {{A Tactic Language for Hiproofs}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
  abstract =
    "We introduce and study a tactic language, Hitac, for constructing
    hierarchical proofs, known as hiproofs. The idea of hiproofs is to
    superimpose a labelled hierarchical nesting on an ordinary proof
    tree. The labels and nesting are used to describe the organisation
    of the proof, typically relating to its construction process. This
    can be useful for understanding and navigating the proof. Tactics
    in our language construct hiproof structure together with an
    underlying proof tree. We provide both a big-step and a small-step
```

```
    operational semantics for evaluating tactic expressions. The
    big-step semantics captures the intended meaning, whereas the
    small-step semantics hints at possible implementations and
    provides a unified notion of proof state. We prove that these
    notions are equivalent and construct valid proofs.",
  paper = "Aspi08.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Atke18,
  author = "Atkey, Robert",
  title = {{Syntax and Semantics of Quantitative Type Theory}},
  booktitle = "LICS",
  year = "2018",
  publisher = "ACM",
  isbn = "978-1-4503-5583-4",
  abstract =
    "We present Quantitative Type Theory, a Type Theory that records
    usage information for each variable in a judgement, based on a
    previous system by McBride. The usage information is used to give
    a realizability semantics using a variant of Linear Combinatory
    Algebras, refining the usual realizability semantics of Type
    Theory by accurately tracking resource behaviour. We define the
    semantics in terms of Quantitative Categories with Families, a
    novel extension of Categories with Families for modelling resource
    sensitive type theories.",
  paper = "Atke18.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Atte15,
  author = "Atten, Mark van and Sundholm, Goran",
  title = {{L.E.J. Brouwer's 'Unrelability of the Logical Principles'
           translation}},
  year = "2015",
  link = "\url{https://arxiv.org/pdf/1511.01113.pdf}",
  abstract =
    "We present a new English translation of L.E.J. Brouwer's paper 'De
    onbetrouwbaarheid der logische principes' (The unreliability of the
    logical principles) of 1908, together with a philosophical and
    historical introduction. In this paper Brouwer for the first time
    objected to the idea that the Principle of the Excluded Middle is
    valid. We discuss the circumstances under which the manuscript was
    submitted and accepted, Brouwer's ideas on the principle of the
    excluded middle, and its consistency and partial validity, and his
```

```
    argument against the possibility of absolutely undecidable
    propositions. We not that principled objections to the general
    exculded middle similar to Brouwer's had been advanced in print by
    Jules Molk two years before. Finally, we discuss the influence on
    George Griss' negationless mathematics",
  paper = "Atte15.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Avig07,
  author = "Avigad, Jeremy",
  title = {{A Formally Verified Proof of the Prime Number Theorem}},
  journal = "ACM Trans. Comput. Logic",
  volume = "9",
  number = "1",
  pages = "2",
  year = "2007"
}
```

———-

— **axiom.bib** —

```
@article{Avig14b,
  author = "Avigad, Jeremy and Harrison, John",
  title = {{Formally Verified Mathematics}},
  journal = "Communications of the ACM",
  volume = "57",
  number = "4",
  pages = "66-75",
  year = "2014",
  abstract =
    "With the help of computational proof assistants, formal
    verification could beome the new standard for rigor in
    mathematics.",
  paper = "Avig14b.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Avig17b,
  author = "Avigad, Jeremy",
  title = {{Proof Theory}},
  year = "2017",
```

```
  abstract =
    "Proof theory began in the 1920's as part of Hilbert's program,
    which aimed to secrure the foundations of mathematics by modeling
    infinitary mathematics with formal axiomatic systems and proving
    those systems consistent using restricted, finitary means. The
    program thus viewed mathematics as a system of reasoning with
    precise linguistic norms, govenered by rules that can be described
    and studied in concrete terms. Today such a viewpoint has
    applications in mathematics, computer science, and the philosophy
    of mathematics.",
  paper = "Avig17b.pdf",
  keywords = "private communication"
}
```

———-

— **axiom.bib** —

```
@misc{Avig18,
  author = "Avigad, Jeremy",
  title = {{Mathematical Language from a Design Perspective}},
  year = "2018",
  link = "\url{http://www.andrew.cmu.edu/user/avigad/Talks/san_diego.pdf}",
  paper = "Avig18.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Avig19,
  author = "Avigad, Jeremy",
  title = {{The Mechanization of Mathematics}},
  year = "2019",
  comment = "slides from The Big Proof Workshop",
  paper = "Avig19.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Avig20,
  author = "Avigad, Jeremy",
  title = {{Mathematical Logic}},
  year = "2020",
  comment = "book preprint",
  paper = "Avig20.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Avro92,
  author = "Avron, Arnon and Honsell, Furio and Mason, Ian A. and
            Pollack, Robert",
  title = {{Using Typed Lambda Calculus to Implement Formal Systems on
            a Machine}},
  year = "1992",
  paper = "Avro92.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Awod04,
  author = "Awodey, Steve and Bauer, Andrej",
  title = {{Propositions as [Types]}},
  year = "2004",
  link = "\url{http://math.andrej.com/asset/data/bracket_types.pdf}",
  abstract =
    "Image factorizations in regular categories are stable under
    pullbacks, as they model a natural modal operator in dependent
    type theory. This unary type constructor [A] has turned up
    previously in a syntactic form as a way of erasing computational
    content, and formalizing a notion of proof irrelevance. Indeed,
    semantically, the notion of a {\sl support} is sometimes used as
    surrogate proposition asserting inhabitation of an indexed family.

    We give rules for bracket types in dependent type theory and
    provide complete semantics using regular categories. We show that
    dependent type theory with the unit type, strong extensional
    equality types, strong dependent sums, and bracket types is the
    internal type theory of regular categories, in the same way that
    the usual dependent type theory with dependent sums and products
    is the internal type theory of locally cartesian closed
    categories.

    We also show how to interpret first-order logic in type theory
    with brackets, and we make use of the translation to compare type
    theory with logic. Specifically, we show that the
    propositions-as-types interpretation is complete with respect to a
    certain fragment of intuitionistic first-order logic, in the sense
    that a formula from the fragment is derivable in intuitionistic
    first-order logic if, and only if, its interpretation in dependent
    type theory is inhabited. As a consequence, a modified
    double-negation translation into type theory (without bracket
    types) is complete, in the same sense, for all of classical
```

```
    first-order logic.",
  paper = "Awod04.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Awod12,
  author = "Awodey, Steve",
  title = {{Category Theory Foundations}},
  year = "2012",
  comment = "Oregon Programming Language Summer School 2012",
  link = "\url{https://www.cs.uoregon.edu/research/summerschool/summer12/curriculum.html}",
}
```

———-

— **axiom.bib** —

```
@misc{Awod16,
  author = "Awodey, Steve",
  title = {{Univalence as a Principle of Logic}},
  year = "2016",
  link = "\url{https://www.andrew.cmu.edu/user/awodey/preprints/ualp.pdf}",
  abstract =
    "It is sometmes convenient or useful in mathematics to treat
    isomorphic structures as the same. The recently proposed
    Univalence Axiom for the foundations of mathematics elevates this
    idea to a foundational principle in the setting of Homotopy Type
    Theory. It states, roughly, that isomorphic structures can be
    identified. We explore the motivations and consequences, both
    mathematical and philosophical, of making such a new logical
    postulate.",
  paper = "Awod16.pdf",
  keywords = "printed"
}
```

———-

1.2.2 B

— **axiom.bib** —

```
@misc{Bacc20,
  author = "Baccala, Brent W.",
  title = {{The Facebook Integral}},
  year = "2020",
  link = "\url{https://ec2.freesoft.org/blogs/soapbox/the-facebook-integral}",
```

```
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Bacc20a,
  author = "Baccala, Brent W.",
  title = {{The Facebook Integral (solved with Axiom and Sage}},
  year = "2020",
  link = "\url{https://youtu.be/tz1LwfJlMuo}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Bacc20b,
  author = "Baccala, Brent W.",
  title = {{The Risch Theorem}},
  year = "2020",
  link = "\url{https://youtu.be/BGnLwhXb204}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Back88,
  author = "Backhouse, Roland and Chisholm, Paul and Malcolm, Grant
            and Saaman, Erik",
  title = {{Do-it-yourself Type Theory}},
  year = "1988",
  link = "\url{www.cs.nott.ac.uk/~psarb2/MPC/DOYTypeTheory.pdf}",
  paper = "Back88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Back08,
  author = "Backeljauw, Franky and Becuwe, Stefan and Cuyt, Annie",
  title = {{Validated Evaluation of Special Mathematical Functions}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
```

```
  abstract =
    "Because of the importance of special functions, several books and
    a large collection of papers have been devoted to the numerical
    computation of these functions, the most well-known being the
    Abramowitz and Stegun handbook. But up to this date, no
    environment offers routines for the provable correct evaluation of
    these special functions.

    We point out how series and limit-periodic continued fraction
    representation of the functions can be helpful in this
    respect. Our scalable precision technique is mainly based on the
    use of sharpened a priori truncation and round-off error upper
    bounds, in case of real arguments. The implementation is validated
    in the sense that it returns a sharp interval enclosure for the
    requested function evaluation, at the same cost as the evaluation.",
  paper = "Back08.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Baez09,
  author = "Baez, John C.; Stay, Mike",
  title = {{Physics, Topology, Logic and Computation: A Rosetta Stone}},
  link = "\url{http://arxiv.org/pdf/0903.0340v3.pdf}",
  abstract =
    "In physics, Feynman diagrams are used to reason about quantum
    processes.  In the 1980s, it became clear that underlying these
    diagrams is a powerful analogy between quantum physics and
    topology. Namely, a linear operator behaves very much like a
    ‘‘cobordism’’: a manifold representing spacetime, going between two
    manifolds representing space.  But this was just the beginning: simiar
    diagrams can be used to reason about logic, where they represent
    proofs, and computation, where they represent programs. With the rise
    of interest in quantum cryptography and quantum computation, it became
    clear that there is an extensive network of analogies between physics,
    topology, logic and computation. In this expository paper, we make
    some of these analogies precise using the concept of ‘‘closed
    symmetric monodial category’’. We assume no prior knowledge of
    category theory, proof theory or computer science.",
  paper = "Baez09.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bagn19,
  author = "Bagnara, Roberto and Bagnara, Abramo and Biselli, Fabio
            and Chiari, Michele and Gori, Roberta",
```

```
  title = {{Correct Approximation of IEEE 754 Floating-Point
           Arithmetic for Program Verification}},
  year = "2019",
  link = "\url{https://arxiv.org/abs/1903.06119}",
  abstract =
    "Verification of programs using floating-point arithmetic is
    challenging on several accounts. One of the difficulties of
    reasoning about such programs is due to the peculiarities of
    floating-point arithmetic: rounding errors, infinities,
    non-numeric objects (NaNs), signed zeros, denormal numbers,
    different rounding modes... One possibility to reason about
    floating-point arithmetic is to model a program computation path
    by means of a set of ternary constraints of the form $z=x op y$
    and use constraint propagation techniques to infer new information
    on the variables' possible values. In this setting, we define and
    prove the correctness of algorithms to precisely bound the value
    of one of the variables $x$, $y$, or $z$, starting from the bounds
    known for the other two. We do this for each of the operations and
    for each rounding mode defined by the IEEE 754 binary
    floating-point standard, even in the case the rounding mode in
    effect is only partially known. This is the first time that such
    so-called filtering algorithms are defined and their correctness
    is formally proved. This is an important slab for paving the way
    to formal verification of programs that use floating-point
    arithmetics.",
  paper = "Bagn19.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Bahr15,
  author = "Bahr, Patrick and Hutton, Graham",
  title = {{Calculating Correct Compilers}},
  journal = "Functional Programming",
  year = "2015",
  link = "\url{www.cs.nott.ac.uk/~pszgmh/ccc.pdf}",
  abstract =
    "In this article we present a new approach to the problem of
    calculating compilers. In particular, we develop a simple but
    general technique that allos us to derive correct compilers from
    high-level semantics by systematic calculation, with all details
    of the implementation of the compilers falling naturally out of
    the calculation process. Our approach is based upon the use of
    standard equational reasoning techniques, and has been applied to
    calculate compilers for a wide range of language features and
    their combination, including arithmetic expressions, exceptions,
    state, various forms of lambda calculi, bounded and unbounded
    loops, non-determinism, and interrupts. All the calculations in
    the article have been formalised using the Coq proof assistant,
    which serves as a convenient interactive tool for developing and
```

```
    verifying the calculations.",
  paper = "Bahr15.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Bahr19,
  author = "Bahr, Patrick and Hutton, Graham",
  title = {{Calculating Correct Compilers II}},
  journal = "Functional Programming",
  year = "2019",
  link = "\url{www.cs.nott.ac.uk/~pszgmh/ccc2.pdf}",
  abstract =
    "In 'Calculating Correct Compilers' (Bahr and Hutton, 2015) we
    developed a new approach to calculating compilers directly from
    specifications of their correctness. Our approach only required
    elementary reasoning techniques, and has been used to calculate
    compilers for a wide range of language features and their
    combination. However, the methodology was focused on stack-based
    target machines, whereas real compilers often target
    register-based machines. In this article, we show how our approach
    can naturally be adapted to calculate compilers for register
    machines.",
  paper = "Bahr19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bail19,
  author = "Bailleux, Olivier",
  title = {{Subsumption-driven Clause Learning with DPLL+Restarts}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1906.07508.pdf}",
  abstract =
    "We propose to use a DPLL+restart to solve SAT instances by
    successive simplifications based on the production of clauses that
    subsume the initial clauses. We show that this approach allows the
    refutation of pebbling formulae in polynomial time and linear
    space, as effectvely as with a CDCL solver.",
  paper = "Bail19.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bake93,
  author = "Bakel, Steffan van",
  title = {{Principal Type Schemes for the Strict Type Assignment System}},
  journal = "J. Logic and Computation",
  volume = "3",
  number = "6",
  pages = "643-670",
  year = "1993",
  abstract =
    "We study the strict type assignment system, a restriction on the
    intersection type discipline and prove that it has the principal
    type property. W define, for a term $M$, the principal pair (of
    basis and type). We specify three operations on pairs, and prove
    that all pairs deducible for $M$ can be obtained from the
    principal one by these operations, and that these map deducible
    pairs to deducible pairs.",
  paper = "Bake93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bake84,
  author = "Baker, Henry",
  title = {{The Nimble Type Inferencer for Common Lisp-84}},
  year = "1984",
  link = "\url{http://home.pipeline.com/~hbaker1/TInference.html}",
  abstract =
    "We describe a framework and an algorithm for doing type inference
    analysis on programs written in full Common Lisp-84 (Common Lisp
    without the CLOS object-oriented extensions). The objective of
    type inference is to determine tight lattice upper bounds on the
    range of runtime data types for Common Lisp program variables and
    temporaries. Depending upon the lattice used, type inference can
    also provide range analysis information for numeric
    variables. This lattice upper bound information can be used by an
    optimizing compiler to choose more restrictive, and hence more
    efficient, representations for these program variables. Our
    analysis also produces tighter control flow information, which can
    be used to eliminate redundant tests which result in dead
    code. The overall goal of type inference is to mechanically
    extract from Common Lisp programs the same degree of
    representation information that is usually provided by the
    programmer in traditiional strongly-typed languages. In this way,
    we can provide some classes of Common Lisp programs execution time
    efficiency expected only for more strongly-typed compiled languages.

    The Nimble type inference system follows the traditional
    lattice/algebraic data flow techniques [Kaplan80], rather than the
    logical/theorem-proving unification techniques of ML [Milner78]. It
    can handle polymorphic variables and functions in a natural way, and
```

```
    provides for ``case-based'' analysis that is quite similar to that used
    intuitively by programmers. Additionally, this inference system can
    deduce the termination of some simple loops, thus providing
    surprisingly tight upper lattice bounds for many loop variables.

    By using a higher resolution lattice, more precise typing of primitive
    functions, polymorphic types and case analysis, the Nimble type
    inference algorithm can often produce sharper bounds than
    unification-based type inference techniques. At the present time,
    however, our treatment of higher-order data structures and functions
    is not as elegant as that of the unification techniques.",
  paper = "Bake84.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Bake91,
  author = "Baker, Henry G.",
  title = {{Pragmatic Parsing in Common Lisp}},
  journal = "ACM Lisp Pointers",
  volume = "IV",
  number = "2",
  pages = "3-15",
  year = "1991",
  abstract =
    "We review META, a classic technique for building recursive descent
    parsers, that is both simple and efficient.  While META does not
    handle all possible regular or context-free grammars, it handles a
    surprisingly large fraction of the grammars encountered by Lisp
    programmers.  We show how META can be used to parse streams, strings
    and listsincluding Common Lisp's hairy lambda expression parameter
    lists.  Finally, we compare the execution time of this parsing method
    to the built-in methods of Common Lisp.",
  paper = "Bake91.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Bake91a,
  author = "Baker, Henry",
  title = {{CLOStrophobia: Its Etiology and Treatment}},
  journal = "OOPS Messenger",
  volume = "2",
  number = "4",
  publisher = "ACM",
  year = "1991",
```

```
  pages = "4-15",
  abstract =
    "The Common Lisp Object System (CLOS) has received some praise and
    some criticism, both deserved. One of the most controversial
    features of standard CLOS is its linearly-ordered 'class
    precedence list', which is used to linearly order the execution of
    its 'combination' methods. In addition to the problems already
    known regarding the linear ordering of superclasses, we show that
    the standard CLOS class precedence ordering produces gratuitously
    complex and non-intuitive behavior. We then show that a slight
    modification of the standard ordering rules produces a linear
    ordering which can achieve most of the goals of CLOS more
    efficiently, and without impacting most programs. We describe a
    subset of CLOS called {\sl Static} CLOS which preserves much of
    the praise due CLOS, while eliminating some of the
    criticism. Static CLOS is tuned for {\sl deliver} of debugged
    code, rather than for prototype development. This 'delivery' CLOS
    determines as many methods as possible at compile-time using 'type
    inference' techniques. While these techniques generally result in
    faster-executing code, the space requirements can grow quite
    large. We argue that this space explosion can be partially
    ameliorated through the use of our modified class precedence
    ordering.",
  paper = "Bake91a.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Bake92,
  author = "Baker, Henry G.",
  title = {{A Decision Procedure for Common Lisp's SUBTYPEP Predicate}},
  journal = "Lisp and Symbolic Computation",
  volume = "5",
  number = "3",
  year = "1992",
  pages = "157-190",
  abstract =
    "Common Lisp [CL84,CL90] includes a dynamic datatype system of
    moderate complexity, as well as predicates for checking the types of
    language objects. Additionally, an interesting predicate of two 'type
    specifiers' SUBTYPEP is included in the language. This subtypep
    predicate provides a mechanism with which to query the Common Lisp
    type system regarding containment relations among the various built-in
    and user-defined types. While subtypep is rarely needed by an
    applications programmer, the efficiency of a Common Lisp
    implementation can depend critically upon the quality of its subtypep
    predicate: the run-time system typically calls upon subtypep to decide
    what sort of representations to use when making arrays; the compiler
    calls upon subtypep to interpret user declarations, on which efficient
    data representation and code generation decisions are based.
```

```
    As might be expected due to the complexity of the Common Lisp type
    system, there may be type containment questions which cannot be
    decided. In these cases subtypep is expected to return 'can't
    determine', in order to avoid giving an incorrect
    answer. Unfortunately, most Common Lisp implementations have abused
    this license by answering 'can't determine' in all but the most
    trivial cases. In particular, most Common Lisp implementations of
    SUBTYPEP fail on the basic axioms of the Common Lisp type system
    itself [CL84,p.33]. This situation is particularly embarrassing for
    Lispthe premier 'symbol processing language'in which the
    implementation of complex symbolic logical operations should be
    relatively easy. Since subtypep was presumably included in Common Lisp
    to answer the hard cases of type containment, this 'lazy evaluation'
    limits the usefulness of an important language feature.

    This paper shows how those type containment relations of Common Lisp
    which can be decided at all, can be decided simply and quickly by a
    decision procedure which can dramatically reduce the number of
    occurrences of the 'can't determine' answer from subtypep . This
    decision procedure does not require the conversion of a type specifier
    expression to conjunctive or disjunctive normal form, and therefore
    does not incur the exponential explosion in space and time that such a
    conversion would entail.

    The lattice mechanism described here for deciding subtypep is also
    ideal for performing type inference [Baker90]; the particular
    implementation developed here, however, is specific to the type system
    of Common Lisp [Beer88].",
  paper = "Bake92.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@misc{Bake93a,
  author = "Baker, Henry G.",
  title = {{Linear Logic and Permutation Stacks -- The Forth Shall Be First}},
  year = "1993",
  link = "\url{http:/home.pipeline.com/~hbaker1/ForthStack.html}",
  abstract =
    "Girard's linear logic can be used to model programming languages
    in which each bound variable name has exactly one ``occurence'' --
    i.e., no variable can have implicit ``fan-out'', multiple uses
    require explicit duplication. Among other nice properties,
    ``linear'' languages need no garbage collector, yet have no
    dangling reference problems. We show a natural equivalence between
    a ``linear'' programming language and a stack machine in which the
    top items can undergo arbitrary permutations. Such permutation
    stack machines can be considered combinator abstractions of
    Moore's {\sl Forth} programming language.",
```

```
  paper = "Bake93a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bake93b,
  author = "Baker, Henry G.",
  title = {{Equal Rights for Functional Object or, The More Things
            Change, the More They Are the Same}},
  journal = "OOPS",
  publisher = "ACM",
  pages = "2-27",
  year = "1993",
  abstract =
    "We argue that intensional {\sl object identity} in
    object-oriented programming languages and databases is best
    defined operationally by sode-effect semantics. A corollary is
    that ``functional'' objects have extensional semantics. This model
    of object identity, which is analogous to the ormal forms of
    relational algebra, provides cleaner semantics for the
    value-transmission operations and built-in primitive equality
    predicate of a programming language, and eliminates the confusion
    surrounding ``call-by-value'' and ``call-by-reference'' as well as
    the confusion of multiple equality predicates.

    Implementation issues are discusssed, and this model is shown to
    have significant performance advantages in persistent, parallel,
    distributed and multilingua processing environments. This model
    also provides insight into the ``type equivalence'' problem of
    Algol-68, Pascal and Ada.",
  paper = "Bake93b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Bake09,
  author = "Baker, Josef B. and Sexton, Alan P. and Sorge, Volker",
  title = {{A Linear Grammar Approach to Mathematical Formula
            Recognition from PDF}},
  journal = "LNCS",
  volume = "5625",
  year = "2009",
  abstract =
    "Many approaches have been proposed over the years for the
    recognition of mathematical formulae from scanned documents. More
    recently a need has arisen to recognise formulae from PDF
    documents. Here we can avoid ambiguities introduced by traditional
```

```
    OCR approaches and instead extract perfect knowledge of the
    characters used in formulae directly from the document. This can
    be exploited by formula recognition techniques to achieve correct
    results and high performance.

    In this paper we revist an old grammatical approach to formula
    recognition, that of Anderson from 1968, and assess its
    applicability with respect to data extracted from PDF
    documents. We identify some problems of the original method when
    applied to common mathematical expressions and show how they can
    be overcome. The simplicity of the original method leads to a very
    efficient recognition technique that not only is very simple to
    implement but also yields results of high accuracy for the
    recognition of mathematical formulae from PDF documents.",
  paper = "Bake09.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Bake16b,
  author = "Baker, Martin",
  title = {{add coerce from PermutationGroup to GroupPresentation}},
  link = "\url{https://groups.google.com/forum/?hl=en\#!topic/fricas-devel/EtLwgd2dWNU}",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@misc{Bake17,
  author = "Baker, Martin",
  title = {{Finite Group Implementation}},
  link = "\url{http://www.euclideanspace.com/prog/scratchpad/mycode/discrete/finiteGroup/}",
  year = "2017"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Balz17,
  author = "Balzer, Stephanie and Pfenning, Frank",
  title = {{Manifest Sharing with Session Types}},
  booktitle = "Proc. ACM Program. Lang.",
  publisher = "ACM",
  year = "2017",
  link = "\url{https://www.cs.cmu.edu/~fp/papers/icfp17.pdf}",
```

```
  abstract =
    "Session-typed languages building on the Curry-Howard isomorphism
    between linear logic and session-typed communication guarantee session
    fidelity and deadlock freedom. Unfortunately, these strong guarantees
    exclude many naturally occurring programming patterns pertaining to
    shared resources. In this paper, we introduce sharing into a
    session-typed language where types are stratified into linear and
    shared layers with modal operators connecting the layers. The
    resulting language retains session fidelity but not the absence of
    deadlocks, which can arise from contention for shared processes. We
    illustrate our language on various examples, such as the dining
    philosophers problem, and provide a translation of the untyped
    asynchronous $\Pi$-calculus into our language.",
  paper = "Balz17.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Balz18,
  author = "Balzer, Stephanie and Pfenning, Frank and Toninho, Bernardo",
  title = {{A Universal Session Type for Untyped Asynchronous Communication}},
  journal = "CONCUR",
  volume = "30",
  number = "1",
  year = "2018",
  link = "\url{https://www.cs.cmu.edu/~fp/papers/univtype18.pdf}",
  abstract =
    "In the simply-typed $\lambda$-calculus we can recover the full
    range of expressiveness of the untyped $\lambda$-calculus solely
    by adding a single recursive type
    $U = U\rightarrow U$. In contract,
    in the session-typed $\Pi$-calculus, recursion alone is
    insufficient to recover the untyped $\Pi$-calculus, primarily due
    to linearity: each channel just has two unique endpoints. In this
    paper, we show that shared channels with a corresponding sharing
    semantics (base on the the language SILL$_s$ developed in prior
    work) are enough to embed the untyped asynchronous $\Pi$-calculus
    via a universal shared session type $U_s$. We show that
    our encoding of the asynchronous $\Pi$-calculus satisfies
    operational correspondence and preserves observable actions (i.e.,
    processes are weakly bisimilar to their encoding). Moreover, we
    clarify the expressiveness of SILL$_s$ by developing an
    operationally correct encoding of SILL$_s$ in the asynchronous
    $\Pi$-calculus.",
  paper = "Balz18.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Barb00,
  author = "Barbeau, Edward J.",
  title = {{Mathematical Fallacies, Flaws, and Flimflam}},
  publisher = "American Mathematical Society",
  year = "2000",
  paper = "Barb00.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Bare84,
  author = "Barendregt, H. P.",
  title = {{The Lambda Calculus: Its Syntax and Semantics}},
  publisher = "Elsevier Science",
  year = "1984",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Bare91,
  author = "Barendregt, Hendrik Pieter",
  title = {{An Introduction to Generalized Type Systems}},
  journal = "Journal of Functional Programming",
  volume = "1",
  number = "2",
  year = "1991",
  pages = "125-154",
  abstract =
    "Programming languages often come with type systems. Some of these are
    simple, others are sophisticated. As a stylistic representation of
    types in programming languages several versions of typed lambda
    calculus are studied. During the last 20 years many of these systems
    have appeared, so there is some need of classification. Working
    towards a taxonomy, Barendregt (1991) gives a fine-structure of the
    theory of constructions (Coquand and Huet 1988) in the form of a
    canonical cube of eight type systems ordered by inclusion. Berardi
    (1988) and Terlouw (1988) have independently generalized the method of
    constructing systems in the -cube. Moreover, Berardi (1988, 1990)
    showed that the generalized type systems are flexible enough to
    describe many logical systems. In that way the well-known
    propositions-as-types interpretation obtains a nice canonical form.",
  paper = "Bare91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bare97,
  author = "Barendregt, Henk",
  title = {{The Impact of the Lambda Calculus}},
  link = "\url{http://www-users.mat.umk.pl/~adwid/materialy/doc/church.pdf}",
  year = "1997",
  paper = "Bare97.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Barr79,
  author = "Barrett, William A. and Couch, John D.",
  title = {{Compiler Construction: Theory and Practice}},
  publisher = "Science Research Associates",
  year = "1979",
  isbn = "0-574-21335-X",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Bart85,
  author = "Barthe, Gilles",
  title = {{Implicit Coercions in Type Systems}},
  journal = "LNCS",
  volume = "1158",
  pages = "1-15",
  year = "1985",
  abstract =
    "We propose a notion of pure type system with implicit
    coercions. In our framework, judgements are extended with a
    context of coerions $\Delta$ and the application rule is modified
    so as to allow coercions to be left implicit. The setting supports
    multiple inheritance and can be applied to all type theories with
    $\Pi$-types. One originality of our work is to propose a
    computational interpretation of implict coercions. In this paper,
    we demonstrate how this interpretation allows a strict control on
    the logical properties of pure type systems with implicit coercions.",
  paper = "Bart85.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Bart85a,
  author = "Barthe, Gilles and Ruys, Mark and Barendregt, Henk",
  title = {{A Two-Level Approach Towards Lean Proof-Checking}},
  journal = "LNCS",
  volume = "1158",
  pages = "16-35",
  year = "1985",
  abstract =
    "We present a simple and effective methodology for equational
    reasoning in proof checkers. The method is based on a two-level
    approach distinguishing between syntax and semantics of
    mathematical theories. The method is very general and can be
    carried out in any system with inductive and oracle types. The
    potential of our two-level approach is illustrated by some
    examples developed in Lego.",
  paper = "Bart85a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bart96,
  author = "Barthe, G. and Elbers, H.",
  title = {{Towards Lean Proof Checking}},
  year = "1996",
  abstract =
    "Logical formal systems are inefficient at computations. In order
    to increase their efficiency, we aim to extend these systems with
    computational power. In this paper, we suggest a general, powerful
    syntax, called oracle types, to extend type theories with
    computational power; the resulting systems, which combine the
    logical abilities of logical formal systems and the computational
    power of term rewriting systems, provide a suitable environment
    for theorem proving. As a practical application, we present an
    extension of the theorem prover Lego with oracle types and
    illustrate the use of this new system in performing algebraic
    computations. Our implementation of oracle types is very flexible
    and allows rewriting to be performed either inside Lego or by
    Reduce, an efficient symbolic computation system. In our view, the
    main novelty of our approach is to combine a sound theoretical
    foundation with an efficient implementation. Besides, our work
    provides the first attempt to combine symbolic computation systems
    with theorem provers such as Coq and Lego, which are based on
    intensional type theories.",
  paper = "Bart96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Barw96,
  author = "Barwise, Jon and Moss, Lawrence",
  title = {{Vicious Circles}},
  publisher = "CSLI Publications",
  year = "1996",
  isbn = "1-57586-008-2",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Bate85,
  author = "Bates, Joseph L. and Constable, Robert L.",
  title = {{Proofs as Programs}},
  journal = "ACM TOPLAS",
  volume = "7",
  number = "1",
  year = "1985",
  abstract =
    "The significant intellectual cost of programming is for problem
    solving and explaining, not for coding. Yet programming systems offer
    mechanical assistance for the coding process exclusively. We
    illustrate the use of an implemented program development system,
    called PRL ('pearl'), that provides automated assistance with the
    difficult part. The problem and its explained solution are seen as
    formal objects in a constructive logic of the data domains. These
    formal explanations can be executed at various stages of
    completion. The most incomplete explanations resemble applicative
    programs, the most complete are formal proofs.",
  paper = "Bate85.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Baue07,
  author = "Bauer, Andrej and Stone, Christopher A.",
  title = {{RZ: a Tool for Bringing Constructive and Computational
            Mathematics Closer to Programming Practice}},
  year = "2007",
  link = "\url{http://math.andrej.com/wp-content/uploads/2007/01/cie-long.pdf}",
  abstract =
    "Realizability theory is not only a fundamental tool in logic and
    computability, but also has direct application to the design and
```

```
    implementation of programs: it can produce interfaces for the data
    structure corresponding to a mathematical theory. Our tool, called
    RZ, serves as a bridge between the worlds of constructive
    mathematics and programming. By using the realizability
    interpretation of constructive mathematics, RZ translates
    specifications in constructive logic into annotated interface code
    in Objective Caml. The system supports a rich input language
    allowing descriptions of complex mathematical structures. RZ does
    not extract code from proofs, but allows any implementation
    method, from handwritten code to code extracted from proofs by
    other tools.",
  paper = "Baue07.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Baue19,
  author = "Bauer, Andrej",
  title = {{How to Implement Type Theory in an Hour}},
  year = "2019",
  link = "\url{https://vimeo.com/286652934}",
  comment = "\url{https://github.com/andrejbauer/spartan-type-theory}"
}
```

———-

— **axiom.bib** —

```
@misc{Baue19a,
  author = "Bauer, Andrej",
  title = {{Derivations as  computations}},
  year = "2019",
  link = "\url{https://youtube.com/watch?v=YZqOVsuyQyQ}"
}
```

———-

— **axiom.bib** —

```
@misc{Baue20,
  author = "Bauer, Andrej and Haselwarter, Philipp G. and
            Lumsdaine, Peter LeFanu",
  title = {{A General Definition of Dependent Type Theories}},
  year = "2020",
  link = "\url{https://arxiv.org/pdf/2009.05539.pdf}",
  abstract =
    "We define a general class of dependent type theories,
    encompassing Martin Lof's intuitionistic type theories and
```

```
    variants and extensions. The primary aim is pragmatic: to unify
    and organise their study, allowing results and constructions to be
    given in reasonable generality, rather than just for specific
    theories. Compared with other approaches, our definition stays
    closer to the direct or naive reading of syntax, yielding the
    traditional presentation of specific theories as closely as
    possible.

    Specifically, we give three main definitions: {\sl raw type
    theories}, a minimal setup for discussing dependently typed
    derivability; {\sl acceptable type theories}, including extra
    conditions ensuring well-behavedness; and {\sl well-presented type
    theories}, generalisig how in traditional presentations, the
    well-behavedness of a type theory is established step by step as
    the type theory is built up. Following these, we show that various
    fundamental fitness-for-purpose metatheorems hold in this
    generality.

    Much of the present work has been formalised in the proof
    assistant Coq.",
  paper = "Baue20.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Baum95,
  author = "Baumgartner, Gerald and Stansifer, Ryan D.",
  title = {{A Proposal to Study Type Systems for Computer Algebra}},
  type = "technical report",
  institution = "RISC-LINZ",
  number = "90-07.0",
  year = "1995",
  abstract =
    "It is widely recognized that programming languages should offer
    features to help structure programs. To achieve this goal, languages
    like Ada , Modula-2 , object-oriented languages, and functional
    languages have been developed. The structuring techniques available
    so far (like modules, classes, parametric polymorphism) are still not
    enough or not appropriate for some application areas. In symbolic
    computation, in particular computer algebra, several problems occur
    that are dicult to handle with any existing programming
    language. Indeed, nearly all available computer algebra systems suer
    from the fact that the underlying programming language imposes too
    many restrictions.

    We propose to develop a language that combines the essential features
    from functional languages, object-oriented languages, and computer
    algebra systems in a semantically clean manner. Although intended for
    use in symbolic computation, this language should prove interesting as
    a general purpose programming language.  The main innovation will be
```

```
    the application of sophisticated type systems to the needs of computer
    algebra systems. We will demonstrate the capabilities of the language
    by using it to implement a small computer algebra library. This
    implementation will be compared against a straightforward Lisp
    implementation and against existing computer algebra systems. Our
    development should have an impact both on the programming languages
    world and on the computer algebra world.",
  paper = "Baum95.pdf",
  keywords = "printed, axiomref"

}
```

———-

— **axiom.bib** —

```
@misc{Baze14,
  author = "Bazerman, Gershom",
  title = {{Homotopy Type Theory: What's the Big Idea}},
  year = "2014",
  comment = "Lambda Jam 2014",
  link = "\url{https://www.youtube.com/watch?v=OupcXmLER7I}"
}
```

———-

— **axiom.bib** —

```
@article{Beer87,
  author = "Beer, Randall D.",
  title = {{Preliminary Report on A Practical Type Inference System
            for Common Lisp}},
  journal = "Lisp Pointers",
  volume = "1",
  number = "2",
  year = "1987",
  pages = "5-11",
  paper = "Beer87.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bend10,
  author = "Bendersky, Eli",
  title = {{Top Down Operator Precedence}},
  year = "2010",
  link = "\url{https://eli.thegreenplace.net/2010/01/02/top-down-operator-precedence-parsing}",
  keywords = "DONE"
```

```
}
```

———-

— **axiom.bib** —

```
@book{Benk99,
  author = "Benker, Hans",
  title = {{Practical User of MATHCAD: Solving Mathematical Problems
            with a Computer Algebra System}},
  publisher = "Springer-Verlag",
  year = "1999",
  isbn = "978-1-85233-166-5",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{DDMF19,
  author = "Benoit, Alexandre and Chyzak, Frederic and Darrasse, Alexis
            and Gregoire, Thomas and Koutschan, Christoph and
            Mezzarobba, Marc and Salvy Bruno",
  title = {{Digital Dictionary of Mathematical Functions}},
  year = "2019",
  link = "\url{ddfm.msr-inria.inria.fr/1.9.1/ddmf}",
  abstract =
    "Interactive site on Mathematical Functions with properties,
    truncated expansions, numerical evaluations, plots, and more. The
    functions currently presented are elementary functions with
    special functions of a single variable. More functions -- special
    functions with parameters, orthogonal polynomials, sequences --
    will be added with the project advances.",
  paper = "DDMF19.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bent93,
  author = "Benthem Jutting, L.S. van and McKinna, J. and Pollack, R.",
  title = {{Checking Algorithms for Pure Type Systems}},
  booktitle = "Types for Proofs and Programs",
  publisher = "Springer",
  year = "1993",
  pages = "19-61",
  abstract =
    "This work is motivated by the problem of finding reasonable
```

```
    algorithms for typechecking Pure Type Systems [Bar91] (PTS). There
    are several implementations of formal systems that are either PTS
    or closely related to PTS. For example, LEGO [LP92] implements the
    Pure Calculus of Constructions (PCC) [CH88], the Extended Calculus
    of Constructions [Luo90] and the Edinburgh Logical Framework (LF)
    [HHP87]. ELF [Pfe89] implements LF; CONSTRUCTOR [Hel91] implements
    arbitrary PTS with finite set of sorts. Are these implementations
    actually correct? Of course, we may enumerate all derivations of a
    given PTS, and Jutting [vBJ93] has shown that a large class of
    normalizing PTS have decidable typechecking by computing the
    normal forms of types, but such techniques are obviously not
    usable in practice. Algorithms in the literature for particular
    type systems, such as Huet's Constructive Engine [Hue89], do not
    obviously extend even to such tame classes as the normalizing and
    functional PTS.",
  paper = "Bent93.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Berg92,
  author = "Berger, Emery",
  title = {{FP + OOP = Haskell}},
  institution = "University of Texas",
  number = "TR-92-30",
  year = "1992",
  abstract =
    "The programming language Haskell adds object-oriented functionality
    (using a concept known as type classes) to a pure functional
    programming framework. This paper describes these extensions and
    analyzes its accomplishments as well as some problems."
}
```

———-

— **axiom.bib** —

```
@article{Berg85,
  author = "Berger, U. and Schwichtenberg, H.",
  title = {{The Greatest Common Divisor: A Case Study for Program
            Extraction from Classical Proofs}},
  journal = "LNCS",
  volume = "1158",
  pages = "36-46",
  year = "1985",
  paper = "Berg85.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bern19,
  author = "Bernstein, Daniel J. and Yang, Bo-Yin",
  title = {{Fast Constant-Time GCD and Modular Inversion}},
  year = "2019",
  link = "\url{https://gcd.crypto.to/safegcd-20190413.pdf}",
  abstract =
    "This paper introduces streamlined constant-time variants of
    Euclid's algorithm, both for polynomial inputs and for integer
    inputs. As concrete applications, this paper saves time in (1)
    modular inversion for Curve25519, which was previously believed to
    be handled much more efficiently by Fermat's method, and (2) key
    generation for the ntruhrss701 and sntrup4591761 lattice-based
    cryptosystems.",
  paper = "Bern19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bert08,
  author = "Bertot, Yves and Gonthier, Georges and Biha, Sidi Ould and
            Pasca, Ioana",
  title = {{Canonical Big Operators}},
  booktitle = "Theorem Proving in Higher Order Logics",
  publisher = "Springer",
  pages = "86-101",
  year = "2008"
}
```

———-

— **axiom.bib** —

```
@misc{Bhat19,
  author = "Bhat, Siddharth",
  title = {{Computing Equivalent Gate Sets Using Grobner Bases}},
  link = "\url{https://bollu.github.io/#computing-equivalent-gat-sets-using-grobner-bases}",
  year = "2019"
}
```

———-

— **axiom.bib** —

```
@article{Biha09,
  author = "Biha, Sidi Ould",
  title = {{Finite Group Representation Theory with Coq}},
```

```
  journal = "LNCS",
  volume = "5625",
  year = "2009",
  abstract =
    "Representation theory is a branch of algebra that allows the
    study of groups through linear applications, i.e. matrices. Thus
    problems in abstract groups can be reduced to problems on
    matrices. Representation theory is the basis for character
    theory. In this paper we present a formalization of finite groups
    representation theory in the Coq system that includes a
    formalization of Maschke's theorem on reducible finite group algebra.",
  paper = "Biha09.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bini85,
  author = "Bini, Dario and Pan, Victor",
  title = {{Algorithms for Polynomial Division}},
  booktitle = "European COnference on Computer Algebra",
  publisher = "Springer",
  pages = "1-3",
  year = "1985",
  comment = "LNCS 204",
  paper = "Bini85.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Birk35,
  author = "Birkhoff, Garrett",
  title = {{On the Structure of Abstract Algebra}},
  journal = "Proc. of the Cambridge Philosophical Society",
  volume = "31",
  year = "1935",
  paper = "Birk35.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Birt80,
  author = "Birtwistle, Graham M.",
  title = {{Simula Begin}},
  year = "1980",
  publisher = "Chartwell-Bratt",
```

```
  isbn = "9780862380090"
}
```

———-

— **axiom.bib** —

```
@book{Bish12,
  author = "Bishop, Errett",
  title = {{Foundations of Constructive Analysis}},
  publisher = "ISHI Press",
  year = "2012",
  isbn = "978-4-87187-714-5",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Bitt01,
  author = "Bittner, Calvin John and Grossman, Bertrand M. and
            Jenks, Richard Dimick and Watt, Stephen Michael and
            Williams, Richard Quimby",
  title = {{Computer-Program Compilers Comprising a Program
            Augmentation Capability}},
  link = "\url{https://cs.uwaterloo.ca/~smwatt/pub/reprints/2001-us-6223341.pdf}",
  comment = "U.S. Patent 6,223,341",
  paper = "Bitt01.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Blac80,
  author = "Black, A.P.",
  title = {{Exception Handling and Data Abstraction}},
  type = "Research Report",
  institution = "IBM Research",
  number = "RC8059",
  year = "1980"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Blak96,
  author = "Blakley, Bob",
  title = {{The Emperor's Old Armor}},
```

```
  booktitle = "Proc. 1996 New Security Paradigms Workshop",
  publisher = "ACM",
  year = "1996",
  paper = "Blak96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Blan15,
  author = "Blanchette, Jasmin Christian and Haslbeck, Maximilian and
            Matichuk, Daniel and Nipkow, Tobias",
  title = {{Mining the Archive of Formal Proofs}},
  journal = "LNCS",
  volume = "9150",
  year = "2015",
  abstract =
    "The Archive of Formal Proofs is a vast collection of
    computer-checked proofs developed using the proof assistant
    Isabelle. We perform an in-depth analysis of the archive, looking
    at various properties of the proof developments, including size,
    dependencies, and proof style. This gives some insights into the
    nature of formal proofs",
  paper = "Blan15.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Blan05,
  author = "Blanqui, Frederic",
  title = {{Inductivev Types in the Calculus of Algebraic Constructions}},
  journal = "Fundamenta Informaticae",
  volume = "65",
  number = "1-2",
  pages = "61-86",
  year = "2005",
  abstract =
    "In a previous work, we proved that an important part of the Calculus
    of Inductive Constructions (CIC), the basis of the Coq proof
    assistant, can be seen as a Calculus of Algebraic Constructions
    (CAC), an extension of the Calculus of Constructions with functions
    and predicates defined by higher-order rewrite rules.  In this
    paper, we prove that almost all CIC can be seen as a CAC, and that it
    can be further extended with non-strictly positive types and
    inductive-recursive types together with non-free constructors and
    pattern-matching on defined symbols.",
  paper = "Blan05.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Blan99,
  author = "Blanqui, Frederic and jouannaud, Jean-Pierre and Okada, Mitsuhiro",
  title = {{The Calculus of Algebraic Constructions}},
  booktitle = "Rewriting Techniques and Applications RTA-99",
  year = "1999",
  publisher = "LNCS 1631",
  link = "\url{https://hal.inria.fr/inria-00105545v1/document}",
  abstract =
    "This paper is concerned with the foundations of the Calculus of
    Algebraic Constructions (CAC), an extension of the Calculus of
    Constructions by inductive data types. CAC generalizes inductive
    types equipped with higher-order primitive recursion, by providing
    definition s of functions by pattern-matching which capture recursor
    definitions for arbitrary non-dependent and non-polymorphic inductive
    types satisfying a strictly positivity condition. CAC also
    generalizes the first-order framework of abstract data types by
    providing dependent types and higher-order rewrite rules.",
  paper = "Blan99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Boas12,
  author = "Boas, Peter van Emde",
  title = {{Turing Machines for Dummies}},
  journal = "LNCS",
  volume = "7147",
  pages = "14-30",
  year = "2012",
  abstract =
    "Various methods exists in the litearture for denoting the con-
    figuration of a Turing Machine. A key difference is whether the head
    position is indicated by some integer (mathematical representation) or
    is specified by writing the machine state next to the scanned tape
    symbol (intrinsic representation).

    From a mathematical perspective this will make no difference. How-
    ever, since Turing Machines are primarily used for proving undecidability
    and/or hardness results these representations do matter. Based on
    a number of applications we show that the intrinsic representation
    should be preferred.",
  paper = "Boas12.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Bodn01,
  author = "Bodnar, Gabor and Kaltenbacher, Barbara and Pau, Petru and
            Schicho, Josef",
  title = {{Exact Real Computation in Computer Algebra}},
  journal = "LNCS",
  volume = "2630",
  pages = "279-292",
  year = "2001",
  abstract =
    "Exact real computation allows many of the advantages of numerical
    computation (e.g. high performance) to be accessed also in
    symbolic computation, providing validated results. In this paper
    we present our approach to build a transparent and easy to use
    connection between the two worlds, using this paradigm. The main
    discussed topics are representation of exact real objects,
    operations on exact real matrices, polynomial greatest common
    divisor and root computation. Some of these problems are
    ill-posed; we use regularization methods to solve them.",
  paper = "Bodn01.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Bohr16,
  author = "Bohrer, Brandon and Crary, Karl",
  title = {{A Proof-Producing Verified Prolog Compiler}},
  year = "2016",
  link = "\url{www.cs.cmu.edu/~bbohrer/pub/twam-iclp2016-long.pdf}",
  abstract =
    "We have designed and implemented a verified compiler for a
    dialect of Prolog. Our compiler is verified using proof-producing
    compilatoin: every compiled program is accompanied with a formal
    proof that it is equivalent to a particular source program. Our
    formal proofs take the form of type information for our new
    verifying abstract machine which we call the TWAM, whose type
    system natively understands logic programs specified in the
    logical framework LF. We present a soundness metatheorem for the
    TWAM showing that well-typed TWAM programs are sound proof-search
    procedures. In doing so, we reduce our trusted computing base from
    the entire compiler to the TWAM typechecker.",
  paper = "Bohr16.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Bohr17,
  author = "Bohrer, Brandon and Crary, Karl",
  title = {{TWAM: A Certifying Abstract Machine for Logic Programs}},
  journal = "ACM Trans. on Computational Logic",
  volume = "1",
  number = "1",
  year = "2017",
  link = "\url{https://arxiv.org/pdf/1801.00471.pdf}",
  abstract =
    "Type-preserving (or typed) compilation uses typing derivations to
    certify correctness properties of compilation. We have designed and
    implemented a type-preserving compiler for a simply-typed dialect of
    Prolog we call T-Prolog. The crux of our approach is a new certifying
    abstract machine which we call the Typed Warren Abstract Machine
    (TWAM). The TWAM has a dependent type system strong enough to specify
    the semantics of a logic program in the logical framework LF. We
    present a soundness metatheorem which constitutes a partial
    correctness guarantee: well-typed programs implement the logic
    program specified by their type. This metatheorem justifies our design
    and implementation of a certifying compiler from T-Prolog to TWAM.",
  paper = "Bohr17.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Boiz93,
  author = "Boizumault, Patrice and Djamboulian, Ara M. and Fattouh, Jamal",
  title = {{The Implementation of Prolog}},
  year = "1993",
  publisher = "Princeton Legacy Library",
  isbn = "9780691609393",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Bool07,
  author = "Boolos, George S. and Burgess, John P. and Jeffrey, Richard C.",
  title = {{Computability and Logic}},
  publisher = "Cambridge University Press",
  year = "2007",
  isbn = "978-0-521-70146-4",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Bost13,
  author = "Bostan, Alin and Schost, Eric",
  title = {{A Simple and Fast Algorithm for Computing Exponentials
            of Power Series}},
  journal = "Information Processing Letters",
  volume = "13",
  pages = "754-756",
  year = "2013",
  abstract =
    "As was initially shown by Brent, exponentials of truncated power
    series can be computed using a constant number of polynomial
    multiplications. This note gives a relatively simple algorithm
    with a low constant factor",
  paper = "Bost13.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Bost97,
  author = "Bostock, David",
  title = {{Intermediate Logic}},
  publisher = "Clarendon Press",
  year = "1997",
  isbn = "0-19-875141-9",
  paper = "Bost97.pdf",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Bott19,
  author = "Bottu, Gert-Jan and Xie, Ningning and Mardirosian, Klara
            and Schrijvers, Tom",
  title = {{Coherence of Type Class Resolution}},
  year = "2019",
  link = "\url{http://youtube.com/watch?v=bmHdOMoCliM}",
  abstract =
    "Elaboration-based type class resolution, as found in languages
    like Haskell, Mercury and PureScript, is generally
    {\sl nondeterministic}: there can be multiple ways to satisfy a
    wanted constraint in terms of global instances and locally given
    constraints. Coherence is the key property that keeps this sane;
    it guarantees that, despite the nondeterminism, programs will
    behave predictably. even though elaboration-based resolution is
```

```
    generally assumed coherent, as far as we know, there is no formal
    proof of this property in the presence of sources of
    nondeterminism, like suerclasses and flexible contexts.

    This paper provides a formal proof to remedy the situation. The
    proof is non-trivial because the semantics elaborates resolution
    into a target language where different elaborations can be
    distinguished by contexts that do not have a source language
    counterpart. Inspired by the notion of full abstraction, we
    present a two-stop strategy that first elaborates
    nondeterministically into an intermediate language that preserves
    contextual equaivalence, and the deterministically elaborates from
    there into the target language. We use an approach based on
    logical relations to establish contextual equivalence and thus
    coherence for the first step of elaboration, while the second
    step's determinism straightforwardly preserves this coherence
    property.",
  paper = "Bott19.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bouc08,
  author = "Bouche, Thierry",
  title = {{Digital Mathematics Libraries: The Good, the Bad, the Ugly}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
  abstract =
    "The mathematicians' Digital mathematics library (DML), which is
    not to be confused with libraries of mathematical objects
    represented in some digital format, is the generous idea that all
    mathematics ever published should end up in digital form so that
    it would be more easily referenced, accessible, usable. This
    concept was formulated at the very beginning of this century, and
    yielded a lot of international activity that culminated around
    years 2002-2005. While it is estimated that a substantial part of
    the existing math literature is already available in some digital
    format, nothing looking like one digital mathematics library has
    emerged, but a multiplicity of competing electronic offers, with
    unique standards, features, business models, access policies,
    etc. -- even though the contents themselves overlap somewhat,
    while leaving wide areas untouched. The millenium's applealing
    idea has become a new Tower of Babel.

    It is not obvious how much of the traditional library functions we
    should give up while going digital. The point of view shared by
    many mathematicians is that we should be able to find a reasonable
    archiving policy fitting all stakeholders, allowing to translate
    the essential features of the past library system -- which is the
    central infrastructure of all math departments worldwide -- in the
```

```
    digital paradigm, while enhancing overall performances thanks to
    dedicated information technology.

    The vision of this library is rather straightforward: a third
    party to the academic publishing system, preserving, indexing, and
    keeping current its digital collections through a distributed
    network of partners curating the physical holdings, and a
    centralized access facility making use of innovative mining and
    interlinking techniques for easy navigation and discovery.

    However, the fragmentation level is so high that the hope of a
    unique portal providing seamless access to everything relevant to
    mathematical research seems now completely out of reach.
    Nevertheless, we have lessons to learn from each one of the
    already numerous projects running. One of them is that there are
    too many items to deal with, and too many different inital choices
    over metadata sets and formats: it won't be possible to find a
    nontrivial greatest common divisor coping with everything already
    available, and manual upgrading is highly improbable.

    This is where future management techniques for loosely formalised
    mathematical knowledge could provide a new impetus by at last
    enabling a minimum set of features across projects borders through
    automated procedures. We can imagine e.g. math-aware OCR on
    scanned pages, concurrently with interpreters of electronic
    sources of born digital texts, both producing searchable full
    texts in a compatible semistructured format. The challenge is
    ultimately to take advantage of the high formalisation of
    mathematical texts rather than merely ignoring it!

    With these considerations in mind, the talk will focus on
    achievements, limitations, and failures of existing digital
    mathematics libraries, taking the NUMDAM and CEDRAM programs as
    principal examples, hence the speaker himself is the target.",
  paper = "Bouc08.pdf",
  keywords = "DONE"
}
```

———-

— axiom.bib —

```
@article{Bowe95,
  author = "Bowen, Jonathan P. and Hinchey, Michael G.",
  title = {{Seven More Myths of Formal Methods}},
  journal = "IEEE Software",
  volume = "12",
  number = "4",
  pages = "34-41",
  year = "1995",
  abstract =
    "New myths about formal methods are gaining tacit acceptance both
    outside and inside the system-development community. The authors
```

```
    address and dispel these myths based on their observations of
    industrial projects. The myths include: formal methods delay the
    development process; they lack tools; they replace traditional
    engineering design methods; they only apply to software; are
    unnecessary; not supported; and formal methods people always use
    formal methods.",
  paper = "Bowe95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bowm20,
  author = "Bowman, William J.",
  title = {{Cur: Designing a Less Devious Proof Assistant}},
  year = "2020",
  link = "\url{https://vimeo.com/432569820}",
  abstract =
    "Dijkstra said that our tools can have a profound and devious
    influence on our thinking. I find this especially true of modern
    proof assistants, with ‘‘devious’’ out-weighing ‘‘profound’’. Cur
    is an experiment in design that aims to be less devious. The
    design emphasizes language extension, syntax manipulation, and
    DSL construction and integration. This enables the user to be in
    charge of how they think, rather than requiring the user to
    contort their thinking to that of the proof assistant. In this
    talk, my goal is to convince you that you want similar
    capabilities in a proof assistant, and explain and demonstrate
    Cur's attempt at solving the problem.",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@inbook{Boye72,
  author = "Boyer, Robert S. and Moore, J Strother",
  title = {{The Sharing of Structure in Theorem Proving Programs}},
  booktitle = "Machine Intelligence 7",
  publisher = "Edinburgh University",
  pages = "110-116",
  year = "1972",
  abstract =
    "We describe how clauses in resolution programs can be represented
    and used without applying substitutions or cons-ing lists of
    literals. The amount of space required by our representation of a
    clause is independent of the number of literals in the clause and
    the depth of function nesting. We introduce the concept of the
    value of an expression in a binding environment which we use to
```

```
    standardize clauses apart and share the structure of parents in
    representing the resolvent. We present unification and resolution
    algorithms for our representation. Some data comparing our
    representation to more conventional ones is given.",
  paper = "Boye72.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Boye81a,
  author = "Boyer, Robert S. and Moore, J Strother",
  title = {{The Correctness Problem in Computer Science}},
  publisher = "Academic Press",
  year = "1981",
  isbn = "0-12-122920-3",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Brad88,
  author = "Bradford, R.J. and Davenport, J.H.",
  title = {{Effective Tests for Cyclotomic Polynomials}},
  journal = "Lecture Notes in Computer Science",
  volume = "358",
  pages = "244-251",
  year = "1988",
  abstract =
    "We present two efficient tests that determine if a given
    polynomial is cyclotomic, or is a product of cyclotomics. The
    first method uses the fact that all the roots of a cyclotomic
    polynomial are roots of unity, and the second the fact that the
    degree of a cyclotomic polynomial is a value of $\phi(n)$, for
    some $n$. We can also find the cyclotomic factors of any
    polynomial.",
  paper = "Brad88.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Brad02a,
  author = "Bradford, Russell and Davenport, James H.",
  title = {{Towards Better Simplification of Elementary Functions}},
  booktitle = "ISSAC 02",
```

```
  publisher = "ACM",
  year = "2002",
  isbn = "1-58113-484-3",
  abstract =
    "We preent an algorithm for simplifying a large class of
    elementary functions in the presence of branch cuts. This
    algorithm works by:
    \begin{itemize}
    \item verifying that the proposed simplification is correct as a
    simplification of multi-valued functions
    \item decomposing $\mathbb{C}$ (or $\mathbb{C}^n$ in the case of
    multivariate simplifications) according to the branch cuts of the
    relevant functions
    \item checking that the proposed identity is valid on each
    component of that decomposition
    \end{itemize}

    This process can be interfaced to an assume facility, and, if
    required, can verify that simplifications are valid ``almost
    everywhere''.",
  paper = "Brad02a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Brad13a,
  author = "Bradford, Russell and Davenport, James H. and England, Matthew
            and Wilson, David",
  title = {{Optimising Problem Formulation for Cylindrical Algebraic
            Decomposition}},
  journal = "LNCS",
  volume = "7961",
  year = "2013",
  abstract =
    "Cylindrical Algebraic Decomposition (CAD) is an important tool
    for the study of real algebraic geometry with many applications
    both within mathematics and elsewhere. It is known to have doubly
    exponential complexity in the number of variables in the worst
    case, but the actual computation time can vary greatly. It is
    possible to offer different formulations for a given problem
    leading to great differences in tractability. In this paper we
    suggest a new measure for CAD complexity which takes into account
    the real geometry of the problem. This leads to new heuristics for
    choosing: the variable ordering for a CAD problem, a designated
    equational constraint, and formulations for truth-table invariant
    CADs (TTICASs). We then consider the possibility of using Groebner
    bases to precondition TTICAD and when such formulations constitute
    the creation of a new problem.",
  paper = "Brad13a.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Brad05,
  author = "Brady, Edwin C.",
  title = {{Practical Implementation of a Dependently Typed Functional
            Programming Language}},
  school = "University of Durham",
  year = "2005",
  link = "\url{https://eb.host.cs.st-andrews.ac.uk/writings/thesis.pdf}",
  abstract =
    "Types express a programs meaning, and checking types ensures that a
    program has the intended meaning. In a dependently typed programming
    language types are predicated on values, leading to the possibility of
    expressing invariants of a programs behaviour in its type. Dependent
    types allow us to give more detailed meanings to programs, and hence
    be more confident of their correctness.

    This thesis considers the practical implementation of a dependently
    typed programming language, using the Epigram notation defined by
    McBride and McKinna.  Epigram is a high level notation for dependently
    typed functional programming elaborating to a core type theory based
    on Luos UTT, using Dybjers inductive families and elimination rules
    to implement pattern matching. This gives us a rich framework for
    reasoning about programs.  However, a na ve implementation introduces
    several run-time overheads since the type sys- tem blurs the
    distinction between types and values; these overheads include the
    duplication of values, and the storage of redundant information and
    explicit proofs.

    A practical implementation of any programming language should be as
    efficient as pos- sible; in this thesis we see how the apparent
    efficiency problems of dependently typed pro- gramming can be overcome
    and that in many cases the richer type information allows us to apply
    optimisations which are not directly available in traditional
    languages. I introduce three storage optimisations on inductive
    families; forcing, detagging and collapsing. I further introduce a
    compilation scheme from the core type theory to G-machine code,
    including a pattern matching compiler for elimination rules and a
    compilation scheme for efficient run- time implementation of Peanos
    natural numbers. We also see some low level optimisations for removal
    of identity functions, unused arguments and impossible case branches.
    As a result, we see that a dependent type theory is an effective base
    on which to build a feasible programming language.",
  paper = "Brad05.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Brad10,
```

```
  author = "Brady, Edwin and Chapman, James and
            Dagand, Pierre-Evariste and Gundry, Adam and
            McBride, Conor and Morris, Peter and Norell, Ulf",
  title = {{An Epigram Implementation}},
  year = "2010",
  link = "\url{http://www.e-pig.org}",
  paper = "Brad10.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Brad17,
  author = "Brady, Edwin",
  title = {{Dependent Types in the Idris Programming Language 1}},
  year = "2017",
  link = "\url{https://www.cs.uoregon.edu/research/summerschool/summer17/topics.php}"

}
```

———-

— **axiom.bib** —

```
@book{Brad17a,
  author = "Brady, Edwin",
  title = {{Type-Driven Development with IDRIS}},
  publisher = "Manning Publications",
  year = "2017",
  isbn = "978-1-61729-302-3",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Brad18,
  author = "Brady, Edwin",
  title = {{Quantitative Type Theory in Action}},
  year = "2018",
  link = "\url{https://www.type-driven.org.uk/edwinb/papers/idris2.pdf}",
  abstract =
    "Dependent types allow us to express precisely {\sl what} a
    function is indented to do. Recent work on Quantitative Type
    Theory (QTT) extends dependent type systems with {\sl linearity},
    also allowing precision in expressing {\sl when} a function can
    run. This is promising, because it suggests the ability to design
    and reason about resource usage protocols, such as we might find
    in distributed and concurrent programming, where the state of a
```

```
    communication channel changes throughout program execution. As
    yet, however, there has not been a full-scale programming language
    with which to experiment with these ideas. Idris 2 is a new
    version of the dependently typed language Idris, with a new core
    language based on QTT, supporting linear and dependent types. In
    this paper, we introduce Idris 2, and describe how QTT has
    influenced its design. We give several examples of the benefits of
    QTT in proactice including: clearly expressing which data is
    erased at run time, at the type level; improving interactive
    program development by reducing the search space for type-driven
    program synthesis; and, resource tracking in the type system
    leading to type-safe concurrent programming with session types.",
  paper = "Brad18.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Brad20,
  author = "Brady, Sean",
  title = {{Drop Your Tools -- Does Expertise Have A Dark Side}},
  year = "2020",
  link = "\url{https://www.youtube.com/watch?v=Yv4tI6939q0}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Brai13,
  author = "Braibant, Thomas and Chlipala, Adam",
  title = {{Formal Verification of Hardware Synthesis}},
  booktitle = "Computer Aided Verification (CAV'13)",
  publisher = "unknown",
  year = "2013",
  abstract =
    "We report on the implmentation of a certified compier for a
    high-level hardware description language (HDL) called Fe-Si
    (FEatherweight Synthesis). Fe-Si is a simplified version of
    Bluespec, an HDL based on a notion of guarded atomic
    actions. Fe-Si is defined as a dependently typed deep embedding in
    Coq. The target language of the compiler corresponds to a
    synthesisable subset of Verilog or VHDL. A key aspect of our
    approach is that input programs to the compiler can be defined and
    proved correct inside Coq. The, we use extraction and a Verilog
    back-end (written in OCaml) to get a certified version of a
    hardware design.",
  paper = "Brai13.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Brea89,
  author = "Breazu-Tannen, Val and Coquand, Thierry and Gunter, Carl A. and
            Scedrov, Andre",
  title = {{Inheritance and Explicit Coercion}},
  booktitle = "Logic in Computer Science",
  year = "1989",
  isbn = "0-8186-1954-6",
  abstract =
    "A method is presented for providing semantic interpretations for
    languages which feature inheritance in the framework of statically
    checked, rich type disciplines. The approach is illustrated by an
    extension of the language Fun of L. Cardelli and P. Wegner (1985),
    which is interpreted via a translation into an extended polymorphic
    lambda calculus. The approach interprets inheritances in Fun as
    coercion functions already definable in the target of the
    translation. Existing techniques in the theory of semantic domains can
    then be used to interpret the extended polymorphic lambda calculus,
    thus providing many models for the original language. The method
    allows the simultaneous modeling of parametric polymorphism, recursive
    types, and inheritance, which has been regarded as problematic because
    of the seemingly contradictory characteristics of inheritance and type
    recursion on higher types. The main difficulty in providing
    interpretations for explicit type disciplines featuring inheritance is
    identified. Since interpretations follow the type-checking
    derivations, coherence theorems are required, and the authors prove
    them for their semantic method.",
  paper = "Brea89.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Brea91,
  author = "Breazu-Tannen, Val and Coquand, Thierry and Gunter, Carl A. and
            Scedrov, Andre",
  title = {{Inheritance as Implicit Coercion}},
  journal = "Information and Computation",
  volume = "93",
  number = "1",
  year = "1991",
  pages = "172-221",
  abstract =
    "We present a method for providing semantic interpretations for
    languages with a type system featuring inheritance polymorphism. Our
    approach is illustrated on an extension of the language Fun of
    Cardelli and Wegner, which we interpret via a translation into an
    extended polymorphic lambda calculus. Our goal is to interpret
    inheritances in Fun via coercion functions which are definable in the
```

```
    target of the translation. Existing techniques in the theory of
    semantic domains can be then used to interpret the extended
    polymorphic lambda calculus, thus providing many models for the
    original language. This technique makes it possible to model a rich
    type discipline which includes parametric polymorphism and recursive
    types as well as inheritance. A central difficulty in providing
    interpretations for explicit type disciplines featuring inheritance in
    the sense discussed in this paper arises from the fact that programs
    can type-check in more than one way. Since interpretations follow the
    type-checking derivations, coherence theorems are required: that is,
    one must prove that the meaning of a program does not depend on the
    way it was type-checked. Proofs of such theorems for our proposed
    interpretation are the basic technical results of this
    paper. Interestingly, proving coherence in the presence of recursive
    types, variants, and abstract types forced us to reexamine fundamental
    equational properties that arise in proof theory (in the form of
    commutative reductions) and domain theory (in the form of strict
    vs. non-strict functions).",
  paper = "Brea91.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Brid99,
  author = "Bridges, Douglas and Reeves, Steve",
  title = {{Constructive Mathematics in Theory and Programming
           Practice}},
  journal = "Philosophia Mathematica",
  volume = "7",
  number = "3",
  year = "1999",
  pages = "65-104",
  paper = "Brid99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Bron88a,
  author = "Bronstein, Manuel",
  title = {{Fast Reduction of the Risch Differential Equation}},
  journal = "Lecture Notes in Computer Science",
  volume = "358",
  pages = "64-72",
  year = "1988",
  abstract =
    "Since Risch (1969) reduced the problem of integrating
    exponentials to solving a first order linear differntial equation
```

```
    in a differential field, there has been considerable interest in
    solving \[y^\prime + fy = g\] for $y$ in a given differential
    field $K$, where $f,g\in K$. Risch (1969) gave an algorithm for a
    more general equation. His algorithm, however, required factoring
    the denominators of $f$ and $g$, which is an obstacle to efficient
    implementations.

    Later, Rothstein (1976) and Davenport (1986) presented algorithms
    for equation $(R)$ that both rely on a squarefree factorization of
    the denominator for $y$. The algorithm in (Davenport 1986) has
    been implemented in the Scratchpad II (see Jenks et al.) and Maple
    (see Char et al, 1985) computer algebra systems. This algorithm,
    however, requires $f$ to be in a certain form (called {\sl weakly
    normalized}), but no computer algorithm that makes $f$ weakly
    normalized has been published.

    We present here a weaker definition of weak-normality, which
    \begin{enumerate}
    \item can always be obtained in a tower of transcendental
    elementary extensions,
    \item gives us an explicit formula for the denominator of $y$, so
    equation $(R)$ can be reduced to a polynomial equation in one
    reduction step
    \end{enumerate}

    As a consequence, we obtain a new algorithm for solving equation
    $(R)$. OUr algorithm is very similar to the one described in
    Rothstein (1976), except that we use weak normality to prevent
    finite cancellation, rather than having to find integer roots of
    polynomials over the constant field of $K$ in order to detect it.",
  paper = "Bron88a.pdf",
  keywords = "axiomref, printed"
}
```

———-

— axiom.bib —

```
@inproceedings{Bron07a,
  author = "Bronstein, Manuel and Maza, Marc Moreno and
            Watt, Stephen M.",
  title = {{Generic Programming Techniques in Aldor}},
  booktitle = "Proc. of AWFS",
  publisher = "unknown",
  year = "2007",
  pages = "72-77",
  abstract =
    "Certain problems in mathematical computing present unusual
    challenges in structuring software libraries. Although generics,
    or templates as they are sometimes called, have been in wide use
    now for many years, new ways to use them to improve library
    architecture continue to be discovered. We find mathemaical
    software to be particularly useful in exploring these ideas
```

```
    because the domain is extremely rich while being
    well-specified. In this paper we present two techniques for using
    generics in mathematical computation: The first allows efficient
    formulation of generic algorithms for modular computations. The
    second allow mathematical domains to be endowed with additional
    algorithms after they are defined and to have these algorithms
    used in generic libraries.",
  paper = "Bron07a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Broo86a,
  author = "Brooks, Frederick P.",
  title = {{No Silver Bullet -- Essence and Accident in Software
            Engineering}},
  booktitle = "The Mythical Man-Month, Anniversary Edition",
  publisher = "Elsevier Science",
  pages = "1069-1076",
  year = "1986",
  comment = "chapter 16",
  abstract =
    "There is no single development, in either technology or
    management technique, which by itself promises even one
    order-of-magnitude improvement within a decade in productivity, in
    reliability, in simpliity",
  paper = "Broo86a.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Broo87,
  author = "Brooks, Frederick P.",
  title = {{No Silver Bullet: Essence and Accidents of Software Engineering}},
  journal = "IEEE Computer",
  volume = "20",
  number = "4",
  pages = "10-19",
  year = "1987",
  abstract =
    "Fashioning complex conceptual constructs is the essence; accidental
    tasks arise in representing the constructs in language. Past progress
    has so reduced the accidental tasks that future progress now depends
    upon addressing the essence.",
  paper = "Broo87.pdf",
  keywords = "printed"
```

```
}
```

———-

— **axiom.bib** —

```
@inproceedings{Broo82,
  author = "Brooks, Rodney A. and Gabriel, Richard P. and Steele, Guy L.",
  title = {{S-1 Common Lisp Implementation}},
  booktitle = "Proc ACM Symp. on Lisp and functional programming",
  publisher = "ACM",
  year = "1982",
  isbn = "0-89791-082-6",
  abstract =
    "We are developing a Lisp implementation for the Lawrence
    Livermore National Laboratory S-1 Mark IIA computer. The dialect
    of Lisp is an extension of COMMON Lisp [Steele;1982], a descendant
    of Maclisp [Moon;1974] and Lisp Machine Lisp [Weinreb;1981].",
  paper = "Broo82.pdf",
  keywords = "printed"

}
```

———-

— **axiom.bib** —

```
@article{Broo82a,
  author = "Brooks, Rodney A. and Gabriel, Richard P. and Steele, Guy L.",
  title = {{An Optimizing Compiler for Lexically Scoped LISP}},
  journal  = "ACM SIGPLAN Notices",
  volume = "17",
  number = "6",
  year = "1982",
  pages = "261-275",
  abstract =
    "We are developing an optimizing compiler for a dialect of the
    LISP language. The current target architecture is the S-1, a
    multiprocessing supercomputer designed at Lawrence Livermore
    National Laboraty. While LISP is usually thought of as a language
    primarily for symbolic processing and list manipulation, this
    compiler is also intended to compete with the S-1 PASCAL and
    FORTRAN compilers for quality of compiled numerical code. The S-1
    is designed for extremely high-speed signal processing as well as
    for symbolic computation; it provides primitive operations on
    vectors of floating-point and complex numbers. The LISP compiler
    is designed to exploit the architecture heavily.

    The compiler is structurally and conceptually similar to the
    BLISS-11 compiler and the compilers produced by PQCC. In
    particular, the TNBIND technique has been borrowed and extended.
```

```
    Particularly interesting properties of the compiler are:
    \begin{itemize}
    \item Extensive use of source-to-source transformation
    \item Use of an intermediate form that is expression-oriented
    rather than statement oriented
    \item Exploitation of tail-recursive function calls to represent
    coplex control structures
    \item Efficient compilation of code that can manipulate procedural
    object that require heap-allocated environments
    \item Smooth run-time interfacing between the ``numerical world''
    and ``LISP pointer world'', including automatic stack allocation
    of objects that ordinarlly must be heap-allocated
    \end{itemize}

    Each of these techniques has been used before, but we believe
    their synthesis to be original and unique.

    The compiler is table-driven to a great extent, more so than
    BLISS-11 but less so than a PQCC compiler. We expect to be able to
    redirect the compiler to other target architectures such as the
    VAX or PDP-10 with relatively little effort.",
  paper = "Broo82a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Broo84,
  author = "Brooks, Rodney A. and Gabriel, Richard P.",
  title = {{A Critique of Common Lisp}},
  booktitle = "Symposium on Lisp and Functional Programming",
  pages = "1-8",
  year = "1984",
  abstract =
    "A major goal of the COMMON LISP committee was to define a Lisp
    language with sufficient power and generality that people would be
    happy to stay within its confines and thus write inherently
    transportable code. We argue that the resulting language
    definition is too large for many short-term and medium-term
    potential applications. In addition many parts of COMMON LISP
    cannot be implemented very efficiently on stock hardware. We
    further argue that the very generality of the design with its
    different efficieny profiles on different architectures works
    agains the goal of transportability.",
  paper = "Broo84.pdf",
  keywords = "printed"
}
```

———-

,

— **axiom.bib** —

```
@inproceedings{Broo86,
  author = "Brooks, Rodney A. and Posner, David B. and McDonald, James L.
            and White, Jon L. and Benson, Eric and Gabriel, Richard P.",
  title = {{Design of An Optimising, Dynamically Retargetable Compiler
            for Common Lisp}},
  booktitle = "Conf. on Lisp and Functional Programming",
  publisher = "ACM",
  pages = "67-85",
  year = "1986",
  abstract =
    "We outline the components of a retargetable cross-compiler for
    the Common Lisp language. A description is given of a method for
    modeling the various hardware features in the compiler's database,
    and a breakdown is shown of the compiler itself into various
    machine-independent and machine-dependent modules. A novel feature
    of this development is the dynamic nature of the retargeting:
    Databses for multiple hardware architectures are a standard part
    of the compiler, and the internal interfaces used by the compiler
    are such that the machine-dependent modules may be instantly
    switched from one to another. Examples of generated code in
    several environments will be given to demonstrate the high quality
    of the output available, even under this very modular approach..",
  paper = "Broo86.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Brow19,
  author = "Brown, Chad E. and Gauthier, Thibault",
  title = {{Self-Learned Formula Synthesis in Set Theory}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1912.01525.pdf}",
  abstract =
    "A reinforcement learning algorithm accomplishes the task of
    synthesizing a set-theoretical formula that evaluates to a given
    truth value for given assignments.",
  paper = "Brow19.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Brow71,
  author = "Brown, W.S.",
  title = {{On Euclid's Algorithm and the Computation of Polynomial
            Greatest Common Divisors}},
  journal = "J. ACM",
```

```
  volume = "18",
  number = "4",
  pages = "478-504",
  year = "1971",
  abstract =
    "This paper examines the computation of polynomial greatest common
    divisors by various generalizations of Euclid's algorithm. The
    phenomenon of coefficient growth is described, and the history of
    successful efforts first to control it and then to eliminate it is
    related.

    The recently developed modular algorithm is presented in careful
    detail, with special attention to the case of multivariate
    polynomials.

    The computing times for the classical algorithm and for the
    modular algorithm are analyzed, and it is shown that the modular
    algorithm is markedly superior. In fact, in the multivariate case,
    the maximum computing time for the modular algorithm is strictly
    dominated by the maximum computing time for the first
    pseudo-division in the classical algorithm.",
  paper = "Brow71.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Brow71a,
  author = "Brown, W.S. and Traub, J.F.",
  title = {{On Euclid's Algorithm and the Theory of Subresultants}},
  journal = "J. ACM",
  volume = "18",
  number = "4",
  pages = "505-514",
  year = "1971",
  abstract =
    "This papers presents an elementary treatment of the theory of
    subresultants, and examines the relationship of the subresultants
    of a given pair of polynomials to their polynomial remainder
    sequence as determined by Euclid's algorithm. Two important
    versions of Euclid's algorithm are discussed. The results are
    essentially the same as those of Collins, but the presentatino is
    briefer, simpler, and somewhat more general.",
  paper = "Brow71a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bruc92,
  author = "Bruce, Kim and Mitchell, John C.",
  title = {{PER Models of Subtyping, Recursive Types and Higher-Order
            Polymorphism}},
  booktitle = "POPL '92",
  pages = "316-327",
  year = "1992",
  abstract =
    "We relate standard techniques for solving recursive domain equations
    to previous models with types interpreted as partial equivalence
    relations (per's) over a $D_\infty$ lambda model. This motivates a
    particular choice of type functions, which leads to an extension of
    such models to higher-order polymorphism. The resulting models provide
    natural interpretations for function spaces, records, recursively
    defined types, higher-order type functions, and bounded polymorphic
    types $\forall X <: Y.A$ where the bound may be of a higher kind. In
    particular, we may combine recursion and polymorphism in a way that
    allows the bound $Y$ in $\forall X <: Y.A$ to be recursively
    defined. The model may also be used to interpret so-called
    ``F-bounded polymorphism''. Together, these features allow us to
    represent several forms of type and type functions that seem to arise
    naturally in typed object-oriented programming.",
  paper = "Bruc92.pdf"
}
```

———-

— axiom.bib —

```
@inproceedings{Bruc93,
  author = "Bruce, Kim B.",
  title = {{Safe type checking in a statically-typed object-oriented
           programming language}},
  booktitle = "POPL 93",
  year = "1993",
  isbn = "0-89791-560-7",
  pages = "285-298",
  abstract =
    " In this paper we introduce a statically-typed, functional,
    object-oriented programming language, TOOPL, which supports classes,
    objects, methods, instance variable, subtypes, and inheritance. It has
    proved to be surprisingly difficult to design statically-typed
    object-oriented languages which are nearly as expressive as Smalltalk
    and yet have no holes in their typing systems. A particular problem
    with statically type checking object-oriented languages is determining
    whether a method provided in a superclass will continue to type check
    when inherited in a subclass. This program is solved in our language
    by providing type checking rules which guarantee that a method which
    type checks as part of a class will type check correctly in all legal
    subclasses in which it is inherited. This feature enables library
    providers to provide only the interfaces of classes with executables
    and still allow users to safely create subclasses. The design of TOOPL
    has been guided by an analysis of the semantics of the language, which
```

```
    is given in terms of a sufficiently rich model of the F-bounded
    second-order lambda calculus. This semantics supported the language
    design by providing a means of proving that the type-checking rules
    for the language are sound, ensuring that well-typed terms produce
    objects of the appropriate type. In particular, in a well-typed
    program it is impossible to send a message to an object which lacks a
    corresponding method.",
  paper = "Bruc93.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Brea89a,
  author = "Breazu-Tannen, Val Gallier, Jean",
  title = {{Polymorphic Rewriting Concerves Algebraic Strong Normalization
           and Confluence}},
  booktitle = "Automata, Languages and Programming",
  pages = "137-150",
  year = "1989",
  abstract =
    "We study combinations of many-sorted algebraic term rewriting systems
    and polymorphic lambda term rewriting. Algebraic and lambda terms are
    mixed by adding the symbols of the algebraic signature to the
    polymorphic lambda calculus, as higher-order constants.

    We show that if a many-sorted algebraic rewrite system R is strongly
    normalizing (terminating, noetherian), then
    $R+\beta+\nu+type-\beta+type-\nu$ rewriting of mixed terms is also
    strongly normalizing. We obtain this results using a technique which
    generalizes Girard's ``{candidats de reductibiliti\'e}'', introduced in
    the original proof of strong normalization for the polymorphic lambda
    calculus.

    We also show that if a many-sorted algebraic rewrite system $R$ has
    the Church-Rosser property (is confluent), then
    $R+\beta+type-\beta+type-\nu$ rewriting of mixed terms has the
    Church-Rosser property too. Combining the two results, we conclude
    that if $R$ is canonical (complete) on algebraic terms, then
    $R+\beta+type-\beta+type-\nu$  is canonical on mixed terms.

    $\nu$ reduction does not commute with algebraic reduction, in general.
    However, using long $\nu$-normal forms, we show that if $R$ is canonical
    then $R+\beta+type-\beta+type-\nu$ convertibility is still decidable.",
  paper = "Brea89.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Bund83,
  author = "Bundy, Alan",
  title = {{The Computer Modelling of Mathematics Reasoning}},
  year = "1983",
  publisher = "Academic Press",
  isbn = "0-12-141250-4",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Burs69,
  author = "Burstall, R.M.",
  title = {{Proving Properties of Programs by Structural Induction}},
  journal = "Computer Journal",
  volume = "12",
  number = "1",
  pages = "41-48",
  link = "\url{http://www.cse.chalmers.se/edu/year/2010/course/DAT140_Types/Burstall.pdf}",
  year = "1969",
  abstract =
    "This paper discusses the technique of structural induction for
    proving theorems about programs.  This technique is closely related to
    recursion induction but makes use of the inductive definition of the
    data structures handled by the programs.  It treats programs with
    recursion but without assignments or jumps.  Some syntactic extensions
    to Landin's functional programming language ISWIM are suggested which
    make it easier to program the manipulation of data structures and to
    develop proofs about such programs.  Two sample proofs are given to
    demonstrate the technique, one for a tree sorting algorithm and one
    for a simple compiler for expressions.",
  paper = "Burs69.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inbook{Buch82a,
  author = "Buchberger, Bruno and Loos, Rudiger",
  title = {{Algebraic Simplification}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "11-44",
  abstract =
    "Some basic techniques for the simplification of terms are
    surveyed. In two introductory sections the problem of canonical
```

```
    algebraic simplification is formally stated and some elementary
    facts are derived that explain the fundamental role of
    simplification in computer algebra. In the subsequent sections two
    major groups of simplification techniques are presented: special
    techniques for simplifying terms over numerical domains and
    completion algorithms for simplification with respect to sets of
    equations. Within the first group canonical simplification
    algorithms for polynomials, rational expressions, radical
    expressions and transcendental expressions are treated (Sections
    3-7). As examples for completion algorithms the Knuth-Bendix
    algorithm for rewrite rules and an algorithm for completing bases
    of polynomial ideals are described (Sections 8-11).",
  paper = "Buch82.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Buch93,
  author = "Buchberger, Bruno and Collins, George E. and Encarnacion, Mark J.
            and Hong, Hoon and Johnson, Jeremy R. and Krandick, Werner and
            Loos, Rudiger and Mandache, Ana M. and Neubacher, Andreas and
            Vielhaber, Herbert",
  title = {{SACLIB 1.1 User's Guide}},
  year = "1993",
  institution = "Kurt Godel Institute",
  abstract =
    "This paper lists most of the algorithms provided by SACLIB and shows
    how to call them from C. There is also a brief explanation of the
    inner workings of the list processing and garbage collection
    facilities of SACLIB",
  paper = "Buch93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Buen91,
  author = "Buendgen, R. and Hagel, G. and Loos, R. and Seitz, S. and
            Simon, G. and Stuebner, R. and Weber, A.",
  title = {{SAC-2 in ALDES -- Ein Werkzeug fur dis Algorithmenforschung}},
  journal = "MathPAD 1",
  volume = "3",
  year = "1991",
  pages = "33-37"
}
```

———-

— **axiom.bib** —

```
@book{Bund93,
  author = "Bundgen, Reinhard",
  title = {{The ReDuX System Documentation}},
  year = "1993",
  publisher = "WSI"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bund93a,
  author = "Bundgen, Reinhard",
  title = {{Reduce the Redex $->$ ReDuX}},
  booktitle = "Proc. Rewriting Techniques and Applications 93",
  year = "1993",
  pages = "446-450",
  publisher = "Springer-Verlag",
  isbn = "3-540-56868-9"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Butl90,
  author = "Butler, Greg and Cannon, John",
  title = {{The Design of Cayley -- A Language for Modern Algebra}},
  booktitle = "DISCO 1990",
  year = "1990",
  pages = "10-19",
  abstract =
    "Established practice in the domain of modern algebra has shaped the
    design of Cayley. The design has also been responsive to the needs of
    its users. The requirements of the users include consistency with
    common mathematical notation; appropriate data types such as sets,
    sequences, mappings, algebraic structures and elements; efficiency;
    extensibility; power of in-built functions and procedures for known
    algorithms; and access to common examples of algebraic structures. We
    discuss these influences on the design of Cayley's user language.",
  paper = "Butl90.pdf",
  keywords = "axiomref"
}
```

———-

1.2.3 C

— **axiom.bib** —

```
@article{Cair06,
  author = "Cairns, Paul and Gow, Jeremy",
  title = {{Literate Proving: Presenting and Documenting Formal Proofs}},
  journal = "LNCS",
  volume = "4108",
  year = "2006",
  abstract =
    "Literate proving is the analogue for literate programming in the
    mathematical realm. That is, the goal of literate proving is for
    humans to produce clear expositions of formal mathematics that
    could even be enjoyable for people to read whilst remaining
    faithful representations of the actual proofs. This paper
    describes maze, a generic literate proving system. Authors markup
    formal proof files, such as Mizar files, with arbitrary XML and
    use maze to obtain the selected extracts and transform them for
    presentation, e.g. as Latex. To aid its use, maze has built in
    transformations that include pretty printing and proof sketching
    for inclusion in latex documents. These transformations challenge
    the concept of faithfulness in literate proving but it is argued
    that this should be a distinguishing feature of literate proving
    from literate programming.",
  paper = "Cair06.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Calc16,
  author = "Calcagno, Cristiano and Distefano, Dino and
            Dubreil, Jeremy and Gabi, Dominik and
            Hooimeijer, Pieter and Luca, Martino and
            O'Hearn, Peter and Papakonstantinou, Irene and
            Purbrick, Jim and Rodriguez, Dulma",
  title = {{Moving Fast with Software Verification}},
  year = "2016",
  abstract =
    "For organisations like Facebook, high quality software is
    imprtant. However, the pace of change and increasing complexity of
    modern code makes it difficult to produce error-free
    software. Available tools are often lacking in helping programmers
    developer more reliable and secure applications.

    Formal verificatin is a technique able to detect software errors
    statically, before a product is actually shipped. Although this
    aspect makes this technology very appealing in principle, in
    practice there have been many difficulties that have hindered the
    application of software verification in industrial
    environments. In particular, in an organisation like Facebook
    where the release cycle is fast compared to more traditional
    industries, the deployment of formal techniques is highly
    challenging.
```

```
    This paper describes our experience in integrating a verification
    tool based on static analysis into the software development cycle
    at Facebook.",
  paper = "Calc16.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Call08,
  author = "Callaghan, Paul",
  title = {{Coercive Subtyping via Mappings of Reduction Behaviour}},
  journal = "Electronic Notes in Theoretical Computer Science",
  volume = "196",
  pages = "53-68",
  year = "2008",
  abstract =
    "This paper reports preliminary work on a novel approach to
    Coercive Subtyping that is based on relationships between
    reduction behaviour of source and target types in coerced
    terms. Coercive subtyping is a subset of record-based subtyping,
    allowing so-called coercion functions to carry the subtyping. This
    allows many novel and powerful forms of subtyping and
    abbreviation, with applicaions including interfaces to theorem
    provers and programming with dependent type systems. However, the
    use of coercion functions introduces non-trivial overheads, and
    requires difficult proof of properties such as coherence in order
    to guarantee sensible results. These points restrict the
    practicality of coercive subtyping. We begin with the idea that
    coercing a value $v$ from type $U$ to a type %T% intuitively means
    that we wish to compute with $v$ as if it was a value in $T$, not
    that $v$ must be converted into a value in $T$. Instead, we
    explore how to compute on $U$ in terms of computation on $T$, and
    develop a framework for mapping computations on some $T$ on
    computations on some $U$ via a simple extension of the elimination
    rule of $T$. By exposing how computations on different types are
    relatd, we gain insight on and make progress with several aspects
    of coercive subtyping, including (a) distinguishing classes of
    coercion and finding reasons to deprecate use of some classes; (b)
    alternative techniques for improving key properties of coercions;
    (c) greater efficiency from implementations of coercions.",
  paper = "Call08.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Calm93,
```

```
  author = "Calmet, J. and Tjandra, I.A.",
  title = {{A Unified-Algebra-based Specification Language for
            Symbolic Computing}},
  booktitle = "DISCO 1993",
  year = "1993",
  pages = "122-133",
  publisher = "Springer",
  abstract =
    "A precise and perspicuous specification of mathematical domains
    of computation and their inherently related type inference
    mechanisms is a prerequisite for the design and systematic
    development of a system for symbolic computing. This paper
    describes FORMAL, a language for giving modular and
    well-structured specifications of such domains and particularly of
    ‘‘mathematical objects’’. A novel framework for algebraic
    specification involving so-called ‘‘unified algebras’’ has been
    adopted, where sorts are treated as values. The adoption of this
    framework aims also at being capable of specifying polymorphism,
    unifying the notions of ‘‘parametric’’ and ‘‘inclusion’’
    polymorphisms. Furthermore, the operational nature of the
    specification formalisms allows a straightforward transformation
    into an executable form.",
  paper = "Calm93.pdf",
  keywords = "printed, axiomref"
}
```

———-

— axiom.bib —

```
@article{Calm09,
  author = "Calmet, Jacques",
  title = {{Abstraction-Based Information Technology: A Framework for
            Open Mechanized Reasoning}},
  journal = "LNCS",
  volume = "5625",
  year = "2009",
  abstract =
    "OMRS (Open Mechanized Reasoning Systems) was designed for
    Automated Theorem Proving and then extended to Computer
    Algebra. These are the two domains at the heart of the Calculemus
    approach. An obvious question is to assess whether such an
    approach can be extended to new domains either within AI or
    outside of AI. There have been several attempts to turn the world
    into a computational system. This talk stays away from such
    general attempts and introduces a framework that is fully set
    within AI. It extends the basic concepts of OMRS to diverse fields
    ranging from information technology to sociology through law as
    illustrated by examples. The main motivation is to claim that
    whatever the selected approach, Artificial Intelligence is gaining
    enough strength and power to reach new frontiers and to turn
    challenges that are not a priori of a purely computational nature
    into AI domains.",
```

```
  paper = "Calm09.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Calm10,
  author = "Calmet, Jacques and Campbell, John A.",
  title = {{A Revisited Perspective on Symbolic Mathematical Computing
           and Artificial Intelligence}},
  journal = "LNCS",
  volume = "6167",
  year = "2010",
  abstract =
    "We provide a perspective on the current state and possible future
    of links between symbolic mathematical computing and artificial
    intelligence, on the occasion of the 10th biennial conference
    (AISMC, later AISC) devoted to those connections. It follows a
    similar perspective expressed for the first such conference in 1992
    and then revised and expanded 5 years later. Issues related to the
    computational management of mathematical knowledge are
    highlighted.",
  paper = "Calm10.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Calu07,
  author = "Calude, C.S. and Calude, E. and Marcus, S.",
  title = {{Proving and Programming}},
  type = "technical report",
  institution = "Centre for Discrete Mathematics and Theoretical Computer
                 Science",
  number = "CDMTCS-309",
  year = "2007",
  abstract =
    "There is a strong analogy between proving theorems in mathematics and
    writing programs in computer science.  This paper is devoted to an
    analysis, from the perspective of this analogy, of proof in
    mathematics.  We will argue that while the Hilbertian notion of proof
    has few chances to change, future proofs will be of various types,
    will play different roles, and their truth will be checked
    differently.  Programming gives mathematics a new form of
    understanding.  The computer is the driving force behind these
    changes.",
  paper = "Calu07.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Calu09,
  author = "Calude, Cristian S. and Muller, Christine",
  title = {{Formal Proof: Reconciling Correctness and Understanding}},
  journal = "LNCS",
  volume = "5625",
  year = "2009",
  abstract =
    "Hilbert's concept of formal proof is an ideal of rigour for
    mathematics which has important applications in mathematical
    logic, but seems irrelevant for the practice of mathematics. The
    advent, in the last twenty years, of proof assistants was followed
    by an impressive record of deep mathematical theorems formally
    proved. Formal proof is practically achievable. With formal proof,
    correctness reaches a standard that no pen-and-paper proof can
    match, but an essential component of mathematics -- the insight
    and understanding -- seems to be in short supply. So, what makes a
    proof understandable? To answer this question we first suggest a
    list of symptoms of understanding. We then propose a vision of an
    environment in which users can write and check formal proofs as
    well as query them with reference to the symptoms of
    understanding. In this way, the environment reconciles the main
    features of proof: correctness and understanding.",
  paper = "Calu09.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Camp70,
  author = "Campbell, J.A. and Hearn, Anthony C.",
  title = {{Symbolic Analysis of Feynman Diagrams by Computer}},
  journal = "J. of Computational Physics",
  volume = "5",
  number = "2",
  pages = "280-327",
  year = "1970",
  abstract =
    "We describe a system of programs in the language LISP 1.5 which
    handles all stages of calculation from the specification of an
    elementary-particle process in terms of a Hamiltonian of
    interaction or Feynman diagrams to the derivation of an absolute
    square of the matrix element for the process. Examples of
    significant parts of the program are presented in the text, while
    a detailed listing of this material is contained in two Appendices
    which are avaiable on request from the authors.",
  paper = "Camp70.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Camp84,
  author = "Campbell, J.A.",
  title = {{Implementations of PROLOG}},
  year = "1984",
  publisher = "Ellis Horwood Limited",
  isbn = "0-85312-675-5",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Cann88a,
  author = "Canny, John",
  title = {{Generalized Characteristic Polynomials}},
  journal = "Lecture Notes in Computer Science",
  volume = "358",
  pages = "293-299",
  year = "1988",
  abstract =
    "We generalize the notion of characteristic polynomial for a
    system of linear equations to systems of multivariate polynomial
    equations. The generalization is natural in the sense that it
    reduces to the usual definition when all the polynomials are
    linear. Whereas the constant coefficient of the characteristic
    polynomial is the resultant of the system. This construction is
    applied to solve a traditional problem with efficient methods for
    solving systems of polynomial equations: the presence of
    infinitely many solutions ‘‘at infinity’’. We give a
    single-exponential time method for finding all the isolated
    solution points of a system of polynomials, even in the presence
    of infinitely many solutions at infinity or elsewhere.",
  paper = "Cann88a.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Cann01,
  author = "Cannon, John J. and Playoust, Catherine",
  title = {{An Introduction to Algebraic Programming with Magma}},
  year = "2001",
  publisher = "University of Sydney",
  paper = "Cann01.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Cant16,
  author = "Cantrill, Bryan",
  title = {{Oral Tradition in Software Engineering}},
  link = "\url{https://www.youtube.com/watch?v=4PaWFYm0kEw}",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@article{Card85,
  author = "Cardelli, Luca and Wegner, Peter",
  title = {{On Understanding Types, Data Abstraction, and Polymorphism}},
  journal = "ACM Computing Surveys",
  volume = "17",
  number = "4",
  year = "1985",
  pages = "471-523",
  abstract =
    "Our objective is to understand the notion of type in programming
    languages, present a model of typed, polymorphic programming languages
    that reflects recent research in type theory, and examine the
    relevance of recent research to the design of practical programming
    languages.

    Object-oriented languages provide both a framework and a
    motivation for exploring the interaction among the concepts of type,
    data abstraction, and polymorphism, since they extend the notion of
    type to data abstraction and since type inheritance is an important
    form of polymorphism. We develop a $\lambda$-calculus-based model for type
    systems that allows us to explore these interactions in a simple
    setting, unencumbered by complexities of production programming
    languages.

    The evolution of languages from untyped universes to
    monomorphic and then polymorphic type systems is reviewed. Mechanisms
    for polymorphism such as overloading, coercion, subtyping, and
    parameterization are examined. A unifying framework for polymorphic
    type systems is developed in terms of the typed $\lambda$-calculus
    augmented to include binding of types by quantification as well as
    binding of values by abstraction.

    The typed $\lambda$-calculus is
    augmented by universal quantification to model generic functions with
    type parameters, existential quantification and packaging (information
    hiding) to model abstract data types, and bounded quantification to
    model subtypes and type inheritance. In this way we obtain a simple
    and precise characterization of a powerful type system that includes
```

```
    abstract data types, parametric polymorphism, and multiple inheritance
    in a single consistent framework. The mechanisms for type checking for
    the augmented $\lambda$-calculus are discussed.

    The augmented typed
    $\lambda$-calculus is used as a programming language for a variety of
    illustrative examples. We christen this language Fun because fun
    instead of $\lambda$ is the functional abstraction keyword and because it
    is pleasant to deal with.

    Fun is mathematically simple and can serve
    as a basis for the design and implementation of real programming
    languages with type facilities that are more powerful and expressive
    than those of existing programming languages. In particular, it
    provides a basis for the design of strongly typed object-oriented
    languages",
  paper = "Card85.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Card86,
  author = "Cardelli, Luca",
  title = {{Typechecking Dependent Types and Subtypes}},
  link =
   "\url{http://lucacardelli.name/Papers/Dependent%20Typechecking.US.pdf}",
  booktitle = "Foundations of Logic and Functional Programming",
  year = "1996",
  journal = "LNCS",
  volume = "523",
  pages = "45-57",
  paper = "Card86.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Card88,
  author = "Cardelli, Luca",
  title = {{A Semantics of Multiple Inheritance}},
  journal = "Information and Computation",
  volume = "76",
  number = "2-3",
  year = "1988",
  pages = "138-164",
  paper = "Card88.pdf",
  keywords = "printed"
```

```
}
```

———-

— **axiom.bib** —

```
@inproceedings{Card88a,
  author = "Cardelli, Luca",
  title = {{Structural Subtyping and the Notion of Power Type}},
  booktitle = "POPL '88",
  year = "1988",
  publisher = "ACM",
  paper = "Card88a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Card88b,
  author = "Cardelli, Luca",
  title = {{Basic Polymorphic Typechecking}},
  journal = "Science of Computer Programming",
  volume = "8",
  number = "2",
  year = "1988",
  paper = "Card88b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Card91,
  author = "Cardelli, Luca and Longo, Giuseppe",
  title = {{A Semantic Basis for Quest}},
  journal = "J. of Functional Programming",
  volume = "1",
  number = "4",
  pages = "417-458",
  year = "1991",
  abstract =
    "Quest is a programming language based on impredicative type
    quantifiers and subtyping within a three-level structure of kinds,
    types and type operators, and values.

    The semantics of Quest is rather challenging. In particular,
    difficulties arise when we try to model simultaneously features such
    as contravariant function spaces, record types, subtyping, recursive
    types and fixpoints.
```

```
    In this paper we describe in detail the type inference rules for
    Quest, and give them meaning using a partial equivalence relation
    model of types. Subtyping is interpreted as in previous work by Bruce
    and Longo (1989), but the interpretation of some aspects  namely
    subsumption, power kinds, and record subtyping  is novel. The latter
    is based on a new encoding of record types.

    We concentrate on modelling quantifiers and subtyping; recursion is
    the subject of current work.",
  paper = "Card91.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Card93,
  author = "Cardelli, Luca",
  title = {{Typeful Programmnig}},
  type = "research report",
  year = "1993",
  institution = "Digital Equipment Corporation",
  number = "SRC Research Report 45",
  abstract =
    "There exists an identifiable programming style based on the
    widespread use of type information handled through mechanical
    typechecking techniques.

    This typeful programming style is in a sense independent of the
    language it is embedded in; it adapts equally well to functional,
    imperative, object-oriented, and algebraic programming, and it is
    not incompatible with relational and concurrent programming.

    The main purpose of this paper is to show how typeful programming
    is best supported by sophisticated type systems, and how these
    systems can help in clarifying programming issues and in adding
    power and regularity to languages.

    We start with an introduction to the notions of types, subtypes
    and polymorphism. Then we introduce a general framework, derived
    in part from constructive logic, into which most of the known type
    systems can be accommodated and extended. The main part of the
    paper shows how this framework can be adapted systematically to
    cope with actual programming constructs. For concreteness we
    describe a particular programming language with advanced features;
    the emphasis here is on the combination of subtyping and
    polymorphism. We then discuss how typing concepts apply to large
    programs. We also sketch how typing applies to system programming;
    an area which by nature escapes rigid typing. In summary, we
    compare the most common programming styles, suggesting that many
    of them are compatible with, and benefit from, a typeful discipline.",
  paper = "Card93.pdf",
```

```
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Care14,
  author = "Carette, Jacques and Farmer, William M. and Kohlhase, Michael",
  title = {{Realms: A Structure for Consolidating Knowledge about
           Mathematical Theories}},
  journal = "LNCS",
  volume = "8543",
  year = "2014",
  abstract =
    "Since there are different ways of axiomatizing and developing a
    mathematical theory, knowledge about such a theory may reside in
    many places and in many forms within a library of formalized
    mathematics. We introduce the notion of a realm as a structure for
    consolidating knowledge about a mathematical theory. A realm
    contains several axiomatizations of a theory that are separately
    developed. Views interconnect these developments and establish
    that the axiomatizations are equivalent in the sense of being
    mutually interpretable. A realm also contains an external interface
    that is convenient for users of the library who want to apply the
    concepts and facts of the theory without delving into the details
    of how the concepts are facts were developed. We illustrate the
    utility of realms through a series of examples. We also give an
    outline of the mechanisms that are needed to create and maintain
    realms.",
  paper = "Care14.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Care17,
  author = "Carette, Jacques and Farmer, William M.",
  title = {{Formalizing Mathematical Knowledge as a Biform Theory
            Graph: A Case Study}},
  journal = "LNCS",
  volume = "10383",
  year = "2017",
  abstract =
    "A biform theory is a combination of an axiomatic theory and an
    algorithmic theory that supports the integration of reasoning and
    computation. These are ideal for formalizing algorithms that
    manipulate mathematical expressions. A theory graph is a network
    of theories connected by meaning-preserving theory morphisms that
    map the formulae of one theory to the formulas of another
    theory. Theory graphs are in turn well suited for formalizing
```

```
    mathematical knowledge at the most convenient level of abstraction
    using the most convenient vocabulary. We are interested in the
    problem of whether a body of mathematical knowledge can be
    effectively formalized as a theory graph of biform theories. As a
    test case, we look at the graph of theories encoding natural
    number arithmetic. We used two different formalisms to do this,
    which we describe and compare. The first is realized in
    CTT$_{uqe}$, a version of Church's type theory with quotation and
    evaluation, and the second is realized in Agda, a dependently
    typed programming language.",
  paper = "Care17.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Care18,
  author = "Carette, Jacques and Farmer, William M. and Sharoda, Yasmine",
  title = {{Biform Theories: Project Description}},
  journal = "LNCS",
  volume = "11006",
  year = "2018",
  abstract =
    "A biform theory is a combination of an axiomatic theory and an
    algorithmic theory that supports the integration of reasoning and
    computation. These are ideal for specifying and reasoning about
    algorithms that manipulate mathematical expressions. However,
    formalizing biform theories is challenging as it requires the
    means to express statements about the interplay of what these
    algorithms do and what their actions mean mathematically. This
    paper describes a project to develop a methodology for expressing,
    manipulating, managing, and generating mathematical knowledge as a
    network of biform theories. It is a subproject of MathScheme, a
    long-term project at McMaster University to produce a framework
    for integrating formal deduction and symbolic computation.",
  paper = "Care18.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Care19,
  author = "Carette, Jacques and O'Connor, Russell and Sharoda, Yasmine",
  title = {{Building on the Diamonds between Theories: Theory
            Presentation Combinators}},
  year = "2019",
  abstract =
    "To build a large library of mathematics, it seems more efficient
    to take advantage of the inherent structure of mathematical
```

```
    theories. Various theory presentation combinators have been
    proposed, and some have been implemented, in both legacy and
    current systems. Surprisingly, the 'standard library' of most
    systems do not make pervasive use of these combinators.

    We present a set of combinators optimized for reuse, via the tiny
    theory approach. Our combinators draw their power from the
    inherent structure already present in the {\sl category of
    contexts} associated to a dependently typed language. The current
    work builds on ideas originating in CLEAR and Specware and their
    descendents (both direct and intellectual). Driven by some design
    criteria for user-centric library design, our library-building
    experience via the systematic use of combinators has fed back into
    the semantics of these combinators, and later into an updated
    syntax for them.",
  paper = "Care19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Carl03,
  author = "Carlisle, David and Dewar, Mike",
  title = {{NAG Library Documentation}},
  journal = "LNCS",
  volume = "2594",
  year = "2003",
  abstract =
    "This paper describes the management and evolution of a large
    collection of 1200 documents detailing the functionality in NAG
    Library products.

    This provides a case study addressing many of the issues which
    concern the ``MKM'' project, involving conversion of legacy
    formats (SGML and Latex) to XML, and inferring semantic content
    from mainly presentational mathematical expressions.",
  paper = "Carl03.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Carl84,
  author = "Carlsson, Mats",
  title = {{On Implementing Prolog in Functional Programming}},
  journal = "New Generation Computing",
  volume = "2",
  pages = "347-359",
  year = "1984",
```

```
  abstract =
    "This report surveys techniques for implementing the programming
    language Prolog. It focuses on explaining the procedural semantics
    of the language in terms of functional programming constructs. The
    techniques {\sl success continuations} and {\sl proof streams} are
    introduced, and it is shown how Horn clause interpreters can be
    built upon them. Continuations are well known from denotational
    semantics theory, in this paper it is shown that they are viable
    constructs in actual programs.",
  paper = "Carl84.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Carn19,
  author = "Carneiro, Mario",
  title = {{Metamath Zero: The Cartesian Theorem Prover}},
  link = "\url{https://arxiv.org/pdf/1910.10703.pdf}",
  year = "2019",
  abstract =
    "As the usage of theorem prover technology expands, so too does
    the reliance on correctness of the tools. Metamath Zero is a
    verification system that aims for simplicity of logic and
    implementation, without compromising on efficiency of
    verification. It is formally specified in its own language, and
    supports a number of translations to and from other proof
    languages. This paper describes the abstract logic of Metamath
    Zero, essentially a multi-sorted first order logic, as well as the
    binary proof format and the way in which it can ensure essentially
    linear time verification while still being concise and efficient
    at scale. Metamath Zero currently holds the record for fastest
    verification of the {\tt set.mm} Metamath library of proofs in ZFC
    (including 71 of Wiedijk's 100 formalization targets), at less
    than 200 ms. Ultimately, we intend to use it to verify the
    correctness of the implementation of the verifier down to binary
    executable, so it can be used as a root of trust for more compolex
    proof systems.",
  paper = "Carn19.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Carn19a,
  author = "Carneiro, Mario",
  title = {{The Type Theory of Lean}},
  year = "2019",
```

```
  link = "\url{https://github.com/digama0/lean-type-theory/releases/v1.0}",
  comment = "\url{https://www.youtube.com/watch?v=3sKrSNhSxik}",
  abstract =
    "This thesis is a presentation of dependent type theory with
    inductive types, a hierarchy of universes, with an impredicative
    universe of propositions, proof irrelevance, and subsingleton
    elimination, along with axioms for propositional extensionality,
    quotient types, and the axiom of choice. This theory is notable
    for being the axiomatic framework of the Lean theorem prover. The
    axiom system is given here in complete detail, including
    ``optional'' features of the type system such as {\bf let} binders
    and definitions. We provide a reduction of the theory to a
    finitely axiomatized fragment utilizing a fixed set of inductive
    types (the {\bf W}-type plus a few others), to ease the study of
    this framework.

    The metatheory of this theory (which we will Lean) is studied. In
    particular, we prove unique typing of the definitional equality,
    and use this to construct the expected set-theoretic model, from
    which we derive consistency of Lean relative to {\bf ZFC+} \{there
    are $n$ inaccessible cardinals $\vert~n<\omega$\} (a relatively
    weak large cardinal assumption). As Lean supports models of
    {\bf ZFC} with $n$ inaccessible cardinals, this is optimal.

    We also show a number of negative results, where the theory is
    less nice than we would like. In particular, type checking is
    undecidable, and the type checking as implemented by the Lean
    theorem prover is a decideable non-transitive underapproximation
    of the typing judgment. Non-transitivity also leads to lack of
    subject reduction, and the reduction relation does not satisfy the
    Church-Rosser property, so reduction to a normal form does not
    produce a decision procedure for definitional equality. However, a
    modified reduction relation allows us to restore the Church-Rosser
    property at the expense of guaranteed termination, so that unique
    typing is shown to hold.",
  paper = "Carn19a.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@misc{Carn19b,
  author = "Carneiro, Mario",
  title = {{The Type Theory of Lean (slides)}},
  year = "2019",
  link = "\url{https://github.com/digama0/lean-type-theory/releases/v1.0}",
  paper = "Carn19b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Carn19c,
  author = "Carneiro, Mario",
  title = {{Specifying Verified x86 Software from Scratch}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1907.01283.pdf}",
  abstract =
    "We present a simple framework for specifying and proving facts
    about the input/output behavior of ELF binary files on the x86-64
    architecture. A strong emphasis has been placed on simplicity at
    all levels: the specification says only what it needs to about the
    target executable, the specification is performed inside a simple
    logic (equivalent to first-order Peano arithmetic), and the
    verification language and proof checker are custom-designed to
    have only what is necessary to perform efficient general purpose
    verification. This forms a part of the Metamath Zero project, to
    build a minimal verifier that is capable of verifying its own
    binary. In this paper, we will present the specification of the
    dynamic semantics of x86 machine code, together with enough
    information about Linux system calls to perform simple IO.",
  paper = "Carn19c.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Cart09,
  author = "Carter, Nathan",
  title = {{Visual Group Theory}},
  publisher = "Mathematical Association of America",
  year = "2009",
  isbn = "978-0-88385-757-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Cast19,
  author = "Castagna, Giuseppe and Lanvin, Victor and Petrucciani, Tommaso
            and Siek, Jeremy G.",
  title = {{Gradual Typing: A New Perspective}},
  link = "\url{https://www.irif.fr/~gc//papers/popl19.pdf}",
  year = "2019",
  abstract =
    "We define a new, more semantic interpretation of gradual types and use
    it to gradualize two forms of polymorphism: subtyping polymorphism
    and implicit parametric polymorphism. In particular, we use the new
```

```
    interpretation to define three gradual type systems Hindley-Milner,
    with subtyping, and with union and intersection types in terms of two
    preorders, subtyping and materialization. These systems are defined
    both declaratively and algorithmically. The declarative presentation
    consists in adding two subsumption-like rules, one for each preorder,
    to the standard rules of each type system. This yields more
    intelligible and streamlined definitions and shows a direct
    correlation between cast insertion and materialization. For the
    algorithmic presentation, we show how it can be defined by reusing
    existing techniques such as unification and tallying.",
  paper = "Cast19.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Cegl19,
  author = "Ceglowski, Maciej",
  tilte = {{The Website Obesity Crisis}},
  year = "2019",
  link = "\url{https://www.youtube.com/watch?v=iYpl0QVCrRU}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Cerv96,
  author = "Cervesato, Iliano and Pfenning, Frank",
  title = {{A Linear Logical Framework}},
  booktitle = "Logic in Computer Science",
  publisher = "IEEE",
  pages = "264-275",
  link = "\url{http://www.cs.cmu.edu/~fp/papers/lics96.pdf}",
  year = "1996",
  abstract =
    "We present the linear type theory LLF as the formal basis for a
    conservative extension of the LF logical framework. LLF combines
    the expressive power of dependent types with linear logic to
    permit the natural and concise represention of a whole new class
    of deductive systems, names those dealing with state. As an
    example we encode a version of Mini-ML with references including
    its type system, its operational semantics, and a proof of type
    preservation. Another example is the encoding of a sequent
    calculus for classical linear logic and its cut elimination
    theorem. LLF can also be given an operational interpretation as a
    logic programming language under which the representations above
    can be used for type inference, evaluation and cut-elimination.",
  paper = "Cerv96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Cerv00,
  author = "Cervesato, Iliano and Pfenning, Frank",
  title = {{A Linear Logic Framework}},
  journal = "Information and Computation",
  volume = "179",
  number = "1",
  pages = "19-75",
  link = "\url{http://www.cs.cmu.edu/~fp/papers/llf00.pdf}",
  year = "2002",
  abstract =
    "We present the linear type theory $\lambda^{\Pi-o\&\top}$ as the
    formal basis for LLF, a conservative extension of the logical
    framework LF. LLF combines the expressive power of dependent types
    with linear logic to permit the natural and concise representation
    of a whole new class of deductive systems, namely those dealing
    with state. As an example we encode a version of Mini-ML with
    mutable references including its type system and its operational
    semantics, and describe how to take practical advantage of the
    representation of its computations.",
  paper = "Cerv00.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Chan89,
  author = "Chandy, K. Mani and Misra, Jayadev",
  title = {{Parallel Program Design}},
  year = "1989",
  publisher = "Addison-Wesley Publishing",
  isbn = "0-201-05866-9",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Chan90,
  author = "Chang, C.C. and Keisler, H. Jerome",
  title = {{Model Theory}},
  publisher = "North Holland",
  year = "1990",
  comment = "Studics in Logic and the Foundations of Mathematics",
```

```
  volume = "73",
  abstract =
    "Since the second edition of this book (1977), Model Theory has
    changed radically, and is now concerned with fields such as
    classification (or stability) theory, nonstandard analysis,
    model-theoretic algebra, recursive model theory, abstract model
    theory, and model theories for a host of nonfirst order logics. Model
    theoretic methods have also had a major impact on set theory,
    recursion theory, and proof theory.

    This new edition has been updated to take account of these changes,
    while preserving its usefulness as a first textbook in model
    theory. Whole new sections have been added, as well as new exercises
    and references. A number of updates, improvements and corrections have
    been made to the main text"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Chan17,
  author = "Chang, Stephen and Knauth, Alex and Greenman, Ben",
  title = {{Type Systems as Macros}},
  booktitle = "Principles of Programming Languages",
  publisher = "ACM",
  year = "2017",
  abstract =
    "We present TURNSTILE, a metalanguage for creating typed embedded
    languages. To implement the type system, programmers write type
    checking rules resembling traditional judgment syntax. To
    implement the semantics, they incorporate elaborations into these
    rules. TURNSTILE critically depends on the idea of linguistic
    reuse. It exploits a macro system in a novel way to simultaneously
    type check and rewrite a surface program into a target
    language. Reusing a macro system also yields modular
    implementations whose rules may be mixed and matched to create
    other languages. Combined with typical compiler and runtime reuse,
    TURNSTILE produces performant typed embedded languages with little
    effort.",
  paper = "Chan17.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Chan20,
  author = "Chang, Stephen and Ballantyne, Michael and Turner, Milo
            and Bowman, William J.",
  title = {{Dependent Type Systems as Macros}},
```

```
  journal = "Proc. ACM Program Lang.",
  volume = "4",
  year = "2020",
  abstract =
    "We present {\tt TURNSTYLE+}, a high-level, macros-based metaDSL
    for building dependently typed languages. With it, programmers may
    rapidly prototype and iterate on the design of new dependently
    typed features and extensions. Or they may create entirely new
    DSLs whose dependet type ``power'' is tailored to a specific
    domain. Our framework's support of language-oriented programming
    also makes it suitable for experimenting with systems of
    interacting components, e.g. a proof assistant and its companion
    DSLs. This paper explains the implementation details of
    {\tt TURNSTYLE+}, as well as how it may be used to create a wide
    variety of dependently typed languages, from a lightweight one
    with indexed types, to a full spectrum proof assistant, complete
    with a tactic system and extensions for features like sized types
    and SMT interaction.",
  paper = "Chan20.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Char84,
  author = "Char, Bruce W. and Geddes, Keith O. and Gonnet, Gaston H.",
  title = {{GCDHEU: Heuristic polynomial GCD algorithm base on Integer
            GCD computation}},
  journal = "LNCS",
  volume = "174",
  pages = "285-296",
  year = "1984",
  abstract =
    "The design of algorithms for polynomial GCD computation has been
    a continuing area of research since the beginning of the
    development of symbolic computation systems. The earliest efforts
    were mainly directed at PRS (Polynomial Remainder Sequence)
    algorithms which are a direct generalization of Euclid's
    algorithm. The main algorithms of this type are the Reduced PRS
    algorithm and the Subresultant PRS algorithm. Hearn discusses the
    use of trial divisions to further improve the performance of PRS
    algorithms. The first fundamentally different polynomial GCD
    algorithm was the modular algorithm. To amek the modular algorithm
    competative for sparse multivariate polynomals, Zippel developed
    the sparse modular algorithm. Another modular-type algorithm was
    the Hensel-based EZ GCD algorithm which was later improved as the
    EEZ GCD algorithm.

    the present paper discusses a new heuristic algorithm, GCDHEU,
    which is found to be very efficient for problems in a small number
    of variables. The heuristic algorithm can be viewed as
```

```
    amodular-type algorithm in that it uses evaluation and
    interpolation, but only a single evaluation per variable is
    used. The heuristic algorithm can be incorporated into a
    reorganized form of the EEZ GCD algorithm such that the base of
    the EEZ GCD algorithm, rather than a univariate GCD algorithm, is
    GCDHEU which is often successful for problems in up to four variables.",
  paper = "Char84.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Char91,
  author = "Char, Bruce and Geddes, Keith O. and Gonnet, Gaston H. and
            Leong, Benton and Monagan, Michael B. and Watt, Stephen M.",
  title = {{Maple V Language Reference Manual}},
  publisher = "Springer",
  year = "1991",
  isbn = "978-0-387-94124-0"
}
```

———-

— **axiom.bib** —

```
@book{Char91a,
  author = "Char, Bruce and Geddes, Keith O. and Gonnet, Gaston H. and
            Leong, Benton and Monagan, Michael B. and Watt, Stephen M.",
  title = {{Maple V Library Reference Manual}},
  publisher = "Springer",
  year = "1991",
  isbn = "978-1-4757-2133-1",
  abstract =
    "The design and implementation of the Maple system is an on-going
    project of the Symbolic Com putation Group at the University of
    Waterloo in Ontario, Canada. This manual corresponds with version V
    (roman numeral five) of the Maple system. The on-line help subsystem
    can be invoked from within a Maple session to view documentation on
    specific topics. In particular, the command ?updates points the user
    to documentation updates for each new version of Maple. The Maple
    project was first conceived in the autumn of 1980, growing out of
    discussions on the state of symbolic computation at the University of
    Waterloo. The authors wish to acknowledge many fruitful discussions
    with colleagues at the University of Waterloo, particularly Morven
    Gen tleman, Michael Malcolm, and Frank Tompa. It was recognized in
    these discussions that none ofthe locaIly-available systems for
    symbolic computation provided the facilities that should be expected
    for symbolic computation in modern computing environments. We
    concluded that since the basic design decisions for the then-current
    symbolic systems such as ALTRAN, CAMAL, REDUCE, and MACSYMA were based
```

```
    on 1960's computing technology, it would be wise to design a new
    system ``from scratch//. Thus we could take advantage of the software
    engineering technology which had become available in recent years, as
    well as drawing from the lessons of experience. Maple's basic features
    (elementary data structures, Input/output, arithmetic with numbers,
    and elementary simplification) are coded in a systems programming
    language for efficiency."
}
```

———-

— **axiom.bib** —

```
@article{Chen08b,
  author = "Chen, Changbo and Maza, Marc Moreno and Pan, Wei and Xie, Yuzhen",
  title = {{On the Verification of Polynomial System Solvers}},
  journal = "Frontiers of Computer Science in China",
  volume = "2",
  number = "1",
  pages = "55-66",
  year = "2008",
  abstract =
    "We discuss the verification of mathematical software solving
    polynomial systems symbolically by way of triangular
    decomposition. Standard verification techniques are highly
    resource consuming and apply only to polynomial systems which are
    easy to solve. We exhibit a new approach which manipulates
    constructible sets represented by regular systems. We provide
    comparative benchmarks of different verification procedures
    applied to four solvers on a large set of well-known polynomial
    systems. Our experimental results illustrate the high efficiency
    of our new approach. In particular, we are able to verify
    triangular decomposition of polynomial systems which are not easy
    to solve.",
  paper = "Chen08b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Chat19,
  author = "Chatley, Robert and Donaldson, Alastair and
            Mycroft, Alan",
  title = {{The Next 7000 Programming Languages}},
  journal = "LNCS",
  volume = "10000",
  pages = "250-282",
  year = "2019",
  abstract =
   "Landin's seminal paper ``The next 700 programming languages''
```

```
   considered programming languages prior to 1966 and speculated on
   the next 700. Half-a-century on, we cast programming languages in a
   Darwinian 'tree of life' and explore languages, their features
   (genes) and language evolution from a viewpoint of 'survival of the
   fittest'.

   We investigatge this thesis by exploring how various anguages fared
   in the past, and then consider the divergence between the languages
   {\sl empirically used in 2017} and the langauge features one might
   have expected if the languages of the 1960s had evolved optimally
   to fill programming niches.

   This leads us to characterise three divergences, or 'elephants in
   the room', were actual current language use, or feature provision,
   differs from that which evolution might suggest. We conclude by
   speculating on future language evolution.",
  paper = "Chat19.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Chen14,
  author = "Chen, Changbo and Covanov, Svyatoslav and Mansouri, Farnam
            and Maza, Marc Moreno and Xie, Ning and Xie, Yuzhen",
  title = {{Basic Polynomial Algebra Subprograms}},
  journal = "Communications in Computer Algebra",
  volume = "48",
  number = "3/4",
  pages = "197-201",
  year = "2014",
  paper = "Chen14.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Chen15,
  author = "Chen, Changbo and Maza, Marc Moreno",
  title = {{Simplification of Cylindrical Algebraic Formulas}},
  journal = "LNCS",
  volume = "9301",
  pages = "119-134",
  year = "2015",
  comment = "Int. Workshop on Computer Algebra in Scientific Computing",
  paper = "Chen15.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Chen16,
  author = "Chen, Changbo and Maza, Marc Moreno",
  title = {{Quantifier Elimination by Cylindrical Algebraic
            Decomposition based on Regular Chains}},
  journal = "Journal of Symbolic Computation",
  volume = "75",
  pages = "74-93",
  year = "2016",
  abstract =
    "A quantifier elimination algorithm by cylindrical algebraic
    decomposition based on regular chains is presented. The main idea
    is to refine a complex cylindrical tree until the signs of
    polynomials appearing in the tree are sufficient to distinguish
    the true and false cells. We report an implementation of our
    algorithm in the RegularChains library in MAPLE and illustrate its
    effectiveness by examples.",
  paper = "Chen16.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Chen92,
  author = "Chen, Kung and Hudak, Paul and Odersky, Martin",
  title = {{Parametric Type Classes}},
  booktitle = "Proc. ACM Conf. on LISP and Functional Programming",
  year = "1992",
  pages = "170-181",
  abstract =
    "We propose a generalization to Haskell's type classes where a class
    can have type parameters besides the placeholder variable. We show
    that this generalization is essential to represent container classes
    with overloaded data constructor and selector operations. We also show
    that the resulting type system has principal types and present
    unification and type reconstruction algorithms.",
  paper = "Chen92.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Chen04,
  author = "Cheng, Eugenia",
  title = {{How to write proofs: A quick guide}},
  link = "\url{http://cheng.staff.shef.ac.uk/proofguide/proofguide.pdf}",
  year = "2004",
```

```
  paper = "Chen04.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Chen06,
  author = "Cheng, Howard and Labahn, George",
  title = {{On Computing Polynomial GCDs in Alternate Bases}},
  booktitle = "ISSAC '06",
  publisher = "ACM",
  year = "2006",
  pages = "47-54",
  abstract =
    "In this paper, we examine the problem of computing the greatest
    common divisor (GCD) of univariate polynomials represented in
    different bases. When the polynomials are represented in Newton
    basis or a basis of orthogonal polynomials, we show that the
    well-known Sylvester matrix can be generalized. We give
    fraction-free and modular algorithms to directly compute the GCD
    in the alternate basis. These algorithms are suitable for
    computation in domains where growth of coefficients in
    intermediate computations are a central concern. In the cases of
    Newton basis and bases using certain orthogonal polynomials, we
    also show that the standard subresultant algorithm can be applied
    easily. If the degrees of the input polynomials is at most $n$ and
    the degree of the GCD is at least $n/2$, our algorithms outperform
    the corresponding algorithms using the standard power basis.",
  paper = "Chen06.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Chen08,
  author = "Cheng, Howard and Labahn, George and Zhou, Wei",
  title = {{Computing Polynomial LCD and GCD in Lagrange Basis}},
  journal = "Communications in Computer Algebra",
  volume = "42",
  number = "3",
  pages = "129-130",
  year = "2008",
  paper = "Chen08.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Chis07,
  author = "Chiswell, Ian and Hodges, Wilfrid",
  title = {{Mathematical Logic}},
  publisher = "Oxford University Press",
  year = "2007",
  isbn = "978-0-19-857100-1",
  paper = "Chis07.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Chli17,
  author = "Chlipala, Adam",
  title = {{Formal Reasoning About Programs}},
  year = "2017",
  publisher = "MIT",
  link = "\url{http://adam.chlipala.net/frap/frap_book.pdf}",
  abstract =
    "Briefly, this book is about an approach to bringing software
    engineering up to speed with more traditional engineering disciplines,
    providing a mathematical foundation for rigorous analysis of realistic
    computer systems. As civil engineers apply their mathematical canon to
    reach high certainty that bridges will not fall down, the software
    engineer should apply a different canon to argue that programs behave
    properly. As other engineering disciplines have their computer-aided
    design tools, computer science has proof assistants, IDEs for logical
    arguments. We will learn how to apply these tools to certify that
    programs behave as expected.

    More specifically: Introductions to two intertangled subjects: the Coq
    proof assistant, a tool for machine-checked mathematical theorem
    proving; and formal logical reasoning about the correctness of
    programs.",
  paper = "Chli17.pdf",
  keywords = "printed, reviewed"
}
```

———-

— **axiom.bib** —

```
@misc{Chli17a,
  author = "Chlipala, Adam",
  title = {{Coming Soon: Machine-Checked Mathematical Proofs in Everyday
            Software and Hardware Development}},
  link = "\url{https://media.ccc.de/v/34c3-9105-coming_soon_machine-checked_mathematical_proofs_in_everyday_soft
  year = "2017",
  abstract =
```

"Most working engineers view machine-checked mathematical proofs as an academic curiosity, if they have ever heard of the concept at all. In contrast, activities like testing, debugging, and code review are accepted as essential. They are woven into the lives of nearly all developers. In this talk, I will explain how I see machine-checked proofs enabling new everyday activities for developers of computer software and hardware. These activities have the potential to lower development effort dramatically, at the same time as they increase our assurance that systems behave correctly and securely. I will give a cosmological overview of this field, answering the FAQs that seem to stand in the way of practicality; and I will illustrate the principles with examples from projects that you can clone from GitHub today, covering the computing stack from digital hardware design to cryptographic software and applications.

Today's developers of computer software and hardware are tremendously effective, compared to their predecessors. We have found very effective ways of modularizing and validating our work. The talk is about ammunition for these activities from a perhaps-unexpected source.

Modularity involves breaking a complex system into a hierarchy of simpler pieces, which may be written and understood separately. Structured programming (e.g., using loops and conditionals instead of gotos) helps us read and understand parts of a single function in isolation, and data abstraction lets us encapsulate important functionality in objects, with guarantees that other code can only access the private data by calling public methods. That way, we can convince ourselves that the encapsulated code upholds certain essential properties, regardless of which other code it is linked with. Systematic unit testing also helps enforce contracts for units of modularity. Each of these techniques can be rerun automatically, to catch regressions in evolving systems, and catch those regressions in a way that accurately points the finger of responsibility to particular modules.

Validation is an important part of development that encompasses testing, debugging, code review, and anything else that we do to raise our confidence that the system behaves as intended. Experienced engineers know that validation tends to take up the majority of engineering effort. Often that effort involves mentally taxing activities that would not otherwise come up in coding. One example is thinking about test-case coverage, and another is including instrumentation that produces traces to consult during debugging.

It is not hard for working developers to imagine great productivity gains from better ways to break systems into pieces or raise our confidence in those pieces. The claim I will make in this talk is that a key source of such insights has been neglected: machine-checked mathematical proofs. Here the basic functionality is an ASCII language for defining mathematical objects, stating theorems about them, and giving proofs of theorems. Crucially, an algorithm checks that purported proofs really do establish the theorems. By going about these activities in the style of programming, we inherit usual

```
    supporting tools like IDEs, version control, continuous integration,
    and automated build processes. But how could so esoteric a task as
    math proofs call for that kind of tooling, and what does it have to do
    with building real computer systems?

    I will explain a shared vision to that end, developed along with many
    other members of my research community. Let me try to convince you
    that all of the following goals are attainable in the next 10 years.

    \begin{itemize}
    \item We will have complete computer systems implementing moderately
    complex network servers for popular protocols, proved to implement
    those protocols correctly, from the level of digital circuits on
    up. We will remove all deployed code (hardware or software) from the
    trusted computing base, shifting our trust to much smaller
    specifications and proof checkers.
    \item Hobbyists will be able to design new embedded computing
    platforms by mixing and matching open-source hardware and software
    components, also mixing and matching the proofs of these components,
    guaranteeing no bugs at the digital-abstraction level or higher, with
    no need for debugging.
    \item New styles of library design will be enabled by the chance to
    attach a formal behavioral specification to each library. For
    instance, rank-and-file programmers will able to assemble their own
    code for cryptographic protocols, with code that looks like reference
    implementations in Python, but getting performance comparable to what
    experts handcraft in assembly today. Yet that benefit would come with
    no need to trust that library authors have avoided bugs or intentional
    backdoors, perhaps even including automatic proofs of cryptographic
    security properties.
    \end{itemize}

    Main technical topics to cover to explain my optimism:
    \begin{itemize}
    \item The basic functionality of proof assistants and why we should
    trust their conclusions
    \item How to think about system decomposition with specifications and
    proofs, including why, for most components, we do not need to worry
    about specification mistakes
    \item The different modes of applying proof technology to check or
    generate components
    \item The engineering techniques behind cost-effective proof authoring
    for realistic systems
    \item A hardware case study: Kami, supporting component-based digital
    hardware authoring with proofs
    \item A software case study: Fiat Cryptography, supporting
    correct-by-construction auto-generation of fast code for
    elliptic-curve cryptography
    \item Pointers to where to look next, if you would like to learn more
    about this technology
    \end{itemize}"
}
```

———-

— **axiom.bib** —

```
@misc{Chli17b,
  author = "Chlipala, Adam and Arvind and Sherman, Benjamin and
            Choi, Joonwon and Vijayaraghavan, Murali",
  title = {{Kami: A Platform for High-Level Parametric Hardware
            Specification and its Modular Verification}},
  year = "2017",
  abstract =
    "It has become fairly standard in the programming languages
    research world to verify functional programs in proof assistants
    using induction, algebraic simplification, and rewriting. In this
    paper, we introduce Kami, a Coq library that uses labelled
    transition systems to enable similar expressive and modular
    reasoning for hardware designs expressed in the style of the
    Bluespec language. We can specify, implement, and verify realistic
    designs entirely within Coq, ending with automatic extraction into
    a pipeline that bottoms out in FPGAs. Our methodology has been
    evaluated in a case study verifying an infinite family fo
    multicore systems, with cache-coherent shared memory and pipelined
    cores implementing (the base integer subset of) the RISC-V
    instruction set.",
  paper = "Chli19b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Choj17,
  author = "Chojecki, Przemyslaw",
  title = {{DeepAlgebra -- An Outline of a Program}},
  journal = "LNCS",
  volume = "10383",
  year = "2017",
  abstract =
    "We outline a program in the area of formalization of mathematics
    to automate theorem proving in algebra and algebraic geometry. We
    propose a construction of a dictionary between automated theorem
    provers and (La)Tex exploiting syntactic parsers. We describe its
    application to a repository of human-written facts and definitions
    in algebraic geometry (The Stacks Project). We use deep learning
    techniques.",
  paper = "Choj17.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Chow18,
  author = "Chow, Timothy Y.",
  title = {{The Consistency of Arithmetic}},
  year = "2018",
  link = "\url{http://timothychow.net/consistent.pdf}",
  paper = "Chow18.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Chri12,
  author = "Christiansen, David Thrane",
  title = {{Converting Regular Expressions to Discrete Finite Automata}},
  year = "2012",
  link = "\url{http://davidchristiansen.dk/tutorials/regex-to-nfa.pdf}",
  paper = "Chri12.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Chri13,
  author = "Christiansen, David Thrane",
  title = {{Didirectional Typing Rules: A Tutorial}},
  year = "2013",
  link = "\url{http://davidchristiansen.dk/tutorials/bidirectional.pdf}",
  paper = "Chri13.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Chri14,
  author = "Christiansen, David Thrane",
  title = {{A Tutorial on Polymorphic Type Derivations}},
  year = "2014",
  link = "\url{http://davidchristiansen.dk/tutorials/type-rule-tutorial.pdf}",
  paper = "Chri14.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Chri18,
  author = "Christiansen, David Thrane",
  title = {{A Little Taste of Dependent Types}},
  year = "2018",
  link = "\url{https://www.youtube.com/watch?v=VxINoKFm-S4}",
  abstract =
    "Dependent types let us use the same programming language for
    compile-time and run-time code, and are inching their way towards the
    mainstream from research languages like Coq, Agda and Idris. Dependent
    types are useful for programming, but they also unite programming and
    mathematical proofs, allowing us to use the tools and techniques we
    know from programming to do math.

    The essential beauty of dependent types can sometimes be hard to find
    under layers of powerful automatic tools. The Little Typer is an
    upcoming book on dependent types in the tradition of the The Little
    Schemer that features a tiny dependently typed language called Pie. We
    will demonstrate a proof in Pie that is also a program."
}
```

———-

— **axiom.bib** —

```
@misc{Chri18a,
  author = "Christiansen, David Thrane",
  title = {{Coding for Types: The Universe Pattern in Idris}},
  year = "2018",
  link = "\url{https://www.youtube.com/watch?v=AWeT_G04a0A}"
}
```

———-

— **axiom.bib** —

```
@misc{Chri19,
  author = "Christiansen, David Thrane",
  title = {{Bidirectional Type Checking}},
  year = "2019",
  link = "\url{http://www.youtube.com/watch?v=utyBNDj7s2w}"
}
```

———-

— **axiom.bib** —

```
@book{Chud89,
  author = "Chudnovsky, David V. and Jenks, Richard D.",
  title = {{Computer Algebra. Pure and Applied Mathematics}},
  publisher = "Springer",
  year = "1989"
```

```
}
```

———-

— **axiom.bib** —

```
@book{Chur41,
  author = "Church, Alonzo",
  title = {{The Calculi of Lambda Conversion}},
  year = "1941",
  publisher = "Princeton University Press",
  paper = "Alon41*.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Ciol11,
  author = "Ciolli, Gianni and Gentili, Graziano and Maggesi, Marco",
  title = {{A Certified Proof of the Cartan Fixed Point Theorem}},
  journal = "J. Autom. Reasoning",
  volume = "47",
  number = "3",
  pages = "319-336",
  year = "2011"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Clae00,
  author = "Claessen, Koen and Hughes, John",
  booktitle = "Proc. 5th ACM SIGPLAN Conf. on Functional Programming",
  publisher = "ACM",
  pages = "268-279",
  year = "2000",
  link = "\url{cs.tufts.edu/~nr/cs257/archive/john-hughes/quick.pdf}",
  abstract =
    "QuickCheck is a tool which aids the Haskell programmer in
    formulating and testing properties of programs. Properties are
    described as Haskell functions, and can be automatically tested on
    random input, but it is also possible to define custom test data
    generators. We present a number of case studies, in which the tool
    was successfully used, and also point out some pitfalls to
    avoid. Random testing is especially suitable for functional
    programs because properties can be stated at a fine grain. When a
    function is built from separately tested components, then random
    testing suffices to obtain good coveraoge of the definition under
    test.",
```

```
  paper = "Clae00.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Clar82,
  author = "Clark, K.L. and Tarnlund, S.-A.",
  title = {{Logic Programming}},
  publisher = "Academic Press",
  year = "1982",
  isbn = "0-12-175520-7",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Clar84,
  author = "Clark, K.L. and McCabe, F.G.",
  title = {{micro-PROLOG: Programming in Logic}},
  year = "1984",
  publisher = "Prentice-Hall",
  isbn = "0-13-581264-X",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Clar11,
  author = "Clark, Kevin",
  title = {{An Algorithm that Decides PRIMES in Polynomial Time}}
, year = "2011",
  link =
   "\url{https://sites.math.washington.edu/~morrow/336_11/papers/kevin.pdf}",
  paper = "Clar11.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Cleb19,
  author = "Clebsch, Sylvan",
  title = {{Starting with Semantics}},
```

```
  year = "2019",
  link = "\url{https://www.youtube.com/watch?v=JbS8a-Ba0Ck}"
}
```

———-

— **axiom.bib** —

```
@article{Cloc18,
  author = "Clochard, Martin and Gondelman, Leon and Pereira, Mario",
  title = {{The Matrix Reproved}},
  journal = "Journal of Automated Reasoning",
  volume = "60",
  number = "3",
  pages = "365-383",
  year = "2018",
  link = "\url{https://hal.inria.fr/hal-01617437/document}",
  abstract =
    "In this paper we describe a complete solution for the
    first challenge of the VerifyThis 2016 competition held at the 18th
    ETAPS Forum. We present the proof of two variants for the
    multiplication of matrices: a naive version using three nested loops
    and Strassen's algorithm. The proofs are conducted using the Why3
    platform for deductive program verification and automated theorem
    provers to discharge proof obligations. In order to specify and
    prove the two multiplication algorithms, we develop a new Why3
    theory of matrices. In order to prove the matrix identities on which
    Strassen's algorithm is based, we apply the proof by reflection
    methodology, which we implement using ghost state. To our knowledge,
    this is the first time such a methodology is used under an
    auto-active setting.",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Cloc91,
  author = "Clocksin, William F.",
  title = {{Clause and Effect}},
  year = "1991",
  publisher = "Springer",
  isbn = "3-540-62971-8",
  keywords = "shelf"
}
```

———-

,

— **axiom.bib** —

```
@misc{Cloo12,
  author = "Cloostermans, Bouke",
  title = {{Quasi-Linear GCD Computation and Factoring RSA Moduli}},
  school = "Eindhoven Univ. of Technology",
  comment = "Bachelor Project",
  year = "2012",
  abstract =
    "This bachelor project consists of two parts. First, we describe
    two subquadratic GCD algorithms and test our implementation of
    these algorithms. Both algorithms work in a similar recursive way
    and have a running time of $O(n log^2n log log n)$ where $n$ is
    the amount of digits of the input. Second, we describe and compare
    three algorithms which compute GCD's of more than two numbers. Two
    algorithms use a treelike structure to compute these GCD's and the
    other algorithm builds a coprime base from the input set. The
    algorithms which uses a treelike structure to compute these GCD's
    is faster than building a coprime base although all algorithms are
    quasilinear. The downside of these algorithm is that they only run
    well if few common factors exist in the input set. These
    algorithms have been used to factorize roughly 0.4\% of a 11.1
    million RSA keys dataset. However, this is only possible if the
    keys are badly generated.",
  paper = "Cloo12.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@article{Coen04,
  author = "Coen, Claudio Sacerdoti and Zacchiroli, Stefano",
  title = {{Efficient Ambiguous Parsing of Mathematical Formulae}},
  journal = "LNCS",
  volume = "3119",
  pages = "347-362",
  year = "2004",
  isbn = "3-540-23029-7",
  abstract =
    "Mathematical notation has the characteristic of being ambiguous:
    operators can be overloaded and information that can be deduced is
    often omitted. Mathematicians are used to this ambiguity and can
    easily disambiguate a formula making use of the context and of their
    ability to find the right interpretation.

    Software applications that have to deal with formulae usually avoid
    these issues by fixing an unambiguous input notation. This solution is
    annoying for mathematicians because of the resulting tricky syntaxes
    and becomes a show stopper to the simultaneous adoption of tools
    characterized by different input languages.

    In this paper we present an efficient algorithm suitable for ambiguous
    parsing of mathematical formulae. The only requirement of the
```

```
    algorithm is the existence of a 'validity' predicate over abstract
    syntax trees of incomplete formulae with placeholders. This
    requirement can be easily fulfilled in the applicative area of
    interactive proof assistants, and in several other areas of
    Mathematical Knowledge Management.",
  paper = "Coen04.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Coen06,
  author = "Coen, Claudio Sacerdoti and Tassi, Enrico and
            Zacchiroli, Stefano",
  title = {{Tinycals: Step by Step Teacticals}},
  booktitle = "Proc. User Interfaces for Theorem Provers",
  year = "2006",
  pages = "125-142",
  abstract =
    "Most of the state-of-the-art proof assistants are based on procedural
    proof languages, scripts, and rely on LCF tacticals as the primary
    tool for tactics composition. In this paper we discuss how these
    ingredients do not interact well with user interfaces based on the
    same interaction paradigm of Proof General (the de facto standard in
    this field), identifying in the coarse-grainedness of tactical
    evaluation the key problem. We propose tinycals as an alternative to a
    subset of LCF tacticals, showing that the user does not experience the
    same problem if tacticals are evaluated in a more fine-grained
    manner. We present the formal operational semantics of tinycals as
    well as their implementation in the Matita proof assistant.",
  paper = "Coen06.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Coen07a,
  author = "Coen, Claudio Sacerdoti and Zacchiroli, Stefano",
  title = {{Spurious Disambiguation Error Detection}},
  booktitle = "MKM 2007 Mathematical Knowledge Management",
  year = "2007",
  abstract =
    "The disambiguation approach to the input of formulae enables the user
    to type correct formulae in a terse syntax close to the usual
    ambiguous mathematical notation. When it comes to incorrect formulae
    we want to present only errors related to the interpretation meant by
    the user, hiding errors related to other interpretations (spurious
    errors). We propose a heuristic to recognize spurious errors, which
    has been integrated with our former efficient disambiguation
    algorithm.",
```

```
  paper = "Coen07a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Coen10,
  author = "Coen, Claudio Sacerdoti",
  title = {{Declarative Representation of Proof Terms}},
  journal = "J. Automated Reasoning",
  volume = "44",
  number = "1-2",
  pages = "25-52",
  year = "2010",
  abstract =
    "We present a declarative language inspired by the pseudo-natural
    language previously used in Matita for the explanation of proof
    terms. We show how to compile the language to proof terms and how to
    automatically generate declarative scripts from proof terms. Then we
    investigate the relationship between the two translations, identifying
    the amount of proof structure preserved by compilation and
    re-generation of declarative scripts.",
  paper = "Coen10.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Coen11,
  author = "Coer, Claudio Sacerdoti and Tassi, Enrico",
  title = {{Nonuniform Coercions via Unification Hints}},
  booktitle = "Proc. Types for Proofs and Programs",
  volume = "53",
  pages = "19-26",
  year = "2011",
  abstract =
    "We introduce the notion of nonuniform coercion, which is the
    promotion of a value of one type to an enriched value of a different
    type via a nonuniform procedure. Nonuniform coercions are a
    generalization of the (uniform) coercions known in the literature and
    they arise naturally when formalizing mathematics in an higher order
    interactive theorem prover using convenient devices like canonical
    structures, type classes or unification hints. We also show how
    nonuniform coercions can be naturally implemented at the user level in
    an interactive theorem prover that allows unification hints.",
  paper = "Coen11.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Cohe10,
  author = "Cohen, Cyril and Mahboubi, Assia",
  title = {{A Formal Quantifier Elimination for Algebraically Closed Fields}},
  journal = "LNCS",
  volume = "6167",
  year = "2010",
  abstract =
    "We prove formally that the first order theory of algebraically
    closed fields enjoys quantifier elimination, and hence is
    decidable. This proof is organized in two modular parts. We first
    reify the first order theory of rings and prove that quantifier
    elimination leads to decidability. Then we implement an algorithm
    which constructs a quantifier free formula from any first order
    formula in the theory of ring. If the underlying ring is in fact
    an algebraically closed field, we prove that the two formulas have
    the same semantic. The algorithm producing the quantifier free
    formula is programmed in continuation passing style, which leads
    to both a concise program and an elegant proof of semantics
    correctness.",
  paper = "Cohe10.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Cohl18,
  author = "Cohl, Howard S. and Greiner-Petter, Andre and Schubotz, Moritz",
  title = {{Automated Symbolic and Numerical Testing of DLMF Formulae
            Using Computer Algebra Systems}},
  journal = "LNCS",
  volume = "11006",
  year = "2018",
  abstract =
    "We have developed an automated procedure for symbolic and
    numerical testing of formulae extracted from the National
    Institute of Standards and Technology (NIST) Digital Library of
    Mathematical Functions (DLMF). For the NIST Digital Repository of
    Mathematical Formulae, we have developed conversion tools from
    semantic Latex to Computer Algebra System (CAS) MAPLE which relies
    on Youssef's part-of-math tagger. We convert a test data subset of
    4,078 semantics Latex DLMF formulae extracted from the DLMF to the
    native CAS representation and then apply an automated scheme for
    symbolic and numerical testing and verification. Our framework is
    implemented using Java and MAPLE. We describe in detail the
    conversion process which is required so that the CAS is able to
    correctly interpret the mathematical representation of the
    formulae. We describe the improvement of the effectiveness of our
    automated scheme through incremental enhancements (making more
    precise) of the mathematical semantics markup of the formulae.",
  paper = "Cohl18.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@techreport{Coll90,
  author = "Collins, George E. and Loos, Rudiger",
  title = {{Specification and Index of SAC-2 Algorithms}},
  institution = "Univ. of Tubingen",
  type = "technical report",
  year = "1990",
  number = "WSI-90-4"
}
```

———-

— **axiom.bib** —

```
@techreport{Coll88,
  author = "Collofello, James S.",
  title = {{Introduction to Software Verification and Validation}},
  type = "technical report",
  institution = "Carnegie Mellon University",
  number = "SEI-CM-13-1.1",
  year = "1988",
  abstract =
    "SEI curriculum modules document and explication software
    engineering topics. They are intended to be useful in a variety of
    situations -- in the preparation of courses, in the planning of
    individual lectures, in the design of curricula, and in
    professional self-study. Topics do not exist in isolation,
    however, and the question inevitably arises how one module is
    related to another. Because modules are written by different
    authors at different times, the answer could easily be 'not at
    all'. In a young field struggling to define itself, this would be
    an unfortunate situation.

    The SEI deliberately employs a number of mechanisms to achieve
    compatible points of view across curriculum modules and to fill
    the content gaps between them. Modules such as {\sl Introduction
    to Software Verification and Validation} is one of the most
    important devices. In this latest revision, Professor Collofillo
    has more modules to integrate into a coherent picture than when we
    released his first draft more than a year ago -- modules on
    quality assurance, unit testing, technical reviews, and formal
    verification, as well as less directly related modules on
    specification, requirements definition, and design. We believe you
    will find this curriculum module interesting and useful, both in
    its own right and by virtue of the understanding of other modules
    that it facilitates.",
  paper = "Coll88.pdf",
```

```
  keywords = "printed"

}
```

———-

— **axiom.bib** —

```
@misc{Comm16,
  author = "Unknown",
  title = {{A very comprehensive and precise spec}},
  year = "2016",
  link =
  "\url{http://www.commitstrip.com/en/2016/08/25/a-very-comprehensive-and-precise-spec}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Como90,
  author = "Comon, Hubert",
  title = {{Equational Formulas in Order-sorted Algebras}},
  booktitle = "IICALP 90. Automata, Languages and Programming",
  year = "1990",
  pages = "674-688",
  abstract =
    "We propose a set of transformation rules for first order formulas
    whose atoms are either equations between terms or sort constraints t
     s where s is a regular tree language (or a sort in the algebraic
    specification community). This set of rules is proved to be correct,
    terminating and complete. This shows in particular that the first
    order theory of any rational tree language is decidable, extending the
    results of [Mal71,CL89,Mah88]. We also show how to apply our results
    to automatic inductive proofs in equational theories."
}
```

———-

— **axiom.bib** —

```
@article{Como91,
  author = "Comon, Hubert and Lugiez, D. and Schnoebelen, Ph.",
  title = {{A Rewrite-based Type Discipline for a Subset of Computer Algebra}},
  journal = "J. Symbolic Computation",
  volume = "11",
  number = "4",
  year = "1991",
  pages = "349-368",
  abstract =
    "This paper is concerned with the type structure of a system including
```

```
    polymorphism, type properties and subtypes. This type system
    originates from computer algebra but it is not intended to be the
    solution of all type problems in this area.

    Types (or sets of types) are denoted by terms in some order-sorted
    algebra. We consider a rewrite relation in this algebra, which is
    intended to express subtyping. The relations between the semantics and
    the axiomatization are investigated. It is shown that the problem of
    type inference is undecidable but a narrowing strategy for
    semi-decision procedures is described and studied.",
  paper = "Como91.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Cong19,
  author = "Cong, Youyou and Osvald, Leo and Essertel, Gregory M.
            and Rompf, Tiark",
  title = {{Compiling with Continuations, or without? Whatever}},
  booktitle = "Inter. Conf. on Functional Programming",
  publisher = "ACM",
  year = "2019",
  abstract =
    "What makes a good compiler IR? In the context of functional
    languages, there has been an extensive debate on the advantages
    and disadvantages of continuation-passing style (CPS). The
    consensus seems to be that some form of explicit continuations is
    necessary to model jumps in a functional style, but that they
    should have a 2nd-class status, separate from regular functions,
    to ensure efficient code generation. Building on this observation,
    a recent study from PLDI 2017 proposed a direct-style IR with
    explicit join points, which essentially represent local
    continuations, i.e. functions that do not return or escape. While
    this IR can work well in practice, as evidenced by the
    implementation of join points in the Glasgow Haskell Compiler
    (GHC), there still seems to be room for improvement, especially
    with regard to the way continuations are handled in the course of
    optimization.

    In this paper, we contribute to the CPS debate by developing a
    novel IR with the following features. First, we integrate a
    control operator that resembles Felleisen's C, eliminating certain
    redundant rewrites observed in the previous study. Second, we
    treat the non-returning and non-escaping aspects of continuations
    separately, allowing efficient compilation of well-behaved
    functions defined by the user. Third, we define a selective CPS
    translation of our IR, which erases control operators while
    preserving the meaning and typing of programs. These features
    enable optimizations in both direct style and full CPS, as well as
    in any intermediate style with selectively exposed continuations.
```

```
    Thus, we change the spectrum of available options from 'CPS yes or
    no' to 'as much or as little CPS as you want, when you want it'.",
  paper = "Cong19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Conr05,
  author = "Conrad, Brian",
  title = {{Impossibility Theorems for Elementary Integration}},
  year = "2005",
  link =
  "\url{http://www2.maths.ox.ac.uk/cmi/library/academy/LectureNotes05/Conrad.pdf}",
  abstract =
    "Liouville proved that certain integrals, most famously
     $\int{e^{-x^2}}~dx$, cannot be expressed in elementary terms. We
     explain how to give precise meaning to the notion of integration
     ``in elementary terms'', and we formulate Liouville's theorem
     that characterizes the possible form of elementary
     antiderivatives. Using this theorem, we deduce a practical
     criterion for proving such impossibility results in special cases.

     This criterion is illustrated for the Gaussian integral
     $\int{e^{-x^2}}~dx$ from probability theory, the logarithmic
     integral $\int{}~dt/log(t)$ from the study of primes, and
     elliptic integrals. Our exposition is aimed at students who are
     familiar with calculus and elementary abstract algebra (at the
     level of polynomial rings $F(t)$ over a field $F$).",
  paper = "Conr05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Cons85a,
  author = "Constable, R.L. and Knoblock, T.B. and Gates, J.L.",
  title = {{Writing Programs that Construct Proofs}},
  journal = "J. of Automated Reasoning",
  volume = "1",
  number = "3",
  pages = "285-326",
  year = "1985",
  abstract =
    "When we learn mathematics, we learn more than definitions and
    theorems. We learn techniques of proof. In this paper, we describe
    a particular way to express these techniques and incorporate them
    into formal theories and into computer systems used to build such
```

```
    theories. We illustrate the methods as they were applied in the
    $\lambda$-PRL system, essentially using the ML programming
    language from Edinburgh LCF [23] as the formalised
    metalanguage. We report our experience with such an approach
    emphasizing the ideas that go beyond the LCF work, such as
    transformation tactics and special purpose reasoners. We also show
    how the validity of tactics can be guaranteed. The introduction
    places the work in historical context and the conclusion briefly
    describes plans to carry the methods further. The majority of the
    paper presents the $\lambda$-PRL approach in detail.",
  paper = "Cons85a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Cons12,
  author = "Constable, Robert",
  title = {{Proofs as Processes}},
  year = "2012",
  link = "\url{https://www.cs.uoregon.edu/research/summerschool/summer12/curriculum.html}",
  keywords = "DONE"

}
```

———-

— **axiom.bib** —

```
@article{Cont81,
  author = "Unknown",
  title = {{Final Report on the National Commission of New
            Technological Uses of Copyrighted Works}},
  journal = "Computer/Law Journal",
  volume = "3",
  number = "1",
  year = "1981",
  comment = "CONTU Report",
  paper = "Cont81.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Cool92,
  author = "Coolsaet, Kris",
  title = {{A Quick Introduction to the Programming Language MIKE}},
  journal = "Sigplan Notices",
  volume = "27",
```

```
  number = "6",
  year = "1992",
  pages = "37-48",
  abstract =
    "MIKE is a new programming language developed by the author as a base
    language for the development of algebraic and symbolic algorithms. It
    is a structured programming language with a MODULA-2-like syntax
    supporting special features such as transparent dynamic memory
    management, discriminated union types, operator overloading, data
    abstraction and parametrized types. This text gives an overview of the
    main features of the language as of version 2.0."
}
```

———-

— **axiom.bib** —

```
@book{Cope04,
  author = "Copeland, B. Jack",
  title = {{The Essential Turing}},
  publisher = "Oxford University Press",
  year = "2004",
  isbn = "978-0-19-825080-7",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Coqu85,
  author = "Coquand, Thierry and Huet, Gerard",
  title = {{Constructions: A Higher Order Proof System for Mechanizing
            Mathematics}},
  journal = "LNCS",
  volume = "203",
  pages = "151-184",
  year = "1985",
  abstract =
    "We present an extensive set of mathematical propositions and proofs
    in order to demonstrate the power of expression of the theory of
    constructions.",
  paper = "Coqu85.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Coqu90,
```

```
  author = "Coquand, Thierry and Paulin, Christine",
  title = {{Inductively Defined Types}},
  booktitle = "Int. Conf. on Computer Logic",
  publisher = "Springer",
  pages = "50-66",
  year = "1990",
  paper = "Coqu90.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Coqu19,
  author = "Coquand, Thierry and Kinoshita, Yoshiki and Nordstrom, Bengt
            and Takeyama, Makoto",
  title = {{A Simple Type-Theoretic Language: Mini-TT}},
  year = "2019",
  link = "\url{http://www.cse.chalmers.se/~bengt/papers/GKminiTT.pdf}",
  abstract =
    "This paper presents a formal description of a small functional
    language with dependent types. The language contains data types,
    mutual re- cursive/inductive definitions and a universe of small
    types. The syntax, semantics and type system is specified in such a
    way that the imple- mentation of a parser, interpreter and type
    checker is straightforward.  The main difficulty is to design the
    conversion algorithm in such a way that it works for open expressions.
    The paper ends with a complete implementation in Haskell (around 400
    lines of code).",
  paper = "Coqu19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Corl04,
  author = "Corless, Robert and Watt, Stephen and Zhi, Lihong",
  title = {{QR Factoring to Compute the GCD of Univariate Approximate
            Polynomials}},
  journal = "IEEE Trans. on Signal Processing",
  volume = "52",
  number = "12",
  year = "2004",
  abstract =
    "We present a stable and practical algorithm that uses QR factors
    of the Sylvester matrix to compute the greatest common divisor
    (GCD) of univariate approximate polynomials over $\mathbb{R}[x]$
    or $\mathbb{C}[x]$. An approximate polynomial is a polynomial with
    coefficients that are not known with certainty. The algorithm of
    this paper improves over previously published algorithms by
```

```
    handling the case when common roots are near to or outside the
    unit circle, by splitting and reversal if necessary. The algorithm
    has been tested on thousands of examples, including pairs of
    polynomials of up to degree 1000, and is now distributed as the
    program QRGCD in the SNAP package of Maple 9.",
  paper = "Corl04.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Corl11,
  author = "Corless, Robert and Postma, Erik and Stoutemyer, David",
  title = {{GCD of Multivariate Approximate Polynomials using
            Beautification with the Subtractive Algorithm}},
  booktitle = "Int. Workshop on Symbolic-Numeric Computation",
  publisher = "ACM",
  year = "2011",
  paper = "Corl11.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Cosm95,
  author = "Cosmo, Roberto Di",
  title = {{Isomorphisms of TYpes}},
  publisher = "Birkhauser",
  year = "1995",
  isbn = "0-8176-3763-X",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Cous77,
  author = "Cousot, Patrick and Cousot, Radhia",
  title = {{Abstract Interpretation: A Unified Lattice Model for
            Static Analysis of Programs by Construction or
    Approximation of Fixpoints}},
  booktitle = "Symp. on Principles of Programming Languages",
  pages = "238-252",
  year = "1977",
  paper = "Cous77.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Cous13,
  author = "Cousot, Patrick and Cousot, Radhia and Fahndrich, Manuel
            and Logozzo, Francesco",
  title = {{Automatic Inference of Necessary Preconditions}},
  year = "2013",
  abstract =
    "We consider the problem of {\sl automatic} precondition
    inference. We argue that the common notion of {\sl sufficient}
    precondition inference (i.e., under which precondition is the
    program correct?) imposes too large a burden on callers, and hence
    it is unfit for automatic program analysis. Therefore, we define
    the problem of {\sl necessary} precondition inference (ie. under
    which precondition, if violated, will the program {\sl always} be
    incorrect?). We designed and implemented several new abstract
    interpretation-based analyses to infer atomic, disjunctive,
    universally and existentially quantified necessary preconditions.

    We experimentally validated the analyses on large scale industrial
    code. For unannotated code, the inference algorithms find
    necessary preconditions for almost 64\% of methods which contained
    warnings. In 27\% of these cases the inferred preconditions were
    also {\sl sufficient}, meaning all warnings within the method body
    disappeared. For annotated code, the inference algorithms find
    necessary preconditions for over 68\% of methods with warnings. In
    almost 50\% of these cases the preconditions were also
    sufficient. Overall, the precision improvement obtained by
    precondition inference (counted as the additional number of
    methods with no warnings) ranged between 9\% and 21\%.",
  paper = "Cous13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Cram11,
  author = "Cramer, Marcos and Koepke, Peter and Schroder, Bernhard",
  title = {{Parsing and Disambiguation of Symbolic Mathematics in the
            Naproche System}},
  journal = "LNAI",
  volume = "6824",
  pages = "180-195",
  year = "2011",
  publisher = "Springer",
  abstract =
    "The Naproche system is a system for linguistically analysing and
    proof-checking mathematical texts written in a controlled natural
```

```
    language. The aim is to have an input language that is as close as
    possible to the language that mathematicians actually use when
    writing textbooks or papers.

    Mathematical texts consist of a combination of natural language
    and symbolic mathematics, with symbolic mathematics obeying its
    own syntactic rules. We discuss the difficulties that a program
    for parsing and disambiguating symbolic mathematics must face and
    present how these difficulties have been tackled in the Naproche
    system. One of these difficulties is the fact that information
    provided in the preceding context -- including information
    provided in natural language -- can influence the way a symbolic
    expression has to be disambiguated.",
  paper = "Cram11.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Croc14,
  author = "Crocker, David",
  title = {{Can C++ Be Made as Safe as SPARK?}},
  booktitle = "Proc 2014 HILT",
  isbn = "978-1-4503-3217-0",
  year = "2014",
  abstract =
    "SPARK offers a way to develop formally-verified software in a
    language (Ada) that is designed with safety in mind and is further
    restricted by the SPARK language subset. However, much critical
    embedded software is developed in C or C++ We look at whether and how
    benefits similar to those offered by the SPARK language subset and
    associated tools can be brought to a C++ development environment.",
  paper = "Croc14.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Crou19,
  author = "Crouse, Maxwell and Whitehead, Spencer and
            Abdelaziz, Ibrahim and Makni, Bassem and
            Cornelio, Cristina and Kapanipathi, Pavan and
            Pell, Edwin and Srinivas, Kavitha and
            Thost, Veronika and Witbrock, Michael and
            Fokoue, Achille",
  title = {{A Deep Reinforcement Learning Base Approach to Learning
           Transferable Proof Guidance Strategies}},
  year = "2019",
  linke = "\url{https://arxiv.org/pdf/1911.02065.pdf}",
```

```
  abstract =
    "Traditional first-order logic (FOL) reasoning systems usually
    rely on manual heuristics for proof guidance. We propose TRAIL: a
    system that learns to perform proof guidance using reinforcement
    learning. A key design principle of our system is that it is
    general enough to allow transfer to problems in different domains
    that do not share the same vocabulary of the training set. To do
    so, we developed a novel representation of the internal state of a
    prover in terms of clauses and inference actions, and a novel
    neural-based attention mechanism to learn interactions between
    clauses. We demonstrate that this approach enables the system to
    generalize from training to test data across domains with
    different vocabularies, suggesting that the nerual architecture in
    TRAIL is well suited for representing and processing of logical
    formalisms.",
  paper = "Crou19.pdf"
}
```

———-

— axiom.bib —

```
@misc{Crou19a,
  author = "Crouse, Maxwell and Abdelaziz, Ibrahim and
            Cornelio, Cristina and Thost, Veronika and
            Wu, Lingfei and Forbus, Kenneth and Fokoue, Achille",
  title = {{Improving Graph Neural Network Representations of Logical
            Formulae with Subgraph Pooling}},
  year = "2019",
  linke = "\url{https://arxiv.org/pdf/1911.06904.pdf}",
  abstract =
    "Recent advances in the integration of deep learning with
    automated theorem proving have centered around the representation
    of graph-structured representations, in large part driven by the
    rapidly emerging body of research in geometric deep
    learning. Typically, structure-aware neural methods for embedding
    logical formulae have been variants of either Tree LSTMs or
    GNNs. While more effective than character and token-level
    approaches, such methods have often made representational
    trade-offs that limited their ability to effectively represent the
    global structure of their inputs. In this work, we introduce a
    novel approach for embedding logical formulae using DAG LSTMs that
    is designed to overome the limitations of both Tree LSTMs and
    GNNs. The effectiveness of the proposed framework is demonstrated
    on the tasks of premise selection and proof step classification
    where it achieves the state-of-the-art performance on two standard
    datasets.",
  paper = "Crou19a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Crox05,
  author = "Croxford, Martin and Chapman, Roderick",
  title = {{Correctness by Construction: A Manifesto for
            High-Integrity Software}},
  year = "2005",
  link = "\url{https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.190.867}",
  abstract =
    "High-integrity software systems are often so large that
    conventional development processes cannot get anywhere near
    achieving tolerable defect rates. This article presents an
    approach that has delivered software with very low defect rates
    cost-effectively. We describe the technical details of the
    approach and the results achieved, and discuss how to overcome
    barriers to adopting such best practice approaches. We conclude by
    observing that where such approaches are compatible and can be
    deployed in combination, we have the opportunity to realize the
    extremely low defect rates needed for high integrity software
    composed of many million lines of code.",
  paper = "Crox05.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Crua20,
  author = "Cruanes, Simon",
  title = {{Axiom of Choice}},
  comment = "Axiom: That's why we have the Axiom of Choice",
  year = "2020",
  link = "\url{https://leanprover.zulipchat.com/#narrow/stream/113488-general}"
}
```

———-

— **axiom.bib** —

```
@book{Cutl80,
  author = "Cutland, Nigel",
  title = {{Computability: An Introduction to Recusive Function Theory}},
  publisher = "Cambridge University Press",
  year = "1980",
  isbn = "0-521-22384-9",
  paper = "Cutl80.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Cyph17,
  author = "Cypherpunks",
  title = {{Chapter 4: Verification Techniques}},
  link = "\url{http://www.cypherpunks.to/~peter/04_verif_techniques.pdf}",
  year = "2017",
  abstract = "Wherein existing methods for building secure systems are
    examined and found wanting",
  paper = "Cyph17.pdf",
  keywords = "printed, reviewed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Czap84,
  author = "Czapor, Stephen R. and Geddes, Keith O.",
  title = {{A Comparison of Algorithms for the Symbolic Computation of
            Pade Approximants}},
  booktitle = "International Symbposium on Symbolic and Algebraic
               Manipulation",
  pages = "248-259",
  publisher = "Springer",
  year = "1984",
  comment = "LNCS 174",
  abstract =
    "This paper compares three algorithms for the symbolic computation
    of Pad\'e approximants: an $O(n^3)$ algorithm based on the direct
    solutions of the Hankel linear system exploiting only the property
    of symmetry, and $O(n^2)$ algorithm based on the extended
    Euclidean algorithm, and an $O(n~log^2~n)$ algorithm based on a
    divide-and-conquer version of the extended Euclidean algorithm.
    Implementations of these algorithms are presented and some timing
    comparisons are given. It is found that the $O(n^2)$ algorithm is
    often the fastest for practical sizes of problems and,
    surprisingly, the $O(n^2)$ algorithm wins in the important case
    where the power series being approximated has an exact rational
    function representation.",
  paper = "Czap84.pdf"
}
```

———-

1.2.4 D

— **axiom.bib** —

```
@book{Daep11,
  author = "Daepp, Ulrich and Gorkin, Pamela",
  title = {{Reading, Writing, and Proving}},
```

```
  year = "2011",
  publisher = "Springer",
  isbn = "978-1-4419-9478-3",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Dahl72a,
  author = "Dahl, O.-J. and Dijkstra, E.W. and Hoare, C.A.R",
  title = {{Structured Programming}},
  publisher = "Academic Press",
  year = "1972",
  isbn = "0-12-200556-2",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Daik18,
  author = "Unknown",
  title = {{Daikon Invariant Detector Developer Manual}},
  year = "2018",
  version = "5.7.2",
  link =
    "\url{http://plse.cs.washington.edu/daikon/download/doc/developer.pdf}",
  paper = "Daik18.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Daik18a,
  author = "Unknown",
  title = {{Daikon Invariant Detector User Manual}},
  year = "2018",
  version = "5.7.2",
  link =
    "\url{http://plse.cs.washington.edu/daikon/download/doc/daikon.pdf}",
  paper = "Daik18a.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Dale08,
  author = "van Dalen, Dirk",
  title = {{Logic and Structure}},
  publisher = "Springer",
  year = "2008",
  isbn = "978-3-540-20879-2",
  paper = "Dale08.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Daly18a,
  author = "Daly, Timothy",
  title = {{Proving Axiom Sane Talk}},
  comment = "International Conference on Mathematical Software",
  journal = "LNCS",
  volume = "10931",
  year = "2018",
  paper = "Daly18a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Dama82,
  author = "Damas, Luis and Milner, Robin",
  title = {{Principal Type-schemes for Functional Programs}},
  booktitle = "Proc POPL '82",
  year = "1982",
  pages = "207-212",
  paper = "Dama82.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Dava75,
  author = "Davant, James B.",
  title = {{Wittgenstein on Russell's Theory of Types}},
  journal = "Notre Dame Journal of Formal Logic",
  volume = "16",
  number = "1",
  year = "1975",
  paper = "Dava75.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Davexxa,
  author = "Davenport, James",
  title = {{Integration in Finite Terms}},
  year = "unknown",
  paper = "Davexxa.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Dave84c,
  author = "Davenport, James H.",
  title = {{y'+fy=g}},
  booktitle = "International Symbposium on Symbolic and Algebraic
               Manipulation",
  pages = "341-350",
  publisher = "Springer",
  year = "1984",
  comment = "LNCS 174",
  abstract =
    "In this paper, we look closely at the equation of the title,
    originally consider by Risch, which arises in the integration of
    exponentials. We present a minor improvement of Risch's original
    presentation, a generalisation of that presentation to algebraic
    functions $f$ and $g$, and a new algorithm for the solution of
    this equation. Full details of the last two are to appear
    elsewhere.",
  paper = "Dave84c.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave85c,
  author = "Davenport, James and Padget, Julian",
  title = {{HEUGCD: How Elementary Upperbounds Generate Cheaper Data}},
  journal = "LNCS",
  volume = "204",
  year = "1985",
  booktitle = "EUROCAL '85 European Conference on Computer Algebra",
  publisher = "Springer",
  abstract =
    "The work presented in this paper is a direct consequence of the
```

```
    ideas set forth by Char et al (1984b) describing a new technique
    for computing the greatest common divisor of polynomials.",
  paper = "Dave85c.pdf"
}
```

———-

— **axiom.bib** —

```
@inbook{Dave92f,
  author = "Davenport, James H. and Dewar, Michael C. and
            Richardson, Michael G.",
  title = {{Symbolic and Numeric Computation: the Example of IRENA}},
  booktitle = "Symbolic and Numerical Computation for Artificial Intelligence",
  pages = "347-362",
  year = "1992",
  publisher = "Academic Press",
  abstract =
    "Historically symbolic and numeric computation have pursued
    different lines of evolution, have been written in different
    languages and generally seen to be competitive rather than
    complementary techniques. Even when both were used to solve a
    problem ad hoc methods were used to transfer the data between
    them.

    We first discuss the reasons for this dichotomy, and then present
    IRENA, a system being developed by the authors to present an
    integrated environment with all the facilities of Reduce combined
    with the functionality of the NAG FORTRAN library.

    Not only does IRENA allow the Reduce user to make calls to the NAG
    Library interactively, it also converts a natural input
    representation to the required unnatural FORTRAN one and
    vice-versa on output, which results in a much more intuitive
    interface. Many parameters have default values and so need not be
    supplied by th user.",
  paper = "Dave92f.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Dave01,
  author = "Davenport, James H.",
  title = {{Mathematical Knowledge Management}},
  booktitle = "Int. Workshop on Mathematical Knowledge Management",
  link = "\url{https://www.emis.de/proceedings/MKM2001/davenport.pdf}",
  publisher = "RISC",
  year = "2001",
  paper = "Dave01.pdf",
  keywords = "axiomref"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Dave03,
  author = "Davenport, James H.",
  title = {{MKM from Book to Computer: A Case Study}},
  journal = "LNCS",
  volume = "2594",
  year = "2003",
  abstract =
    "[2] is one of the great mathematical knowledge
    repositories. Nevertheless, it was written for a different era,
    and for human readership. In this paper, we describe the sorts of
    knowledge in one chapter (elementary transcendental functions) and
    the difficulties in making this sort of knowledge formal. This
    makes us ask questions about the nature of a Mathematical
    Knowledge Repository, and whether a database is enough, or whether
    more ‘‘intelligence’’ is required.",
  paper = "Dave03.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave07a,
  author = "Davenport, James H.",
  title = {{What Might ‘‘Understand a Function’’ Mean?}},
  journal = "LNCS",
  volume = "4573",
  year = "2007",
  abstract =
    "Many functions in classical mathematics are largely defined in
    terms of their derivatives, so Bessel’s function is ‘‘the’’
    solution of Bessel’s equation, etc. For definiteness, we need to
    add other properties, such as initial values, branch cuts,
    etc. What actually makes up ‘‘the definition’’ of a function in
    computer algebra? The answer turns out to be a combination of
    arithmetic and analytic properties.",
  paper = "Dave07a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave09,
  author = "Davenport, James H. and Kohlhase, Michael",
```

```
  title = {{Unifying Math Ontologies: A Tale of Two Standards}},
  journal = "LNCS",
  volume = "5625",
  year = "2009",
  abstract =
    "One of the fundamental and seemingly simple aims of mathematical
    knowledge management (MKM) is to develop and standardize formats
    that allow to ‘‘represent the meaning of the objects of
    mathematics’’. The open formats OpenMath and MathML address this,
    but differ subtly in syntax, rigor, and structural viewpoints
    (notably over calculus). To avoid fragmentation and smooth out
    interoperability obstacles, effort is under way to align them into
    a joint format OpenMath/MathML 3. We illustrate the issues that
    come up in such an alignment by looking at three main areas: bound
    variables and conditions, calculus (which relates to the previous)
    and ‘‘lifted’’ n-ary operators.",
  paper = "Dave09.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Davi19,
  author = "Davis, Ernest",
  title = {{The Use of Deep Learning for Symbolic Integration}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1912.05752.pdf}",
  abstract =
    "Lample and Charton (2019) describe a system that uses deep
    learning technology to compute symbolic, indefinite integrals, and
    to find symbolic solutions to first- and second-order ordinary
    differential equations, when the solutions are elementary
    functions. They found that, over a particular test set, the system
    could find solutions more successfully than sophisticated packages
    for symbolic mathematics such as Mathematica run with a long
    time-out. This is an impressive accomplishment, as far as it
    goes. However, the system can handle only a quite limited subset
    of the problems that Mathematica deals with, and the test set has
    significant built-in biases. Therefore the claim that this
    outperforms Mathematica on symbolic integration needs to be very
    much qualified.",
  paper = "Davi19.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Davi94,
  author = "Davis, Martin D. and Sigal, Ron and Weyuker, Elaine J.",
```

```
  title = {{Computability, Complexity, and Languages: Fundamentals of
           Theoretical Computer Science}},
  publisher = "Academic Press",
  year = "1994",
  isbn = "978-0122063824"
}
```

———-

— **axiom.bib** —

```
@misc{Davi00,
  author = "Davies, Rowan and Pfenning, Frank",
  title = {{Intersection Types and Computational Effects}},
  year = "2000",
  link = "\url{http://www.cs.cmu.edu/~fp/papers/icpf00.pdf}",
  abstract =
    "We show that standard formulations of intersection type systems
    are unsound in the presence of computational effects, and propose
    a solution similar to the value restriction for polymorphism
    adopted in the revised definition of Standard ML. It differs in
    that it is not tied to let-expressions and requires an additional
    weakening of the usual subtyping rules. We also present a
    bi-directional type-checking algorithm for the resulting language
    that does not require an excessive amount of type annotations and
    illustrate it through some examples. We further show that the type
    assignment system can be extended to incorporate parametric
    polymorphism. Taken together, we see our system and associated
    type-checking algorithm as a significant step towards the
    introduction of intersection types into realistic programming
    languages. The added expressive power would allow many more
    properties of programs to be stated by the programmer and
    statically verified by the compiler.",
  paper = "Davi00.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Dayx16,
  author = "Day, Martin V.",
  title = {{An Introduction to Proofs and the Mathematical Vernacular}},
  year = "2016",
  publisher = "Virginia Tech",
  link = "\url{www.math.vt.edu/people/day/ProofsBook/IPaMV.pdf}",
  paper = "Dayx16.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Demr06,
  author = "Demri, S. and Laroussinie, F. and Schnoebelen, Ph.",
  title = {{A Parametric Analysis of the State-Explosion Problem in
            Model Checking}},
  journal = "Computer and System Sciences",
  volume = "72",
  pages = "547-575",
  year = "2006",
  abstract =
    "In model checking, the state-explosion problem occurs when one
    checks a {\sl nonflat system}, i.e., a system implicitly described
    as a synchronized product of elementary subsystems. In this paper,
    we investigate the complexity of a wide variedty of model-checking
    problems for nonflat systems under the light of
    {\sl parameterized complexity}, taking the number of synchronized
    components as a parameter. We provide precise complexity measures
    (in the parameterized sense) for most of the problems we
    investigate, and evidence that the results are robust.",
  paper = "Demr06.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Dene19,
  author = "Denes, Maxime and Mortberg, Anders and Siles, Vincent",
  title = {{A Refinement-based Approach to Computational Algebra in COQ}},
  year = "2019",
  link = "\url{www.cse.chalmers.se/~mortberg/papers/coqeal.pdf}",
  abstract =
    "We describe a step-by-step approach to the implementation and
    formal verification of efficient algebraic algorithms. Formal
    specifications are expressed on rich data types which are suitable
    for deriving essential theoretical properties. These
    specifications are then refined to concrete implementations on
    more efficient data structures and linked to their abstract
    counterparts. We illustrate this methodology on key applications:
    matrix rank computation, Winograd's fast matrix product,
    Karatsuba's polynomial multiplication, and the gcd of multivariate
    polynomials.",
  paper = "Dene19.pdf",
  keywords = "printed",
}
```

———-

— **axiom.bib** —

```
@article{Depl02,
```

```
  author = "Deplagne, Eric and Kirchner, Claude",
  title = {{Deduction versus Computation: The Case of Induction}},
  journal = "LNCS",
  volume = "2385",
  year = "2002",
  abstract =
    "The fundamental difference and the essential complementarity
    between computation and deduction are central in computer algebra,
    automated deduction, proof assistants and in frameworks making
    them cooperating. In this work we show that the fundamental proof
    method of induction can be udnerstood and implemented as either
    computation or deduction.

    Inductive proofs can be built either explicitly by making use of
    an induction principle or implicitly by using the so-called
    induction by rewriting and inductionless induction methods. When
    mechanizing proof construction, explicit induction is used in
    proof assistants and implicit induction is used in rewrite based
    automated theorem provers. The two approaches are clearly
    complementary but up to now there was no framework able to
    encompass and to understand uniformly the two methods. In this
    work, we propose such an approach based on the general notion of
    deduction modulo. We extend slightly the original version of the
    deduction modulo framework and we provide modularity properites
    for it. We show how this applies to a uniform understanding of the
    so called induction by rewriting method and how this relates
    directly to the general use of the induction principle.",
  paper = "Depl02.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Ders89,
  author = "Dershowitz, Nachum and Jouannaud, Jean-Pierre",
  title = {{Rewrite Systems}},
  year = "1989",
  number = "478",
  institution = "Laboratoire de Recherche en Informatique",
  link = "\url{http://www.cs.tau.ac.il/~nachum/papers/survey-draft.pdf}",
  paper = "Ders89.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
\article{Ders08,
  author = "Dershowitz, Nachum and Gurevich, Yuri",
  title = {{A Natural Axiomatization of Computability and Proof of
```

```
           Church's Thesis}},
  journal = "Bulletin of Symbolic Logic",
  volume = "14",
  number = "3",
  year = "2008",
  abstract =
    "Churchs Thesis asserts that the only numeric functions that can be
    calculated by effective means are the recursive ones, which are the
    same, extensionally, as the Turing- computable numeric functions. The
    Abstract State Machine Theorem states that every classical algorithm
    is behaviorally equivalent to an abstract state machine. This theorem
    presupposes three natural postulates about algorithmic
    computation. Here, we show that augmenting those postulates with an
    additional requirement regarding basic operations gives a natural
    axiomatization of computability and a proof of Churchs Thesis, as
    Gdel and others suggested may be possible. In a similar way, but with
    a different set of basic oper- ations, one can prove Turings Thesis,
    characterizing the effective string functions, andin particularthe
    effectively-computable functions on string representations of numbers.",
  paper = "Ders08.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Deut85,
  author = "Deutsch, David",
  title = {{Quantum Theory, the Church-Turing Principle and the
           Universal Quantum Computer}},
  journal = "Proc. Royal Society of London",
  volume = "400",
  pages = "97-117",
  year = "1985",
  abstract =
    "It is argued that underlying the Church-Turing hypothesis there
    is an implicit physical assertion. Here, this assertion is
    presented explicitly as a physical principle: 'every finitely
    realizable physical system can be perfectly simulated by a
    universal model computing machine operating by finite
    means'. Classical physics and the universal Turing machine,
    because the former is continuous and the latter discrete, do not
    obey the principle, at leeast in the strong form above. A class of
    model computing machines that is the quantum generalization of the
    class of Turing machines is described, and it is shown that
    quantum theory and the 'universal quantum computer' are compatible
    with the principle. Computing machines resembling the universal
    quantum computer could, in principle, be built and would have many
    remarkable properties not reproducible by any Turing
    machine. These do not include the computation of non-recursive
    functions, but they do include 'quantum parallelism', a method by
    which certain probabilistic tasks can be performed faster by a
```

```
    universal quantum computer than by any classical restriction of
    it. The intuitive explanation of these properties places an
    intolerable strain on all interpretations of quantum theory other
    than Everett's. Some of the numerous connections between the
    quantum theory of computation and the rest of physics are
    explored. Quantum complexity theory allows a physically more
    reasonable definition of the 'complexity' or 'knowledge' in a
    physical system than does classical complexity theory.",
  paper = "Deut85.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Dewa90,
  author = "Dewar, M.C. and Richardson, M.G.",
  title = {{Reconciling Symbolic and Numeric Computation in a
            Practical Setting}},
  booktitle = "DISCO 1990",
  year = "1990",
  pages = "195-204",
  publisher = "Springer",
  paper = "Dewa90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Dewa01,
  author = "Dewar, Mike and Carlisle, David",
  title = {{Mathematical Software: The Next Generation?}},
  booktitle = "Int. Workshop on Mathematical Knowledge Management",
  link = "\url{https://www.emis.de/proceedings/MKM2001/printed/dewar.pdf}",
  publisher = "RISC",
  year = "2001",
  paper = "Dewa01.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Difr17,
  author = "Di Franco, Anthony and Rubio-Gonzalez, Cindy",
  title = {{A Comprehensive Study of Real-World Numerical Bug
            Characteristics}},
```

```
  year = "2017",
  link = "\url{https://web.cs.ucdavis.edu/~rubio/includes/ase17.pdf}",
  abstract =
    "Numerical software is used in a wide variety of applications
    including safety-critical systems, which have stringent
    correctness requirements, and whose failures have catastrophic
    consequences that endanger human life. Numerical bugs are known to
    be particularly difficult to diagnose and fix, largely due to the
    use of approximate representations of numbers such as floating
    point. Understanding the characteristics of numerical bugs is the
    first step to combat them more effectively. In this paper, we
    present the first comprehansive study of real-world numerical
    bugs. Specifically, we identify and carefully examine 269
    numerical bugs from five widely-used numerical software libraries:
    NumPy, SciPy, LAPACK, GNU Scientific Library, and Elemental. We
    propose a categorization of numerical bugs, and discuss their
    frequency, symptoms and fixes. Our study opens new directions in
    the areas of program analysis, testing, and automated program
    repair of numerical software, and provides a collection of
    real-world numerical bugs.",
  paper = "Difr17.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Dijk67,
  author = "Dijkstra, E.W.",
  title = {{A Constructive Approach to the Problem of Program Correctness}},
  year = "1967",
  link = "\url{https://www.cs.utexas.edu/users/EWD/ewd02x/EWD209.PDF}",
  abstract =
    "As an alternative to methods by which the correctness of given
    programs can be established a posteriori, this paper proposes to
    control the process of program generation such as to produce a
    priori correct programs. An example is treated to show the form
    that such a control then might take. This exampe comes from the
    field of parallel programming; the way in which it is treated is
    representative for the way in which a whole multiprogramming
    system has actually been constructed.",
  paper = "Dijk67.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Dijk70,
  author = "Dijkstra, Edsger W.",
  title = {{Concern for Correctness as a Guiding Principle for Program
```

```
            Composition}},
  year = "1970",
  comment = "EWD288",
  link = "\url{https://www.cs.utexas.edu/users/EWD/transcriptions/EWD02xx/EWD288.html}",
  paper = "Dijk70.html",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Dijk74,
  author = "Dijkstra, Edsger W.",
  title = {{Programming as a Discipline of Mathematical Nature}},
  year = "1974",
  comment = "EWD361",
  link = "\url{https://www.cs.utexas.edu/users/EWD/ewd03xx/EWD361.PDF}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Dijk78,
  author = "Dijkstra, Edsger W.",
  title = {{A Political Pamphlet From The Middle Ages}},
  year = "1978",
  comment = "EWD638",
  link = "\url{https://www.cs.utexas.edu/users/EWD/transcriptions/EWD06xx/EWD638.html}"
}
```

———-

— **axiom.bib** —

```
@book{Dijk88,
  author = "Dijkstra, Edsger W. and Feijen, W.H.J",
  title = {{A Method of Programming}},
  publisher = "Addison Wesley",
  year = "1988",
  isbn = "0-201-17536-3",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Dijk88a,
```

```
  author = "Dijkstra, Edsger W.",
  title = {{On the Cruelty of Really Teaching Computing Science}},
  year = "1988",
  link = "\url{http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1036.PDF}",
  paper = "Dijk88a.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Dijk90,
  author = "Dijkstra, Edsger W. and Scholten, Carel S.",
  title = {{Predicate Calculus and Program Semantics}},
  publisher = "Springer",
  year = "1990",
  isbn = "0-387-96957-8",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Dijk00,
  author = "Dijkstra, Edsger W.",
  title = {{EWD1305 Archive: Answers to questions from students of
           Software Engineering}},
  link = "\url{https://www.cs.utexas.edu/users/EWD/transcriptions/EWD13xx/EWD1305.html}",
  year = "2000"
}
```

———-

— **axiom.bib** —

```
@misc{Dijk12,
  author = "Dijkstra, Edsger W.",
  title = {{Some Meditations on Advanced Programming}},
  link = "\url{https://www.cs.utexas.edu/users/EWD/transcriptions/EWD00xx/EWD32.html}",
  year = "2012"
}
```

———-

— **axiom.bib** —

```
@book{Ding96,
  author = "Ding, C. and Pei, D. and Salomaa, A.",
  title = {{Chinese Remainder Theorem, Applications in Computing,
            Coding Cryptography}},
```

```
  publisher = "World Scientific",
  year = "1996"
}
```

———-

— **axiom.bib** —

```
@misc{Doct11,
  author = "Doctorow, Cory",
  title = {{The Coming War on General Computation}},
  year = "2011",
  link = "\url{http://opentranscripts.org/transcript/coming-war-on-general-computation}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Domi08,
  author = "Dominguez, Cesar",
  title = {{Formalizing in Coq Hidden Algebras to Specify Symbolic
            Computation Systems}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
  abstract =
    "This work is an attempt to formalize, using the Coq proof
    assistant, the algebraic specification of the data structures
    appearing in two symbolic computation systems for algebraic
    topology called EAT and Kenzo. The specification of these
    structures have been obtained through an operation, called imp
    operation, between different specification frameworks as standard
    algebraic specifications and hidden specifications. Resuing
    previous Coq implementations of universal algebra and category
    theory we have proposed a Coq formalization of the imp operation,
    extending the representation to the particular hidden algebras
    which take part in this operation.",
  paper = "Domi08.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Domi10,
  author = "Dominguez, Cesar and Rubio, Julio",
  title = {{Computing in Coq with Infinite Algebraic Data Structures}},
  journal = "LNCS",
  volume = "6167",
```

```
  year = "2010",
  abstract =
    "Computational content encoded into constructive type theory
    proofs can be used to make computing experiments over concrete
    data structures. In this paper, we explore this possibility when
    working in Coq with chain complexes of infinite type (that is to
    say, generated by infinite sets) as a part of the formalization of
    a hierarchy of homological algebra structures.",
  paper = "Domi10.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Domi19,
  author = "Dominus, Mark",
  title = {{The Least Common Divisor and the Greatest Common Multiple}},
  link = "\url{https://blog.plover.com/math/gcm.html}",
  year = "2019"
}
```

———-

— **axiom.bib** —

```
@misc{Dona77,
  author = "Donahue, J.",
  title = {{On the semantics of ‘‘Data Type’’}},
  comment = "Cornell University",
  year = "1977"
}
```

———-

— **axiom.bib** —

```
@book{Dona92,
  author = "Donald, B.R. and Kapur, D. and Mundy, J.L.",
  title = {{Symbolic and Numerical Computation for Artificial
            Intelligence}},
  publisher = "Academic Press Limited",
  year = "1992",
  isbn = "0 12 220535 9",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Door20,
  author = "Doorn, Floris van and Ebner, Gabriel and
            Lewis, Robert Y.",
  title = {{Maintaining a Library of Formal Mathematics}},
  link = "\url{https://arxiv.org/pdf/2004.03673}",
  year = "2020",
  abstract =
    "The Lean mathematical library mathlib is developed by a community
    of users with very different backgrounds and levels of
    experience. To lower the barrier of entry for contributors and to
    lessen the burden of reviewing contributions, we have developed a
    number of tools for the library which check proof developments for
    subtle mistakes in the code and generate documentation suited for
    our varied audience.",
  paper = "Door20.pdf"
}
```

———-

— **axiom.bib** —

```
@inbook{Dowe01,
  author = "Dowek, Gilles",
  title = {{Handbook of Automated Reasoning, Vol II}},
  publisher = "Elsevier Science",
  year = "2001",
  chapter = "16",
  pages = "1009-1062"
}
```

———-

— **axiom.bib** —

```
@misc{Dris20,
  author = "Driscoll, Kevin R.",
  title = {{Murphy Was An Optimiist}},
  year = "2020",
  link = "\url{www.rvs.uni-bielefeld.de/publications/DriscollMurphyv19.pdf}",
  comment = "Kevin.Driscoll@Honeywell.com",
  paper = "Dris20.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Daum07,
  author = "Daumas, Marc and Lester, David and Munoz, Cesar",
  title = {{Verified Real Number Calculations: A Library for Interval
```

```
           Arithmetic}},
  year = "2007",
  link = "\url{https://arxiv.org/pdf/0708.3721.pdf}",
  abstract =
    "Real number calculations on elementary functions are remarkably
    difficult to handle in mechanical proofs. In this paper, we show
    how these calculations can be performed within a theorem prover or
    proof assistant in a convenient and highly automated as well as
    interactive way. First, we formally establish upper and lower
    bounds for elementary functions. Then, based on these bounds, we
    develop a rational interval arithmetic where real number
    calculations take place in an algebraic setting. In order to
    reduce the dependency effect of interval arithmetic, we integrate
    two techniques: interval splitting and taylor series
    expansions. This pragmatic approach has been developed, and
    formally verified, in a theorem prover. The formal development
    also includes a set of customizable strategies to automate proofs
    involving explicit calculations over real numbers. Our ultimate
    goal is to provide guaranteed proofs of numerical properties with
    minimal human theorem-prover interaction.",
  paper = "Daum07.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Dowe15,
  author = "Dowek, Gilles",
  title = {{Deduction Modulo Theory}},
  year = "2015",
  link = "\url{http://arxiv.org/pdf/1501.06523.pdf}",
  paper = "Dowe15.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Dunf13,
  author = "Dunfield, Joshua and Krishnaswami, Neelakantan R.",
  title = {{Complete and Easy Bidirectional Typechecking for
           Higher-Rank Polymorphism}},
  booktitle = "ICFP'13",
  publisher = "ACM",
  year = "2013",
  link = "\url{https://arxiv.org/pdf/1306.6032.pdf}",
  abstract =
    "Bidirectional typechecking, in which terms either synthesize a
    type or are checked against a known type, has become popular for
```

```
    its scalability (unlike Damas-Milner type inference, bidirectional
    typing remains decidable even for very expressive type systems),
    its error reporting, and its relative ease of
    implementation. Following design principles from proof theory,
    bidirectional typing can be applied to many type constructs. The
    principles underlying a bidirectional approach to polymorphism,
    however, are less obvious. We give a declarative, bidirectional
    account of higher-rank polymorphism, grounded in proof theory;
    this calculus enjoys many properties such as $\eta$-reduction and
    predictability of annotations. We give an algorithm for
    implementing the declarative system; our algorithm is remarkably
    simple and well-behaved, despite being both sound and complete.",
  paper = "Dunf13.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Dunf18,
  author = "Dunfield, Joshua and Krishnaswami, Neelakantan R.",
  title = {{Sound and Complete Bidirectional Typechecking for
            Higher-Rank Polymorphism with Existentials and
            Indexed Types}},
  year = "2018",
  link = "\url{https://arxiv.org/pdf/1601.05106.pdf}",
  abstract =
    "Bidirectional typechecking, in which terms either synthesize a
    type or are checked against a known type, has become popular for
    its applicability to a variety of type systems, its error
    reporting, and its ease of implementation. Following principles
    from proof theory, bidirectional typing can be applied to many
    type constructs. The principles underlying a bidirectional
    approach to indexed types {\sl generalized algebraic datatypes}
    are less clear. Building on proof-theoretic treatments of
    equality, we give a declarative specification of typing based on
    {\sl focalization}. This approach permits declarative rules for
    coverage of pattern matching, as well as support for first-class
    existential types using a focalized subtyping judgment. We use
    refinement types to avoid explicitly passing equality proofs in
    our term syntax, making our calculus similar to languages such as
    Haskell and OCaml. We also extend the declarative specification
    with an explicit rules for deducing when a type is principal,
    permitting us to give a complete declarative specification for a
    rich type system with significant type inference. We also give a
    set of algorithmic typing rules, and prove that it is sound and
    complete with respect to the declarative system. The proof
    requires a number of technical innovations, including proving
    soundness and completeness in a mutually recursive fashion.",
  paper = "Dunf18.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Dunf19,
  author = "Dunfield, Joshua and Krishnaswami, Neel",
  title = {{Bidirection Typing}},
  year = "2019",
  link = "\url{https://www.cl.cam.ac.uk/~nk480/bidir-survey.pdf}",
  abstract =
    "Bidirectional typing combines two modes of typing: type checking,
    which checks that a program satisfies a known type, and type
    synthesis, which determines a type from the program. Using
    checking enables bidirectional typing to break the decidability
    barrier of Damas-Milner approaches; using synthesis enables
    bidirectional typing to avoid the large annotation burden of
    explicitly typed languages. In addition, bidirectional typing
    improves error locality. We highlight the design principles that
    underlie bidirectional type systems, survey the development of
    bidirectional typing from the prehistoric period before Pierce and
    Turner's local type inference to the present day, and provide
    guidance for future investigations.",
  paper = "Dunf19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Dupe96,
  author = "Dupee, Brian J. and Davenport, James H.",
  title = {{An Intelligent Interface to Numerical Routines}},
  journal = "LNCS",
  volume = "1128",
  pages = "252-262",
  year = "1996",
  abstract =
    "Links from Computer Algebra Systems to Numerical Libraries have
    been increasingly made available. However, they remain, like the
    numerical routines which comprise the libraries, difficult to use
    by a novice and there is little help in choosing the appropriate
    routine for any given problem, should there be a choice.

    Computer Algebra Systems use generic names for each problem
    area. For example, 'integrate' (or 'int') is used for integration
    of a function, whatever method the code may use. Numeric
    interfaces still use different names for each method together with
    a variety of extra parameters, some of which may be
    optional. Ideally, we should extend the generic name structure to
    cover numerical routines. This would then, necessarily, require
    algorithms for making an assessment of the efficacy of different
    methods where such a choice exists.
```

```
    This paper considers the link to the NAG Fortran Library from
    version 2.0 of Axiom and shows how we can build on this to extend
    and simplify the interface using an expert system for choosing and
    using the numerical routines.",
  paper = "Dupe96.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
\article{Dybj94,
  author = "Dybjer, Peter",
  title = {{Inductive Families}},
  journal = "Formal Aspects of Computing",
  volume = "6",
  number = "4",
  pages = "440-465",
  year = "1994",
  abstract =
    "A general formulation of inductive and recursive definitions in
    Martin-Lof's type theory is presented. It extends Backhouse's
    'Do-It-Yourself Type Theory' to include inductive definitions of
    families of sets and definitions of functions by recursion on the
    way elements of such sets are generated. The formulation is in
    natural deduction and is intended to be a natural generalization
    to type theory of Martin-Lof's theory of iterated inductive
    definitions of predicate logic.

    Formal criteria are given for correct formation and introduction
    rules of a new set former capturing definition by strictly
    positive, iterated, generalized induction. Moreover, there is an
    inversion principle for deriving elimination and equality rules
    from the formation and introduction rules. Finally, there is an
    alternative schematic presentation of definition by recursion.

    The resulting theory is a flexible and powerful language for
    programming and constructive mathematics. We hint at the wealth of
    possible applications by showing several basic examples: predicate
    logic, generalized induction, and a formalization of the untyped
    lambda calculus.",
  paper = "Dybj94.pdf",
  keywords = "printed"
}
```

———-

1.2.5 E

— **axiom.bib** —

```
@article{Ehri80,
  author = "Ehrig, H. and Kreowski, H.-J. and Padawitz, P.",
  title = {{Algebraic Implementation of Abstract Data Types: Concepts,
            Syntax, Semanics and Correctness}},
  journal = "LNCS",
  volume = "85",
  year = "1980",
  booktitle = "Automata, Languages and Programming",
  publisher = "Springer",
  abstract =
    "A new concept for the implementation of abstract data types is
    proposed: Given algebraic specifications SPEC0 and SPEC1 of
    abstract data types ADT0 and ADT1 an implentation of ADT0 by ADT1
    is defined spearately on the syntactical level of specifications
    and on the semantical level of algebras. This concept is shown to
    satisfy a number of conceptual requirements for the implementation
    of abstract data types. Several correctness criteria are given and
    illustrating examples are provided.",
  paper = "Ehri80.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Ehri80a,
  author = "Ehrig, Hartmut and Kreowski, Hans-Jorg and Thatcher, James
            and Wagner, Eric and Wright, Jesse",
  title = {{Parameterized Data Types in Algebraic Specification Languages}},
  journal = "LNCS",
  volume = "85",
  year = "1980",
  booktitle = "Automata, Languages and Programming",
  publisher = "Springer",
  paper = "Ehri80a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Ehri82,
  author = "Ehrig, H. and Kreowski, H.-J. and Mahr, B. and Padawitz, P.",
  title = {{Algebraic Implementation of Abstract Data Types}},
  journal = "Theoretical Computer Science",
  volume = "20",
  pages = "209-263",
  year = "1982",
  abstract =
    "Starting with a review of the theory of algebraic specifications
```

```
    in the sense of the ADJ-group a new theory for algebraic
    implementation of abstract data types is presented.

    While main concepts of this new theory were given already at
    several conferences this paper provides the full theory of
    algebraic implementations developed in Berlin except of complexity
    considerations which are given in a separate paper. This new
    concept of algebraic implementations includes implementations for
    algorithms in specific programming languages and on the other hand
    it meets also the requirements for stepwise refinement of
    structured programs and software systems as introduced by Dijkstra
    and Wirth. On the syntactical level an algebraic implementation
    corresponds to a system of recursive programs while the semantical
    level is defined by algebraic constructions, called SYNTHESIS,
    RESTRICTION and IDENTIFICATION. Moreover the concept allows
    composition of implementations and a rigorous study of
    correctness. The main results of the paper are different kinds of
    correctness criteria which are applied to a number of illustrating
    examples including the implementation of sets by hash-tables.
    Algebraic implementations of larger systems like a histogram or a
    parts system are given in separate case studies which, however,
    are not included in this paper.",
  paper = "Ehri82.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Ehri85,
  author = "Ehrig, Hartmut and Mahr, Bernd",
  title = {{Fundamentals of Algebraic Specification 1: Equations and
           Initial Semantics}},
  publisher = "Springer Verlag",
  year = "1985",
  isbn = "0-387-13718-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Ehri85a,
  author = "Ehrig, Hartmut and Mahr, Bernd",
  title = {{Fundamentals of Algebraic Specification 2: Module
            Specifications and Constraints}},
  publisher = "Springer Verlag",
  year = "1985",
  isbn = "0-387-51799-5",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Eise14,
  author = "Eisenberg, Richard A. and Vytiniotis, Dimitrios and
            Jones, Simon Peyton and Weirich, Stephanie",
  title = {{Closed Type Families with Overlapping Equations}},
  booktitle = "POPL 14",
  year = "2014",
  abstract =
    "Open, type-level functions are a recent innovatino in Haskell
    that move Haskell towards the expressiveness of dependent types,
    while retaining the lok and feel of a practial programming
    language. This paper shows how to increase expressiveness still
    further, by adding closed type functions whose equations may
    overlap, and may have non-linear patterns over an open type
    universe. Although practically useful and simple to implement,
    these features go {\sl beyond} conventional dependent type theory
    in some respects, and have a subtle metatheory.",
  paper = "Eise14.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Eise16,
  author = "Eisenberg, Richard A.",
  title = {{Dependent Types in Haskell: Theory and Practice}},
  school = "University of Pennsylvania",
  year = "2016",
  abstract =
    "Haskell, as implemented by the Glasgow Haskell Compiler (GHC),
    has been adding new type-level programming features for some
    time. Many of these features -- generalized algebraic datatypes
    (GADT)s, type families, kind polymorphism, and promoted datatypes
    -- have brought Haskell to the doorstep of dependent types. Many
    dependently typed programs can even currently be encoded, but
    often the constructions are painful.

    In this dissertation, I describe Dependent Haskell, which supports
    full dependent types via a backward-compatible extension to
    today's Haskell. An important contribution to this work is an
    implementation, in GHC, of a portion of Dependent Haskell, with
    the rest to follow. The features I have implemented are already
    released, in GHC 8.0. This dissertation contains several practical
    examples of Dependent Haskell code, a full description of the
    differences between Dependent Haskell and today's Haskell, a novel
    dependently typed lambda-calculus (called PICO) suitable for use
```

```
    as an intermediate language for compiling Dependent Haskell, and a
    type inference and elaboration algorithm, BAKE, that translates
    Dependent Haskell to type-correct PICO. Full proofs of type safety
    of PICO and the soundness of BAKE are included in the appendix.",
  paper = "Eise16.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Elba20,
  author = "Elbakyan, Alexandra and Francis, Robin",
  title = {{Should Knowledge Be Free?}},
  year = "2020",
  link = "\url{https://www.youtube.com/watch?v=PriwCi6SzLo}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Elli89,
  author = "Elliott, Conal M.",
  title = {{Higher-order Unification with Dependent Function Types}},
  booktitle = "Rewriting Techniques and Applications",
  year = "1989",
  pages = "121-136",
  abstract =
    "Roughly fifteen years ago, Huet developed a complete semidecision
    algorithm for unification in the simply typed $\lambda$-calculus
    ($\lambda_\rightarrow$). In spite of the undecidability of this
    problem, his algorithm is quite usable in practice. Since then, many
    important applications have come about in such areas as theorem
    proving, type inference, program transformation, and machine learning.

    Another development is the discovery that by enriching
    $\lambda_\rightarrow$ to include {\sl dependent function types},
    the resulting calculus ($\lambda_\Pi$) forms the basis of a very
    elegant and expressive Logical Framework, encompassing the syntax,
    rules, and proofs for a wide class of logics.

    This paper presents an algorithm in the spirit of Huet's, for
    unification in $\lambda_\Pi$. This algorithm gives us the best
    of both worlds: the automation previously possible in
    $\lambda_\rightarrow$ and the greatly enriched expressive power of
    $\lambda_\Pi$. It can be used to considerable advantage in many
    of the current applications of Huet's algorithm, and has important
    new applications as well. These include automated and semi-automated
    theorem proving in encoded logics, and automatic type inference in a
    variety of encoded languages."
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Elli90,
  author = "Elliott, Conal and Pfenning, Frank",
  title = {{A Semi-Functional Implementation of a Higher-Order Logic
            Programming Language}},
  year = "1990",
  link = "\url{http://www.cs.cmu.edu/~fp/papers/elpsml90.pdf}",
  comment = "\url{http://www.cs.cmu.edu/~fp/papers/elpsml-paper.tar.gz}",
  paper = "Elli90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Elli17,
  author = "Elliott, Conal",
  title = {{Compiling to Categories}},
  booktitle = "Proc. ACM Program. Lang. Vol 1",
  publisher = "ACM",
  year = "2017",
  link = "\url{http://conal.net/papers/compiling-to-categories/compiling-to-categories.pdf}",
  abstract =
    "It is well-known that the simply typed lambda-calculul is modeled
    by any cartesian closed category (CCC). This correspondence
    suggests giving typed functional programs a variety of
    interpretations, each corresponding to a different category. A
    convenient way to realize this idea is as a collection of
    meaning-preserving transformations added to an existing compiler,
    such as GHC for Haskell. This paper describes automatic
    differentiation, incremental computation, and interval
    analysis. Each such interpretation is a category easily defined in
    Haskell (outside of the compiler). The general technique appears
    to provide a compelling alternative to deeply embedded
    domain-specific languages.",
  paper = "Elli17.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Elli18,
  author = "Ellis, Ferris",
```

```
  title = {{Strength in Numbers: Unums and the Quest for Reliable
           Arithmetics}},
  year = "2018",
  link = "\url{https://www.youtube.com/watch?v=nVNYjmj_qbY}",
  abstract =
    "In the land of computer arithmetic, a tyrant has ruled since its very
    beginning: the floating point number. Under its rule we have all
    endured countless hardships and cruelties. To this very day the
    floating point number still denies 0.1 + 0.2 == 0.3 and returns
    insidious infinities to software developers everywhere. But a new hero
    has entered fray: the universal number (unum). Can it topple the float
    number system and its century long reign?

    This talk will introduce unums, explain their benefits over floating
    point numbers, and examine multiple real world examples comparing the
    two. For those not familiar with floating point numbers and their
    pitfalls, this talk also includes a primer on the topic. Code examples
    are in Rust, though strong knowledge of the language is not needed."
}
```

———-

— **axiom.bib** —

```
@inproceedings{Emma85,
  author = "Emmanuel, Kounalis",
  title = {{Completeness in Data Type Specifications}},
  booktitle = "European COnference on Computer Algebra",
  publisher = "Springer",
  pages = "348-362",
  year = "1985",
  comment = "LNCS 204",
  paper = "Emma85.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Ende01,
  author = "Enderton, Herbert B.",
  title = {{A Mathematical Introduction to Logic}},
  publisher = "Harcourt Academic Press",
  year = "2001",
  isbn = "0-12-238452-0",
  paper = "Ende01.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Erem19,
  author = "Eremondi, Joseph and Tanter, Eric and Garcia, Ronald",
  title = {{Approximate Normalization for Gradual Dependent Types}},
  booktitle = "Inter. Conf. on Functional Programming",
  publisher = "ACM",
  year = "2019",
  abstract =
    "Dependent types help programmers write highly reliable
    code. However, this reliability comes at a cost: it can be
    challenging to write new prototypes in (or migrate old code to)
    dependently-typed programming languages. Gradual typing makes
    static type disciplines more flexible, so an appropriate notion of
    gradual dependent types could fruitfully lower this cost. However,
    dependent types raise unique challenges for gradual
    typing. Dependent typechecking involves the execution of program
    code, but gradually-typed code can signal runtime type errors or
    diverge. These runtime errors threaten the soundness guarantees
    that make dependen types so attractive, while divergence spoils
    the type-driven programming experience.

    this paper presents GDTL, a gradual dependently-typed language
    that emphasizes pragmatic dependently-typed programming. GDTL
    fully embeds both an untyped and dependently-typed language, and
    allows for smooth transitions between the two. In addition to
    gradual types we introduce gradual terms, which allow the user to
    be imprecise in type indices and to omit proof terms; runtime
    checks ensure type safety. To account for nontermination and
    failure, we distinguish between compile-type normalization and
    run-time execution: compile-time normalization is approximate but
    total, while runtime execution is exact, but may fail or
    diverge. We prove that GDTL has decidable typechecking and
    satisfies all the expected properties of gradual languages. In
    particular, GDTL satisfies the static and dynamic gradual
    guarantees: reducing type precision preserves typedness, and
    altering type precision does not change program behavior outside
    of dynamic type failures. To prove these properties, we were led
    to establish a novel normalization gradual guarantee thata
    captures the monotonicity of approximate normalization with
    respect to imprecision.",
  paper = "Erem19.pdf"
}
```

———-

— axiom.bib —

```
@misc{Ersh77,
  author = "Ershov, A.P.",
  title = {{On the Essence of Compilation}},
  comment = "Proc. IFIP Working Conf. on Formal Description of
             Programming Concepts, Vol. 1",
  year = "1977"
}
```

———-

1.2.6 F

— **axiom.bib** —

```
@misc{Fade17,
  author = "Unknown",
  title = {{Strongly Connected and Completely Specified Moore
            Equivalent of a Mealy Machine}},
  year = "2017",
  link = "\url{https://cs.stackexchange.com/questions/81127/strongly-connected-and-completely-specified-moore-eq
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Farm90,
  author = "Farmer, William M.",
  title = {{A Partial Functions Version of Church's Simple Theory of Types}},
  journal = "The Journal of Symbolic Logic",
  volume = "55",
  number = "3",
  year = "1990",
  pages = "1269-1291",
  abstract =
    "Church's simple theory of types is a system of higher-order logic in
     which functions are assumed to be total. We present in this paper a
     version of Church's system called PF in which functions may be
     partial. The semantics of PF, which is based on Henkin's
     general-models semantics, allows terms to be nondenoting but requires
     formulas to always denote a standard truth value. We prove that PF is
     complete with respect to its semantics. The reasoning mechanism in PF
     for partial functions corresponds closely to mathematical practice,
     and the formulation of PF adheres tightly to the framework of
     Church's system.",
  paper = "Farm90.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Farm95,
  author = "Farmer, William M. and Guttman, J.D. and Thayer, F.J.",
  title = {{Contexts in Mathematical Reasoning and Computation}},
  journal = "J. of Symbolic Computation",
```

```
  volume = "19",
  pages = "201-216",
  year = "1995",
  abstract =
    "Contexts are sets of formulas used to manage the assumptions that
    arise in the course of a mathematical deduction or
    calculation. Although context-dependent reasoning is commonplace in
    informal mathematics, most contemporary symbolic computation systems
    do not utilize contexts in sophisticated ways. This paper describes
    some context-based techniques for symbolic computation, including
    techniques for reasoning about definedness, simplifying abstract
    algebraic expressions, and computing with theorems. All of these
    techniques are implemented in the IMPS Interactive Mathematical Proof
    System. The paper also proposes a general mathematics laboratory that
    combines the functionality of current symbolic computation systems
    with the facilities of a theorem proving system like IMPS.",
  paper = "Farm95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Farm95a,
  author = "Farmer, William M. and Guttman, Joshua D. and
            F. Javier Thayer",
  title = {{The IMPS User's Manual}},
  link = "\url{http://imps.mcmaster.ca/manual/manual.html}",
  year = "1995"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Farm01,
  author = "Farmer, William M. and Mohrenschildt, Martin v.",
  title = {{A Formal Framework for Managing Mathematics}},
  booktitle = "Int. Workshop on Mathematical Knowledge Management",
  link = "\url{https://www.emis.de/proceedings/MKM2001/farmer.pdf}",
  publisher = "RISC",
  year = "2001",
  abstract =
    "Mathematics is a process of creating, exploring, and connnecting
    mathematical models. This paper present a fromal framework for
    managing the mathematics process as well as the mathematical
    knowledge produced by the process.  The central idea of the
    framework is the notion of a biform theory which is simultaneously
    an axiomatic theory and an algorithmic theory. Representing a
    collection of mathematical models, a biform theory provides a
    formal context for both deduction and computation, constructing
```

```
    sound deductive and computational rules, and developing networks
    of biform theories linked by interpretations. The framework is not
    tied to a specific underlying logic; it can be used with many
    popular logics such as first order logic, simple type theory, and
    set theory. Many of the ideas and mechanisms used in the framework
    are inspired by the IMPS Interactive Mathematical Proof System.",
  paper = "Farm01.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Farm13a,
  author = "Farmer, William M.",
  title = {{The Formalization of Syntax-Based Mathematical Algorithms
            Using Quotation and Evaluation}},
  journal = "LNCS",
  volume = "7961",
  year = "2013",
  abstract =
    "Algorithms like those for differentiating functional expressions
    manipulate the syntactic structure of mathematical expressions in
    a mathematically meaningful way. A formalization of such an
    algorithm should include a specification of its computational
    behavior, a specification of its mathematical meaning, and a
    mechanism for applying the algorithm to actual
    expressions. Achieving these goals requires the ability to
    integrate reasoning about the synta of the expressions with
    reaoning about what the expressions mean. A syntax framework is a
    mathematical structure that is an abstract model for a syntax
    reasoning system. It contains a mapping of expressions to
    syntactic values that represent the syntactic structures of the
    expressions; a language for reasoning about syntactic values; a
    quotation mechanism to refer to the syntactic value of an
    expression; and an evaluation mechanism to refer to the value of
    the expression represented by a syntactic value. We present and
    compare two approaches, based on instances of a syntax framework,
    to formalize a syntax-based mathematical algorithm in a formal
    theory $T$. In the first approach the syntactic values for the
    expressions manipulated by the algorithm are members of an
    inductive type in $T$, but quotation and evaluation are functions
    defined in the matatheory of $T$. In the second approach every
    expression in $T$ is represented by a syntactic value, and
    quotation and evalution are operators in $T$ itself.",
  paper = "Farm13a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Farm16,
  author = "Farmer, William M.",
  title = {{Incorporating Quotation and Evaluation into Church's Type
            Theory: Syntax and Semantics}},
  journal = "LNCS",
  volume = "9791",
  year = "2016",
  abstract =
    "CTT$_{qe}$ is a version of Church's type theory that includes
    quotation and evaluation operators that are similar to quote and
    eval in the Lisp programming language. With quotation and
    evaluation it is possible to reason in CTT$_{qe}$ about the
    interplay of the syntax and semantics of expressions and, as a
    result, to formalize syntax-based mathematical algorithms. We
    present the syntax and semantics of CTT$_{qe}$ and give several
    examples that illustrate the usefulness of having quotation and
    evaluation in CTT$_{qe}$. We do not give a proof system for
    CTT$_{qe}$ but we do sketch what a proof system could look like.",
  paper = "Farm16.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Farm17,
  author = "Farmer, William M.",
  title = {{Theory Morphisms in Church's Type Theory with Quotation
            and Evaluation}},
  journal = "LNCS",
  volume = "10383",
  year = "2017",
  abstract =
    "CTT$_{qe}$ is a version of Church's type theory with global
    quotation and evaluation operators that is engineered to reason
    about the interplay of syntax and semantics and to formalize
    syntax-level mathematical algorithms. CTT$_{uqe}$ is a variant of
    CTT$_{qe}$ that admits undefined expressions, partial functions,
    and multiple base types of individuals. It is better suited than
    CTT$_{qe}$ as a logic for building networks of theories connected
    by theory morphisms. This paper presents the syntax and semantics
    of CTT$_{uqe}$, defines a notion of a theory morphism from one
    CTT$_{uqe}$ theory to another, and gives two simple examples
    involving monoids that illustrate the use of theory morphisms in
    CTT$_{qe}$.",
  paper = "Farm17.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Farv20,
  author = "Farvardin, Kavon",
  title = {{From Folklore to Fact: Comparing Implementation of Stacks
            and Continuations}},
  booktitle = "Programming Language Design and Implementation",
  publisher = "ACM SIGPLAN",
  year = "2020",
  link = "\url{https://kavon.farvard.in/papers/pldi20-stacks.pdf}",
  abstract =
    "The efficient implementation of function calls and non-local
    control transfers is a critical part of modern language
    implementations and is important in the implementation of
    everything from recursion, higher-order functions, concurrency and
    coroutines, to task-based parallelism. In a compiler, these
    features can be supported by a variety of mechanisms, including
    call stacks, segmented stacks, and heap-allocated continuation
    closures.

    An implementor of a high-level language with advanced control
    features might ask the question ``what is the best choice for my
    implementation?'' Unfortunately, the current literature does not
    provide much guidance, since previous studies suffer from various
    flaws in methodology and are outdated for modern hardware. In the
    absence of recent, well-normalized measurements and a holistic
    overview of their implementation specifics, the path of least
    resistance when choosing a strategy is to trust folklore, but the
    folklore is also suspect.

    This paper attempts to rememdy this situation by providing an
    ``apples-to-apples'' comparison of five different approaches to
    implementing call stacks and continuations. This comparison uses
    the same source language, compiler pipeline, LLVM-backend, and
    runtime system, with the only differences being those required by
    the differences in implementation strategy. We compare the
    implementation challenges of the different approaches, their
    sequential performance, and their suitability to support advanced
    control mechanisms, including supporting heavily threaded code. In
    addition to the comparison of implementation strategies, the
    paper's contributions also include a number of useful
    implementation techniques that we discover along the way.",
  paper = "Farv20.pdf"
}
```

———-

— axiom.bib —

```
@article{Fass04,
  author = "Fass, Leona F.",
  title = {{Approximations, Anomalies and ``The Proof of Correctness Wars''}},
  journal = "ACM SIGSOFT Software Engineering Notes",
  volume = "29",
  number = "2",
```

```
  year = "2004",
  abstract =
    "We discuss approaches to establishing 'correctness' and describe the
    usefulness of logic-based model checkers for producing better
    practical system designs.  While we could develop techniques for
    'constructing correctness' in our theoretical behavioral-modeling
    research, when applied to Real World processes such as software
    development only approximate correctness might be established and
    anomalous behaviors subsequently found.  This we view as a positive
    outcome since resultant adaptation, or flaw detection and correction,
    may lead to improved development and designs.  We find researchers
    employing model checking as a formal methods tool to develop empirical
    techniques have reached similar conclusions. Thus we cite some
    applications of model checking to generate tests and detect defects in
    such Real World processes as aviation system development,
    fault-detection systems, and security.",
  paper = "Fass04.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Fate96a,
  author = "Fateman, Richard J.",
  title = {{Why Computer Algebra Systems Can't Solve Simple
            Equations}},
  year = "1996",
  link = "\url{https://people.eecs.berkeley.edu/~fateman/papers/y=z2w.pdf}",
  paper = "Fate96a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate99c,
  author = "Fateman, Richard and James, Timothy",
  title = {{Analysis of a Web User Interface for Mathematics}},
  year = "1999",
  link = "\url{https://people.eecs.berkeley.edu/~fateman/papers/tjames.pdf}",
  abstract =
    "What can we learn from the range of over 7000 queries made to
    TILU, a symbolic integration problem-solving server during the
    course of more than two years? We have saved all queries during
    this experiment, and based on our analysis, and experimented with
    improvements in the computer-human interaction components of our
    web server. We believe our experience will be useful in the design
    of similar servers that require human input of mathematics.",
  paper = "Fate99c.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Faxe02,
  author = "Faxen, Karl-Filip",
  title = {{A Static Sematics for Haskell}},
  year = "2002",
  journal = "J. Functional Programming",
  volume = "12",
  number = "4-5",
  pages = "295-357",
  abstract =
    "This paper gives a static semantics for Haskell 98, a non-strict
    purely functional programming language. The semantics formally speci
    es nearly all the details of the Haskell 98 type system, including the
    resolution of overloading, kind inference (including defaulting) and
    polymorphic recursion, the only major omission being a proper
    treatment of ambiguous overloading and its resolution. Overloading is
    translated into explicit dictionary passing, as in all current
    implementations of Haskell. The target language of this translation is
    a variant of the Girard-Reynolds polymorphic lambda calculus featuring
    higher order polymorphism and explicit type abstraction and
    application in the term language. Translated programs can thus still
    be type checked, although the implicit version of this system is
    impredicative. A surprising result of this formalization e ort is that
    the monomorphism restriction, when rendered in a system of inference
    rules, compromises the principal type property.",
  paper = "Faxe02.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Feit63,
  author = "Feit, Walter and Thompson, John G.",
  title = {{Solvability of Groups of Odd Order}},
  journal = "Pacific Journal of Mathematics",
  volume = "13",
  pages = "775-1029",
  year = "1963"
}
```

———-

— **axiom.bib** —

```
@techreport{Fell85,
  author = "Felleisen, Matthias",
```

```
  title = {{Transliterating Prolog into Scheme}},
  type = "technical report",
  number = "182",
  institution = "University of Indiana",
  year = "1985",
  paper = "Fell85.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Fent93,
  author = "Fenton, Norman",
  title = {{How Effective Are Software Engineering Methods}},
  journal = "J. Systems Software",
  volume = "22",
  pages = "141-148",
  year = "1993",
  abstract =
    "For 25 years, software engineers have sought methods which they hope
    can provide a technological fix for the software crisis. Proponents
    of specific methods claim that their use leads to significantly
    improved quality and productivity. Such claims are rarely, if ever,
    backed up by hard evidence. We show that often, where real empirical
    evidence does exist, the results are counter to the views of the
    so-called experts. We examine the impact on the software industry of
    continuing to place our trust in unproven, and often revolutionary,
    methods. The very poor state of the art of empirical assessment in
    software engineering can be explained in part by inappropriate or
    inadequate use of measurement. Numerous empirical studies are flawed
    because of their poor experimental design and lack of adherence to
    proper measurement principles.",
  paper = "Fent93.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Ferg06,
  author = "Ferguson, Craig",
  title = {{Between the Bridge and the River}},
  year = "2006",
  publisher = "Chronicle Books LLC",
  isbn = "978-0-8118-7303-1",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Fetz88,
  author = "Fetzer, James H.",
  title = {{Program Verification: The Very Idea}},
  journal = "Communications of the ACM",
  volume = "31",
  number = "9",
  pages = "1048-1063",
  year = "1988",
  abstract =
    "The notion of program verification appears to trade upon an
    equvocation. Algorithms, as logical structures, are appropriate
    subjects for deductive verification. Programs, as causal models of
    those structures, are not. The success of program verification as a
    generally applicable and completely reliable method for guaranteeing
    program performance is not even a theoretical possibility.",
  paper = "Fetz88.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Feyn02,
  author = "Feynman, Richard and Hey, Anthony J.G.",
  title = {{Feynman and Computation}},
  publisher = "Westview Press",
  year = "2002",
  isbn = "0-8133-4039-X",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Fiek17,
  author = "Fieker, Claus and Hart, William and Hofmann, Tommy and
            Johansson, Fredrik",
  title = {{Nemo/Hecke: Computer Algebra and Number Theory Package
            for the Julia Programming Language}},
  booktitle = "ISSAC'17",
  publisher = "ACM",
  year = "2017",
  pages = "157-164",
  abstract =
    "We introduce two new packages, Nemo and Hecke, written in the
    Julia programming language for computer algebra and number
    theory. We demonstrate that high performance generic algorithms
    can be implemented in Julia, without the need to resort to a
```

```
    low-level C implementation. For specialised algorithms, we use
    Julia's efficient native C interface to wrap existing C/C++
    libraries such as Flint, Arb, Antic and Singular. We give examples
    of how to use Hecke and Nemo and discuss some algorithms that we
    have implemented to provide high performance basic arithmetic.",
  paper = "Fiek17.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Fija17,
  author = "Fijalkow, Nathanael",
  title =
   {{Computing using the generators of a group: the Schreier-Sims' algorithm}},
  year = "2017",
  link = "\url{https://www.cs.ox.ac.uk/blogs/nathanael-fijalkow/2016/01/27/computing-using-the-generators-of-a-g
}
```

———-

— **axiom.bib** —

```
@article{Fitt93,
  author = "Fitt, A.D. and Hoare, G.T.Q",
  title = {{The Closed-Form Integration of Arbitrary Functions}},
  journal = "The Mathematical Gazette",
  volume = "77",
  number = "479",
  pages = "227-236",
  year = "1993",
  paper = "Fitt93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Fitt69,
  author = "Fitting, Melvin",
  title = {{Intuitionistic Logic Model Theory and Forcing}},
  publisher = "North Holland",
  year = "1969",
  paper = "Fitt69.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Floy90,
  author = "Floyd, Robert W. and Knuth, Donald E.",
  title = {{Addition Machines}},
  journal = "SIAM Journal on Computing",
  volume = "19",
  number = "2",
  pages = "329-340",
  year = "1990",
  comment = "Stanford Report STAN-CS-89-1268",
  link =
  "\url{i.stanford.edu/pub/cstr/reports/cs/tr/89/1268/CS-TR-89-1268.pdf}",
  abstract =
    "It is possible to compute gcd(x,y) efficiently with only
    $O(\log xy)$ additions and subtractions, when three arithmetic
    registers are available but not when there are only two. Several
    other functions, such as $x^y \bmod z$, are also efficiently
    computatable in a small number of registers, using only addition,
    subtraction, and comparison.",
  paper = "Floy90.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Fode83,
  author = "Foderaro, John K.",
  title = {{The Design of a Language for Algebraic Computation Systems}},
  school = "U.C. Berkeley, EECS Dept.",
  year = "1983",
  link = "\url{http://digitalassets.lib.berkeley.edu/techreports/ucb/text/CSD-83-160.pdf}",
  abstract =
    "This thesis describes the design of a language to support a
    mathematics-oriented symbolic algebra system. The language, which we
    have named NEWSPEAK, permits the complex interrelations of
    mathematical types, such as rings, fields and polynomials to be
    described. Functions can be written over the most general type that
    has the required operations and properties and the inherited by
    subtypes. All function calls are generic, with most function
    resolution done at compile time. Newspeak is type-safe, yet permits
    runtime creation of tyhpes.",
  paper = "Fode83.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Font16,
```

```
  author = "Font, Josep Marie",
  title = {{Abstract Algebraic Logic: An Introductory Textbook}},
  year = "2016",
  publisher = "College Publications, Kings College",
  isbn = "978-1-84890-207-7"
}
```

———-

— **axiom.bib** —

```
@article{Ford04,
  author = "Ford, Bryan",
  title = {{Parsing Expression Grammars: A Recognition-Based Syntactic
            Foundation}},
  journal = "SIGPLAN Notices",
  volume = "39",
  number = "1",
  pages = "111-122",
  year = "2004",
  abstract =
    "For decades we have been using Chomsky's generative system of
    grammars, particularly context-free grammars (CFGs) and regular
    expressions (REs), to express the syntax of programming languages
    and protocols. The power of generative grammars to express
    ambiguity is crucial to their original purpose of modelling
    natural languages, but this very power makes it unnecessarily
    difficult both to express and to parse machine-oriented languages
    using CFGs. Parsing Expression Grammars (PEGs) provide an
    alternative, recognition-based formal foundation for describing
    machine-oriented syntax, which solves the ambiguity problem by not
    introducing ambiguity in the first place. Where CFGs express
    nondeterministic choice between alternatives, PEGs instead use
    {\sl prioritized choice}. PEGs address frequently felt
    expressiveness limitations of CFGs and REs, simplifying syntax
    definitions and making it unnecessary to separate their lexical
    and hierarchical components. A linear-time parser can be built for
    any PEG, avoiding both the complexity and fickleness of LR parsers
    and the inefficiency of generalized CFG parsing. While PEGs
    provide a rich set of operators for constructing grammars, they
    are reducible to two minimal recognition schemas developed around
    1970, TS/TDPL and gTS/GTDPL, which are here proven equivalent in
    effective recognition power.",
  paper = "Ford04.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Fors11,
```

```
  author = "Forster, Edward Morgan",
  title = {{The Other Side Of The Hedge}},
  year = "1911",
  link = "\url{http://www.101bananas.com/library2/otherside.html}"
}
```

———-

— **axiom.bib** —

```
@misc{Four03,
  author = "Fourer, Robert and Gay, David M. and Kernighan, Brian W.",
  title = {{AMPL Reference}},
  link = "\url{https://ampl.com/BOOK/CHAPTERS/24-refman.pdf}",
  year = "2003",
  paper = "Four03.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Four99,
  author = "Fournie, Michel",
  title = {{High-order compact schemes: Application to bidimensional unsteady
           diffusion-convection problems. II}},
  journal = "C. R. Acad. Sci.",
  volume = "328",
  number = "6",
  pages = "539-542",
  year = "1999",
  abstract =
    "This Note generalizes the construcion and the analysis of high
    order compact difference schemes for unsteady 2D
    diffusion-convection problems cf. Part I by A. Rigal. The accuracy
    (of order 2 in time and 4 in space) and the stability analysis
    yield different choices of the weighting parameters. This work
    realized with symbolic computation systems allow to obtain
    analytical results for a wide class of schemes and to increase the
    stability domain in some cases.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fred16,
  author = "Fredrikson, Matt",
  title = {{Automated Program Verification and Testing}},
  comment = "slides",
```

```
  year = "2016",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@book{Frie96,
  author = "Friedman, Daniel P. and Felleisen, Matthias",
  title = {{The Seasoned Schemer}},
  publisher = "MIT Press",
  year = "1996",
  isbn = "0-262-56100-X",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Frie99,
  author = "Friedman, Daniel P. and Felleisen, Matthias",
  title = {{The Little Schemer}},
  publisher = "MIT Press",
  year = "1999",
  isbn = "0-262-56099-2",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Frie05,
  author = "Friedman, Daniel P. and Byrd, William E. and
            Kiselyov, Oleg",
  title = {{The Reasoned Schemer}},
  publisher = "MIT Press",
  year = "2005",
  isbn = "0-262-56214-6",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Frie15,
  author = "Friedman, Daniel P. and Eastland, Carl",
  title = {{The Little Prover}},
```

```
  publisher = "The MIT Press",
  year = "2015",
  isbn = "978-0-262-52795-8",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Frie18,
  author = "Friedman, Daniel P. and Christiansen, David Thrane",
  title = {{The Little Typer}},
  publisher = "The MIT Press",
  year = "2018",
  isbn = "978-0-262-53643-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Frit93,
  author = "Fritzson, Peter and Engelson, Vadim and Viklund, Lars",
  title = {{Variant Handling, Inheritance and Composition in the
            ObjectMath Computer Algebra Environment}},
  booktitle = "DISCO 1993",
  year = "1993",
  pages = "145-163",
  publisher = "Springer",
  abstract =
    "ObjectMath is a high-level programming environment and modeling
    language for scientific computing which supports variants and
    graphical browsing in the environment and integrates
    object-oriented constructs such as classes and single and multiple
    inheritance within a computer algebra language. In addition,
    composition of objects using the part-of relation and support for
    solution of systems of equations is provided. This environment is
    currently being used for industrial applications in scientific
    computing. The ObjectMath environment is designed to handle
    realistic problems. This is achieved by allowing the user to
    specify transformations and simplifications of formulae in the
    model, in order to arrive at a representation which is efficiently
    solvable. When necessary, equations can be transformed to C++ code
    for efficient numerical solution. The re-use of equations through
    inheritance in general reduces models by a factor of two to three,
    compared to a direct representation in the Mathematica computer
    algebra language. Also, we found that multiple inheritance from
    orthogonal classes facilitates re-use and maintenance of
    application models.",
  paper = "Frit93.pdf",
```

```
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Frue91,
  author = "Fruehwirth, Thom and Shapiro, Ehud and Vardi, Moshe Y. and
            Yardeni, Eyal",
  title = {{Logic programs as types for logic programs}},
  booktitle = "Proc. Sixth Annual IEEE Symp. on Logic in Comp. Sci.",
  publisher = "IEEE",
  pages = "300-309",
  year = "1991",
  abstract =
    "Type checking can be extremely useful to the program development process.
    Of particular interest are descriptive type systems, which let the
    programmer write programs without having to define or mention types.
    We consider here optimistic type systems for logic programs. In such
    systems types are conservative approximations to the success set of the
    program predicates. We propose the use of logic programs to describe
    types. We argue that this approach unifies the denotational and
    operational approaches to descriptive type systems and is simpler
    and more natural than previous approaches. We focus on the use of
    unary-predicate programs to describe types. We identify a proper class
    of unary-predicate programs and show that it is expressive enough to
    express several notions of types. We use an analogy with 2-way automata
    and a correspondence with alternating algorithms to obtain a complexity
    characterization of type inference and type checking. This
    characterization was facilitated by the use of logic programs to
    represent types.",
  paper = "Frue91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Fuhx89,
  author = "Fuh, You-Chin and Mishra, Prateek",
  title = {{Polymorphic Subtype Inference -- Closing the Theory-Practice Gap}},
  journal = "Lecture Notes in Computer Science",
  volume = "352",
  year = "1989",
  pages = "167-183",
  paper = "Fuhx89.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Fuhx90,
  author = "Fuh, You-Chin",
  title = {{Type Inference with Subtypes}},
  journal = "Theoretical Computer Science",
  volume = "73",
  number = "2",
  year = "1990",
  pages = "155-175",
  abstract =
    "We extend polymorphic type inference with a very general notion of
    subtype based on the concept of type transformation. This paper
    describes the following results. We prove the existence of (i)
    principal type property and (ii) syntactic completeness of the
    type-checker, for type inference with subtypes. This result is
    developed with only minimal assumptions on the underlying theory of
    subtypes. As a consequence, it can be used as the basis for type
    inference with a broad class of subtype theories. For a particular
    structural theory of subtypes, those engendered by inclusions
    between type constants only, we show that principal types are
    compactly expressible. This suggests that type inference for the
    structured theory of subtypes is feasible. We describe algorithms
    necessary for such a system. The main algorithm we develop is called
    MATCH, an extension to the classical unification algorithm. A proof of
    correctness for MATCH is given.",
  paper = "Fuhx90.pdf"
}
```

———-

1.2.7 G

— **axiom.bib** —

```
@book{Gabr85,
  author = "Gabriel, Richard",
  title = {{Performance and Evaluation of Lisp Systems}},
  link = "\url{https://www.dreamsongs.com/Files/Timrep.pdf}",
  year = "1985",
  publisher = "MIT Press",
  paper = "Gabr85.pdf"

}
```

———-

— **axiom.bib** —

```
@misc{Gabr91,
  author = "Gabriel, Richard",
```

```
  title = {{Lisp: Good News, Bad News, How to Win Big}},
  year = "1991",
  link = "\url{https://www.dreamsongs.com/WIB.html}"
}
```

———-

— **axiom.bib** —

```
@book{Gabr96,
  author = "Gabriel, Richard",
  title = {{Patterns of Software}},
  link = "\url{https://www.dreamsongs.com/Files/PatternsOfSoftware.pdf}",
  year = "1996",
  publisher = "Oxford University Press",
  paper = "Gabr96.pdf"

}
```

———-

— **axiom.bib** —

```
@misc{Gabr10,
  author = "Gabriel, Richard P.",
  title = {{A Reivew of The Art of the Metaobject Protocol}},
  year = "2010",
  paper = "Gabr10.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Gall88,
  author = "Galligo, Andre and Pottier, Loic and Traverso, Carlo",
  title = {{Greater Easy Common Divisor and Standard Basis Completion
            Algorithms}},
  journal = "Lecture Notes in Computer Science",
  volume = "358",
  pages = "162-176",
  year = "1988",
  abstract =
    "The computation of a standard basis (also alled Groebner basis)
    of a multivariate polynomial ideal over a field $K$ is crucial in
    many appliations. The problem is intransically time and space
    consuming, and many researchers aim to improve the basic algorithm
    due to B. Buchberger [Bu1]. One has investigated the problem of
    the optimal choice of the term ordering depending from the use
    that has to be made of the results, [BS], and the systematic
```

```
    elimination of unnecessary reductions [Bu2],[GM],[Po]. We can call
    all these problems ‘‘combinatorial complexity problems’’.

    The present paper considers arithmetic complexity problems; our
    main problem is how to limit the growth of the coefficients in the
    algorithms and the complexity of the field operations
    involved. The problem is important with every ground field, with
    the obvious exception of finite fields.

    The ground field is often represented as the field of fractions of
    some explict domain, which is usually a quotient of a finite
    extension of $Z$, and the computations are hence reduced to these
    domains.

    The problem of coefficient growth, and the complexity already
    appearing in the calculation of the GCD of tw univariate
    polynomials, which is indeed a very special case of standard basis
    computation; the PRS algorithms of Brown and Collins operate the
    partial coefficient simplifications predicted by a theorem, hence
    succeeding in controlling this complexity.

    Our approach looks for analogies with these algorithms, but a
    general structure theorem is missing, hence our approach relies on
    a limited seqrch of coefficient simplifications. The basic idea is
    the following: since the GCD is usually costly, we can use in its
    place the ‘‘greatest between the common divisors that are easy to
    compute’’ (the GECD), this suggestion allowing different
    instances. A set of such instances is based on the remark that if
    you hve elements in factorized form, then many common divisors are
    immediately evident. Since irreducible factorization, even
    assuming that is exists in our domain, is costly, we use a partial
    factorization basically obtained using a ‘‘lazy multiplication’’
    technique, i.e. performing coefficient multiplications only if
    they are unavoidable. The resulting algorithms were tested with a
    ‘‘simulatied’’ implementation on the integers, and the results
    suggest that a complete implementation should be very efficient,
    at least when the coefficient domain is a multivariate rational
    function field.",
  paper = "Gall88.pdf",
  keywords = "GCD"
}
```

———-

— axiom.bib —

```
@phdthesis{Gane09,
  author = "Ganesalingam, Mohan",
  title = {{The Language of Mathematics}},
  school = "University of Cambridge",
  year = "2009"
}
```

———-

— **axiom.bib** —

```
@article{Gane17,
  author = "Ganesalingam, M. and Gowers, W.T.",
  title = {{A Fully Automatic Theorem Prover with Human-Style Output}},
  journal = "J. Automated Reasoning",
  volume = "58",
  pages = "253-291",
  year = "2017",
  abstract =
    "This paper describes a program that solves elementary
    mathematical problems, mostly in metric space theory, and presents
    solutions that are hard to distinguish from solutions that might
    be written by a human mathematician.",
  paper = "Gane17.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Ganz80,
  author = "Ganzinger, Harald",
  title = {{Transforming Denotational Semantics into Practical
            Attribute Grammars}},
  journal = "LNCS",
  volume = "54",
  pages = "1-69",
  year = "1980",
  paper = "Ganz80.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Gapt19,
  author = "Unknown",
  title = {{GAPT: General Architecture for Proof Theory}},
  year = "2019",
  link = "\url{https://www.logic.at/gapt/downloads/gapt-user-manual.pdf}",
  paper = "Gapt19.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{GAPx17,
```

```
  author = "The GAP Group",
  title = {{GAP - Reference Manual}},
  year = "2017",
  link = "\url{https://www.gap-system.org/Manuals/doc/ref/manual.pdf}"
}
```

———-

— **axiom.bib** —

```
@misc{Garc18,
  author = "Garcia, Ronald",
  title = {{Gradual Typing}},
  year = "2018",
  link = "\url{https://www.youtube.com/watch?v=fQRRxaWsuxI}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Garr14,
  author = "Garret, Ron",
  title = {{The Awesome Power of Theory}},
  link = "\url{http://www.flownet.com/ron/lambda-calculus.html}",
  year = "2014"
}
```

———-

— **axiom.bib** —

```
@article{Gaud80,
  author = "Gaudel, M.C.",
  title = {{Specification of Compilers as Abstract Data Type
            Representations}},
  journal = "LNCS",
  volume = "54",
  pages = "140-164",
  year = "1980",
  abstract =
    "This paper presents a method for specifying and proving
    compilers. This method is based on the algebraic data types
    ideas. The main points are:
    \begin{itemize}
    \item to each language is associated an algebraic abstract data type
    \item the semantic value of a program is given as a term of this
    data type
    \item the translation of the semantic values of source programs
    into semantic values of target programs is specified and proved as
```

```
    the representation of an algebrayc data type by another one.
    \end{itemize}
    A compiler generator, PERLUETTE, which accepts such specifications
    as input is described. The proof technic is discussed.",
  paper = "Gaud80.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Gaut19,
  author = "Gauthier, Thibault",
  title = {{Deep Reinforcement Learning in HOL4}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1910.11797.pdf}",
  abstract =
    "The paper describes an implementation of deep reinforcement
    learning through self-supervised learning within the proof
    assistant HOL4. A close interaction between the machine learning
    modules and the HOL4 library is achieved by the choice of tree
    neural networks (TNNs) as machine learning models and the internal
    use of HOL4 terms to represent tree structures of TNNs. Recursive
    improvement is possible when a given task is expressed as a search
    problem. In this case, a Monte Carlo Tree Search (MCTS) algorithm
    guided by a TNN can be used to explore the search space and
    produce better examples for training the next TNN. As an
    illustration, tasks over propositional and arithmetical terms,
    representative of fundamental theorem proving techniques, are
    specified and learned: truth estimation, end-to-end computation,
    term rewriting and term synthesis.",
  paper = "Gaut19.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gedd88,
  author = "Geddes, Keith O. and Gonnet, Gaston H. and Smedley, Trevor J.",
  title = {{Heuristic Methods for Operations With Algebraic Numbers}},
  journal = "Lecture Notes in Computer Science",
  volume = "358",
  pages = "475-480",
  year = "1988",
  abstract =
    "Algorithms for doing computations involving algebraic numbers
    have been known for quite some time [6,9,12] and implementations
    now exist in many computer algebra systems [1,4,11]. Many of these
    algorithms have been analysed and shown to run in polynomial time
    and space [7,8], but in spite of this many real problems take
    large amounts of time and space to solve. In this paper we
```

```
    describe a heuristic method which can be used for many operations
    involving algebraic numbers. We gie specifics for doing algebraic
    number inverses, and algebraic number polynomial exact division
    and greatest common divisor calculation. The heurist will not
    solve all instances of these problems, but it returns either the
    correct result or with failure very quickly, and succeeds for a
    very large number of problems. The heuristic method is similar to,
    and based on concepts in [3].",
  paper = "Gedd88.pdf",
  keywords = "GCD"
}
```

———-

— **axiom.bib** —

```
@misc{Gent35a,
  author = "Gentzen, G.",
  title = {{Untersuchungen uber das logische Schliessen I and II}},
  booktitle = "Mathematische Zeitschrift",
  year = "1935",
  publisher = "Springer",
  paper = "Gent35a.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Gerd13,
  author = "Gerdt, Vladimir P. and Koepf, Wolfram and Mayr, Ernst W.
            and Vorozhtsov, Evgenii V.",
  title = {{Computer Algebra in Scientific Computing}},
  publisher = "Springer",
  year = "2013",
  comment = "LNCS 8136",
  paper = "Gerd13.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Geuv00,
  author = "Geuvers, Herman",
  title = {{Induction Is Not Derivable in Second Order Dependent Type Theory}},
  journal = "LNCS",
  volume = "2044",
  year = "2000",
  pages = "166-181",
```

```
  abstract =
    "This paper proves the non-derivability of induction in second
    order dependent type theory ($\lambda{}P2$). This is done by
    providing a model construction for $\lambda{}P2$, based on a
    saturated sets like interpretation of types as sets of terms of a
    weakly extensional combinatory algebra. We give countermodels in
    which the induction principle over natural numbers is not
    valid. The proof does not depend on the specific encoding for
    natural numbers that has been chosen (like e.g. polymorphic Church
    numerals), so in fact we prove that there can not be an encoding
    of natural numbers in $\lambda{}P2$ such that the induction
    principle is satisfied. The method extends immediately to other
    data types, like booleans, lists, trees, etc.

    In the process of the proof we establish some general properties
    of the models, which we think are of independent
    interest. Moreover, we show that the Axiom of Choice is not
    derivable in $\lambda{}P2$.",
  paper = "Geuv00.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Gian85a,
  author = "Gianni, Patrizia and Trager, Barry",
  title = {{GCD's and Factoring Multivariate Polynomials using
           Grobner Bases}},
  journal = "LNCS",
  volume = "204",
  pages = "409-410",
  year = "1985",
  booktitle = "EUROCAL '85 European Conference on Computer Algebra",
  publisher = "Springer",
  abstract =
    "This paper shows how Grobner basis computations can be used to
    compute multivariate gcds, perform Hensel lifting, and reduce
    multivariate factorization to univariate. The essential idea is to
    produce an ideal containing the desired polynomial as an element
    of least degree. The construction is somewhat analogous to the
    recent lattice algorithms for polynomial factorization. A major
    difference is that we don't need to perform the hensel lifting as
    a separate step; one Brobner basis computation is sufficient to
    find the desired multivariate factor or gcd. We produce algorithms
    which are vary simple and may be of use on small systems where
    code size is critical. We feel that these results demonstrate the
    fundamental importance of the Grobner basis in computer algebra.",
  paper = "Gian85a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gian88a,
  author = "Gianni, Patrizia and Miller, Victor and Trager, Barry",
  title = {{Decomposition of Algebras}},
  journal = "Lecture Notes in Computer Science",
  volume = "358",
  pages = "300-308",
  year = "1988",
  abstract =
    "In this paper we deal with the problem of decomposing finite
    commative $Q$-algebras as a direct product of local
    $Q$-algebras. We solve this problem by reducing it to the problem
    of finding a decomposition of finite algebras over finite
    field. We will show that it is possible to define a lifting
    process that allows to reconstruct the answer over the rational
    numbers. This lifting appears to be very efficient since it is a
    quadratic lifting that doesn't require stepwise inversions. It is
    easy to see that the Berlekamp-Hensel algorithm for the
    factorization of polynomials is a special case of this argument.",
  paper = "Gian88a.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Gian85,
  author = "Giannini, Paola",
  title = {{Type Checking and Type Deduction Techniques for
            Polymorphic Programming Languages}},
  type = "technical report",
  institution = "Carnegie Mellon University",
  number = "CMU-CS-85-187",
  year = "1985",
  abstract =
    "In this paper we present some of the syntactic issues that arise
    in polymorphic programming languages. In particular we examine
    type checking and deduction in two different polymorphic type
    strucutres: the parametric lambda-calculus (with let construct)
    and the polymorphic or second-order lambda-calculus. In both
    approaches the behavior of types is formalized with type inference
    rules. Examples of programming languages following those
    approaches are presented and some of their specific problems
    studied.",
  paper = "Gian85.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Gilb19,
  author = "Gilbert, Gaetan",
  title = {{A Type Theory with Definitional Proof-Irrelevance}},
  school = "Ecole Nationale Superieure Mines-Telecom Alantique.",
  year = "2019",
  link = "\url{https://gitlab.com/SkySkimmer/thesis/~/jobs/artifacts/master/download?job=build}",
  paper = "Gilb19.zip"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Gira72,
  author = "Girard, Jean-Yves",
  title = {Intrpr\'etation fontionelle et \'elimination des coupures de
           l'arithm\'etique d'orde sup\'erieur},
  school = {Universit\'e Paris VII},
  year = "1972"
}
```

———-

— **axiom.bib** —

```
@article{Gira87,
  author = "Girard, Jean-Yves",
  title = {{Linear Logic}},
  journal = "Theoretical Computer Science",
  volume = "50",
  year = "1987",
  pages = "1-102",
  abstract =
    "The familiar connective of negation is broken into two
    operations: linear negation which is the purely negative part of
    negation and the modality ``of course'' which has the meaning of a
    reaffirmation. Following this basic discovery, a completely new
    approach to the whole area between constructive logics and
    programmation is initiated.",
  paper = "Gira87.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Gira89,
  author = "Girard, Jean-Yves",
  title = {{Proofs and Types}},
```

```
  publisher = "Cambridge University Press",
  year = "1989",
  paper = "Gira89.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Gira95,
  author = "Girard, Jean-Yves",
  title = {{Linear Logic: Its Syntax and Semantics}},
  publisher = "Cambridge University Press",
  year = "1995",
  link = "\url{http://girard.perso.math.cnrs.fr/Synsem.pdf}",
  isbn = "9780521559614",
  paper = "Gira95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Gira95a,
  author = "Girard, Jean-Yves and Lafont, Yves and Regnier, Laurent",
  title = {{Advances in Linear Logic}},
  publisher = "Cambridge University Press",
  year = "1995",
  isbn = "0-521-55961-8",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Glas94,
  author = "Glass, Robert L.",
  title = {{The Software-Research Crisis}},
  journal = "IEEE Software",
  volume = "11",
  number = "6",
  year = "1994",
  abstract =
    "With the advantage of more than 25 years' hindsight, this twenty-first
    century author looks askance at the 'crisis' in software practice and
    expresses deep concern for a crisis in software research.",
  paper = "Glas94.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Glas02,
  author = "Glass, Robert L.",
  title = {{The Proof of Correctness Wars}},
  journal = "Communications of the ACM",
  volume = "45",
  number = "8",
  year = "2002",
  pages = "19-21",
  paper = "Glas02.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Glei05,
  author = "Gleich, David",
  title = {{Finite Calculus: A Tutorial for Solving Nasty Sums}},
  year = "2005",
  abstract =
    "In this tutorial, I will first explain the need for finite
    calculus using an example sum I think is difficult to solve. Next,
    I will show where this sum actually occurs and why it is
    important. Following that, I will present all the mathematics
    behind finite calculus and a series of theorems to make it helpful
    before concluding with a set of examples to show that it really is
    useful.",
  paper = "Glei05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Gode58,
  author = "Godel, Kurt",
  title = {\"Uber eine bisher noch nicht benutzte Erweiterung des Finiten
           Standpunktes},
  journal = "Dialectica 12",
  year = "1958",
  pages = "280-287"
}
```

———-

— **axiom.bib** —

```
@misc{Goel19,
  author = "Goel, Shilpi and Slobodova, Anna and Sumners, Rob and
            Swords, Sol",
  title = {{Verifying X86 Instruction Implementations}},
  year = "2019",
  link = "\url{https://arxiv.org/abs/1912.10285}",
  paper = "Goel19.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@techreport{Gogu76,
  author = "Goguen, J.A. and Thatcher, J.W. and Wagner, E.G. and
            Wright, J.B.",
  title = {{An Initial Algebra Approach to the Specification,
            Correctness and Implementation of Abstract Data Types}},
  type = "Research Report",
  institution = "IBM Research",
  number = "RC-6487",
  year = "1976"
}
```

———-

— **axiom.bib** —

```
@techreport{Gogu89,
  author = "Goguen, Joseph and Meseguer, Jose",
  title = {{Order-sorted Algebra I : Equational Deduction for Multiple
           Inheritance, Overloading, Exceptions, and Partial Operations}},
  type = "technical report",
  institution = "SRI International",
  year = "1989",
  number = "SRIR 89-10"
}
```

———-

— **axiom.bib** —

```
@article{Gogu92,
  author = "Goguen, Joseph and Meseguer, Jose",
  title = {{Order-sorted Algebra I : Equational Deduction for Multiple
           Inheritance, Overloading, Exceptions, and Partial Operations}},
  journal = "Theoretical Computer Science",
  volume = "105",
```

```
  number = "2",
  year = "1992",
  pages = "217-273",
  abstract =
    "This paper generalizes many-sorted algebra (MSA) to order-sorted
    algebra (OSA) by allowing a partial ordering relation on the set of
    sorts. This supports abstract data types with multiple inheritance (in
    roughly the sense of object-oriented programming), several forms of
    polymorphism and overloading, partial operations (as total on
    equationally defined subsorts), exception handling, and an operational
    semantics based on term rewriting. We give the basic algebraic
    constructions for OSA, including quotient, image, product and term
    algebra, and we prove their basic properties, including quotient,
    homomorphism, and initiality theorems. The paper's major mathematical
    results include a notion of OSA deduction, a completeness theorem for
    it, and an OSA Birkhoff variety theorem. We also develop conditional
    OSA, including initiality, completeness, and McKinsey-Malcev
    quasivariety theorems, and we reduce OSA to (conditional) MSA, which
    allows lifting many known MSA results to OSA. Retracts, which
    intuitively are left inverses to subsort inclusions, provide
    relatively inexpensive run-time error handling. We show that it is
    safe to add retracts to any OSA signature, in the sense that it gives
    rise to a conservative extension. A final section compares and
    contrasts many different approaches to OSA. This paper also includes
    several examples demonstrating the flexibility and applicability of
    OSA, including some standard benchmarks like stack and list, as well
    as a much more substantial example, the number hierarchy from the
    naturals up to the quaternions.",
  paper = "Gogu92.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Gogu96,
  author = "Goguen, Joseph A. and Malolm, Grant",
  title = {{Algebraic Semantics of Imperative Programs}},
  publisher = "MIT Press",
  year = "1996",
  isbn = "0-262-07172-X",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Gold83,
  author = "Goldberg, Adele and Robson, David",
  title = {{Smalltalk-80: The Language and Its Implementation}},
  publisher = "Addison-Wesley",
```

```
  year = "1983"
}
```

———-

— **axiom.bib** —

```
@book{Gold84,
  author = "Goldblatt, Robert",
  title = {{Topoi: The Categorical Analysis of Logic}},
  year = "1984",
  publisher = "Dover",
  isbn = "978-0-486-45026-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Gold05,
  author = "Goldstein, Rebecca",
  title = {{Incompleteness}},
  publisher = "Atlas Books",
  year = "2005",
  isbn = "0-393-32760-4",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Goll90,
  author = "Gollan, H. and Grabmeier, J.",
  title = {{Algorithms in Representation Theory and their
           Realization in the Computer Algebra System Scratchpad}},
  journal = "Bayreuther Mathematische Schriften",
  volume = "33",
  year = "1990",
  pages = "1-23",
  algebra = "\newline\refto{package REP1 RepresentationPackage1}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Gons71,
  author = "Gonshor, H.",
```

```
  title = {{Contributions to Genetic Algebras}},
  journal = "Proc. Edinburgh Mathmatical Society (Series 2)",
  volume = "17",
  number = "4",
  month = "December",
  year = "1971",
  issn = "1464-3839",
  pages = "289--298",
  doi = "10.1017/S0013091500009548",
  link = "\url{http://journals.cambridge.org/article_S0013091500009548}",
  algebra = "\newline\refto{domain ALGSC AlgebraGivenByStructuralConstants}",
  abstract =
    "Etherington introduced certain algebraic methods into the study of
    population genetics. It was noted that algebras arising in genetic
    systems tend to have certain abstract properties and that these can be
    used to give elegant proofs of some classical stability theorems in
    population genetics."
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gont07,
  author = "Gonthier, Goerges and Mahboubi, Assia and Rideau, Laurence
            and Tassi, Enrico and Thery, Laurent",
  title = {{A Modular Formalisation of Finite Group Theory}},
  booktitle = "Theorem Proving in Higher Order Logics",
  publisher = "Springer",
  pages = "86-101",
  year = "2007"
}
```

———-

— **axiom.bib** —

```
@article{Gont08,
  author = "Gonthier, Goerges",
  title = {{Formal Proof -- The Four Color Theorem}},
  journal = "Notices Amer. Math. Soc.",
  volume = "55",
  number = "11",
  pages = "1382-1393",
  year = "2008"
}
```

———-

— **axiom.bib** —

```
@article{Gont10,
  author = "Gonthier, Goerges and Mahboubi, Assia",
  title = {{An Introduction to Small Scale Reflection in Coq}},
  journal = "J. Formaliz. Reason.",
  volume = "3",
  number = "2",
  pages = "95-152",
  year = "2010"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gont11,
  author = "Gonthier, Goerges",
  title = {{Advances in the Formalization of the Odd Order Theorem}},
  booktitle = "Interactive Theorem Proving",
  publisher = "Springer",
  pages = "2",
  year = "2011"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gont11a,
  author = "Gonthier, Goerges",
  title = {{Point-Free, Set-Free Concrete Linear Algebra}},
  booktitle = "Interactive Theorem Proving",
  publisher = "Springer",
  pages = "103-118",
  year = "2011"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gont11b,
  author = "Gonthier, Goerges and Ziliani, Beta and Nanevski, Aleksandar
            and Dreyer, Derek",
  title = {{How to make Ad Hoc Proof Automation less Ad Hoc}},
  booktitle = "Int. Conf. on Functional Programming",
  publisher = "ACM",
  pages = "163-175",
  year = "2011"
}
```

———-

— **axiom.bib** —

```
@misc{Gonz18,
  author = "Gonzalez, Gabriel",
  title = {{How to Prove Large Software Projects Correct}},
  year = "2018",
  link = "\url{https://www.youtube.com/watch?v=moAfgDFVLUs}"
}
```

———-

— **axiom.bib** —

```
@book{Gord79f,
  author = "Gordon, Michael J. and Milner, Arthur J. and
            Wadsworth, Christopher P.",
  title = {{Edinburgh LCF}},
  publisher = "Springer-Verlag",
  year = "1979",
  isbn = "3-540-09724-4",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@incollection{Gord79a,
  author = "Gordon, Michael J. and Milner, Arthur J. and
            Wadsworth, Christopher P.",
  title = {{Front Matter}},
  booktitle = "Edinburgh LCF",
  publisher = "Springer-Verlag",
  year = "1979",
  isbn = "978-3540097242",
  pages = "1-8",
  paper = "Gord79a.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Gord79b,
  author = "Gordon, Michael J. and Milner, Arthur J. and
            Wadsworth, Christopher P.",
  title = {{Introduction}},
  booktitle = "Edinburgh LCF",
  publisher = "Springer-Verlag",
  year = "1979",
  isbn = "978-3540097242",
```

```
  pages = "1-12",
  paper = "Gord79b.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Gord79c,
  author = "Gordon, Michael J. and Milner, Arthur J. and
            Wadsworth, Christopher P.",
  title = {{ML}},
  booktitle = "Edinburgh LCF",
  publisher = "Springer-Verlag",
  year = "1979",
  isbn = "978-3540097242",
  pages = "13-61",
  paper = "Gord79c.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Gord79d,
  author = "Gordon, Michael J. and Milner, Arthur J. and
            Wadsworth, Christopher P.",
  title = {{PPLAMBDA}},
  booktitle = "Edinburgh LCF",
  publisher = "Springer-Verlag",
  year = "1979",
  isbn = "978-3540097242",
  pages = "62-86",
  paper = "Gord79d.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Gord79e,
  author = "Gordon, Michael J. and Milner, Arthur J. and
            Wadsworth, Christopher P.",
  title = {{APPENDIX}},
  booktitle = "Edinburgh LCF",
  publisher = "Springer-Verlag",
  year = "1979",
  isbn = "978-3540097242",
  pages = "87-159",
  paper = "Gord79e.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gord15,
  author = "Gordon, M.J.C",
  title = {{Tactics for Mechanized Reasoning: A Commentary on Milner
            (1984) 'The Use of Machines to Assist in Rigorous Proof'}},
  journal = "Philosophical Transactions: Mathematical, Physical and
             Engineering Sciences",
  volume = "373",
  number = "2039",
  pages = "1-11",
  year = "2015",
  abstract =
    "Robin Milner's paper, ``The use of machines to assist in rigorous
    proof'', introduces methods for automating mathematical reasoning
    that are a milestone in the development of computer-assisted
    theorem proving. His ideas, particularly his theory of tactics,
    revolutionized the architecture of proof assistants. His
    methodology for automating rigorous proof soundly, particularly
    his theory of type polymorphism in programming, led to major
    contributions to the theory and design of programming
    languages. His citation for the 1991 ACM A.M. Turing award, the
    most prestigious award in computer science, credits him with,
    among other achievements, 'probably the first theoretically based
    yet practical tool for machine assisted proof construction'. This
    commentary was written to celebrate the 350th anniversary of the
    journal {\sl Philosophical Transactions of the Royal Society}.",
  paper = "Gord15.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Goto19,
  author = "Goto, Kazushige and van de Geijn, Robert A.",
  title = {{Anatomy of High-Performance Matrix Multiplication}},
  journal = "Transactions on Mathematical Software",
  volume = "V",
  number = "N",
  year = "2019",
  link = "\url{https://www.cs.utexas.edu/~flame/pubs/GotoTOMS_revision.pdf}",
  abstract =
    "We present the basic principles which underlie the high
    performance implementation of the matrix multiplication that is
    part of the widely used GotoBLAS library. Design decisions are
    justified by successively refining a model of architectures with
    multilevel memories. A simple but effective algorithm for
```

```
    executing this operation results. Implementations on a broad
    selection of architectures are shown to achieve near-peak
    performacne.",
  paper = "Goto19.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Gowe17,
  author = "Gowers, Timothy",
  title = {{Group actions II: the orbit-stabilizer theorem}},
  year = "2017",
  link = "\url{https://gowers.wordpress.com/2011/11/09/group-actions-ii-the-orbit-stabilizer-theorem/}"
}
```

———-

— **axiom.bib** —

```
@article{Grab87,
  author = "Grabmeier, Johannes and Kerber, Adalbert",
  title = {{The Evaluation of Irreducible Polynomial
           Representations of the General Linear Groups
           and of the Unitary Groups over Fields of
           Characteristic 0}},
  journal = "Acta Applicandae Mathematica",
  volume = "8",
  year = "1987",
  pages = "271-291",
  algebra = "\newline\refto{package REP1 RepresentationPackage1}",
  abstract =
    "We describe an efficient method for the computer evaluation of the
    ordinary irreducible polynomial representations of general linear
    groups using an integral form of the ordinary irreducible
    representations of symmetric groups. In order to do this, we first
    give an algebraic explanation of D. E. Littlewood's modification of
    I. Schur's construction. Then we derive a formula for the entries of
    the representing matrices which is much more concise and adapted to
    the effective use of computer calculations. Finally, we describe how
    one obtains  using this time an orthogonal form of the ordinary
    irreducible representations of symmetric groups  a version which
    yields a unitary representation when it is restricted to the unitary
    subgroup. In this way we adapt D. B. Hunter's results which heavily
    rely on Littlewood's methods, and boson polynomials come into the play
    so that we also meet the needs of applications to physics.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Grab06a,
  author = "Grabmuller, Martin",
  title = {{Algorithm W Step by Step}},
  year = "2006",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.7733&rep=rep1&type=pdf}",
  abstract =
    "In this paper we develop a complete implementation of the classic
    algoirhtm W for Hinley-Milner polymorphic type inference in Haskell",
  paper = "Grab06a.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Grab10,
  author = "Grabowski, Adam and Kornilowicz, Artur and Naumowicz, Adam",
  title = {{Mizar in a Nutshell}},
  journal = "J. Formaliz. Reason.",
  volume = "3",
  number = "2",
  pages = "153-245",
  year = "2010"
}
```

———-

— **axiom.bib** —

```
@book{Grae79,
  author = "Graetzer, George",
  title = {{Universal Algebra}},
  publisher = "Springer",
  isbn = "978-0-387-77486-2",
  year = "1979",
  paper = "Grae79.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Grat20,
  author = "Gratzer, Daniel and Kavvos, G.A. and Nuyts, Andreas
            and Birkedal, Lars",
  title = {{Multimodel Dependent Type Theory}},
  year = "2020",
  link = "\url{https://arxiv.org/pdf/2011.15021.pdf}",
```

```
  abstract =
    "We introduce MTT, a dependent type theory which supports multiple
    modalities. MTT is parameterized by a mode theory which specifies
    a collection of modes, modalities, and transformations between
    them. We show that different choices of mode theory allow us to
    use the same type theory to compute and reason in many modal
    situations, including guarded recursion, axiomatic cohesion, and
    parametric quantification. We reproduce examples from prior work
    in guarded recursion and axiomatic cohesion -- demonstrating that
    MTT constitutes a simple and usable syntax whose instantiations
    intuitively correspond to previous handcrafted modal type
    theories. In some cases, instantiating MTT to a particular
    situation unearths a previously unknown type theory that improves
    upon prior systems. Finally, we investigate the metatheory of
    MTT. We prove the consistency of MTT and establish canonicity
    through an extension of recent type-theoretic gluing
    techniques. These results hold irrespective of the choice of mode
    theory, and thus apply to a wide variety of modal situations.",
  paper = "Grat20.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Grav18,
  author = "Gravel, Katherine and Jananthan, Hayden and Kepner, Jeremy",
  title = {{Visually Representing the Landscape of Mathematical Structures}},
  link = "\url{https://arxiv.org/abs/1809.05930}",
  year = "2018",
  abstract =
    "The information technology explosion has dramatically increased
    the application of new mathematical ideas and has led to an
    increasing use of mathematics across a wide range of fields that
    have been traditionally labeled 'pure' or 'theoretical'. There is
    a critical need for tools to make these concepts readily
    accessible to a broader community. This paper details the creation
    of visual representations of mathematical structures commonly
    found in pure mathematics to enable both students and
    professionals to rapidly understand the relationships among and
    between mathematical structures. Ten broad mathematical structures
    were identified and used to make eleven maps, with 187 unique
    structures in total. The paper presents a method and
    implementation for categorizing mathematical structures and for
    drawing relationships between mathematical structures that
    provides insight for a wide audience. The most in dept map is
    available online for public use.",
  paper = "Grav18.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Gray18,
  author = "Grayson, Daniel R.",
  title = {{An Introduction to Univalent Foundations for Mathematicians}},
  link = "\url{http://arxiv.org/pdfs/1711.01477v3}",
  year = "2018",
  abstract =
    "We offer an introduction for mathematicians to the univalent
    foundations of Vladimir Voevodsky, aiming to explain how he chose to
    encode mathematics in type theory and how the encoding reveals a
    potentially viable foundation for all of modern mathematics that can
    serve as an alternative to set theory",
  paper = "Gray18.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Gray18a,
  author = "Grayson, Daniel R.",
  title = {{The Mathematical Work of Vladimir Voevodsky}},
  year = "2018",
  link = "\url{https://www.youtube.com/watch?v=BanMgvdKP8E}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gree19,
  author = "Greenberg, Michael",
  title = {{The Dynamic Practice and Static Theory of Gradual
            Typing}},
  booktitle = "3rd Summit on Advances in Programming Languages",
  publisher = "Dagstuhl Publishing",
  pages = "6:1-6:20",
  year = "2019",
  abstract =
    "We can tease apart the research on gradual types into two
    'lineages': a pragmatic, implementation-oriented, dynamic-first
    lineage and a formal, type-theoretic, static-first lineage. The
    dynamic-first lineage's focus is on taming particular idioms --
    'pre-existing conditions' in untyped programming languages. The
    static-first lineage's focus is on interoperation and individual
    type system features, rather than the collection of features found
    in any particular language. Both appear in programming languages
    research under the name ``gradual typing'', and they are in active
    conversation with each other.
```

```
    What are these two lineages? What challenges and opportunities
    await the static-first lineage? What progress has been made so
    far?",
  paper = "Gree19.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Grei85,
  author = "Greif, J.M.",
  title = {{The SMP Pattern Matcher}},
  booktitle = "Research Contributions from the Euro. Conf. on Comp. Alg.",
  series = "Lecture Notes in Computer Science Volume 204",
  volume = "2",
  pages = "303-314",
  year = "1985",
  isbn = "0-387-15983-5 (vol. 1),0-387-15984-3 (vol. 2)",
  abstract =
    "A new pattern matcher has been implemented for the symbolic
    manipulation program SMP. The pattern matcher correctly treats
    associative and commutative functions, functions with arbitrary
    symmetries under permutations of their arguments, and patterns
    subject to arbitrary logical predicates. A pattern is said to match
    a candidate instance if the set of expressions it represents is a
    superset of those represented by the instance (either or both may
    contain generic symbols which match possibly infinite sets of
    expressions). The matcher is primarily syntactic, but is able to
    find matches for syntactically different expressions which become
    literally equivalent upon replacing the generic symbols in the
    pattern by their bindings to subexpressions of the instance and
    simplifying the result. The bindings must be determined by literal
    comparison with the instance, not by solving equations.

    This paper discusses issues arising during the construction of
    this pattern matcher, and gives examples of using pattern matching
    for extending the power of built-in operations and adding new
    mathematical knowledge to a symbol manipulation program.",
  paper = "Grei85.pdf",
  keywords = "axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Grie78,
  author = "Gries, David",
  title = {{Programming Methodology}},
  publisher = "Springer-Verlag",
```

```
  year = "1978"
}
```

———-

— **axiom.bib** —

```
@article{Grie76,
  author = "Griesmer, James",
  title = {{Symbolic Mathematical Computation: A Survey}},
  journal = "SIGSAM Bulletin",
  volume = "10",
  number = "2",
  pages = "30-32",
  year = "1976",
  paper = "Grie76.pdf",
  keywords = "printed, axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Grie78a,
  author = "Griesmer, J.H. and Jenks, R.D. and Yun, D.Y.Y",
  title = {{A Taxonomy for Algebraic Computation}},
  journal = "ACM SIGSAM Bulletin",
  volume = "12",
  number = "3",
  pages = "25-28",
  year = "1978",
  abstract =
    "Recently the authors responded to a request from Professor
    Anthony Ralston of the State University of New York at Buffalo to
    participate in the review of a proposed Taxonomy of Computer
    Science and Engineering.",
  paper = "Grie78a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Griv93,
  author = "Grivas, Georgios and Maeder, Roman E.",
  title = {{Matching and Unification for the Object-Oriented Symbolic
            Computation System}},
  booktitle = "DISCO 1993",
  year = "1993",
  pages = "164-176",
  publisher = "Springer",
```

```
  abstract =
    "Term matching has become one of the most important privmitive
    operations for symbolic computation. This paper describes the
    extension of the object-oriented symbolic computation system
    AlgBench with pattern matching and unification facilities. The
    various pattern objects are organized in subclasses of the class
    of the composite expressions. This leads to a clear design and a
    distributed implementation of the pattern matcher in the
    subclasses. New pattern object classes can consequently be added
    easily to the system. Huet's and our simple mark and retract
    algorithm for standard unification as well as Stickel's algorithm
    for associative commutative unification have been implemented in
    an object-oriented style. Unifiers are selected at runtime. We
    extend Mathematica's type-constrained pattern matching by taking
    into account inheritance information from a user-defined hierarchy
    of object types. the argument unification is basically instance
    variable unification. The improvement of the pattern matching
    operation of a rule- and object-based symbolic computation system
    with unification in an object-oriented way seems to be very
    appropriate.",
  paper = "Griv93.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gros02,
  author = "Grossman, Dan",
  title = {{Existential Types for Imperative Languages}},
  booktitle = "Euro. Symp. on Prog. Langs. and Systems",
  publisher = "Springer-Verlag",
  pages = "21-35",
  year = "2002",
  isbn = "3-540-43363-5",
  abstract =
    "We integrate existential types into a strongly typed C-like
    language. In particular, we show how a bad combination of
    existential types, mutation, and aliasing can cause a subtle
    violation of type safety. We explort two independent ways to
    strengthen the type system to restore safety. One restricts the
    mutation of existential packages. The other restricts the types of
    aliases of extential packages. We use our framework to explain why
    other languages with existential types are safe.",
  paper = "Gros02.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gros06,
  author = "Grossman, Dan",
  title = {{Quantified Types for Imperative Languages}},
  journal = "Trans. on Prog. Lang. and Systems",
  volume = "28",
  number = "3",
  year = "2006",
  pages = "429-475",
  abstract =
    "We describe universal types, existential types, and type
    constructors in Cyclone, a strongly-typed C-like language. We show
    how the language naturally supports first-class polymorphism and
    polymorphic recursion while requiring an acceptable amount of
    explicit type information. More importantly, we consider the
    soundness of type variables in the presence of C-style mutation
    and the address-of operator. For polymorphic references, we
    describe a solution more natural for the C level than the ML-style
    ``value restriction''. For existential types, we discover and
    subsequently avoid a subtle unsoundness issue resulting from the
    address-of operator. We develop a formal abstract machine and
    type-safety proof that captures the essence of type variables at
    the C level.",
  paper = "Gros06.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Guil98,
  author = "Guillaume, Alexandre",
  title = {{De ALDOR a Zermelo}},
  school = "University Paris VI",
  year = "1998"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Guen19,
  author = "Gueneau, Armael",
  title = {{Mechanized Verification of the Correctness and Asymptotic
            Complexity of Programs}},
  school = "University of Paris",
  year = "2019",
  link = "\url{http://gallium.inria.fr/~agueneau/phd/manuscript.pdf}",
  abstract =
    "This dissertation is concerned with the question of formally
    verifying that the implementation of an algorithm is not only
    functionally correct (it always returns the right result), but
    also has the right asymptotic complexity (it reliably computes the
```

```
    result in the expected amount of time).

    In the algorithms literature, it is standard practice to
    characterize the performance of an algorithm by indicating its
    asymptotic time complexity, typically using Landau's ``big-O''
    notation. We first argue informally that asymptotic complexity
    bounds are equally useful as formal specifications, because they
    enable modular reasoning: a $O$ bound abstracts over the concrete
    cost expression of a program, and therefore abstracts over the
    specifics of its implementation. We illustrate -- with the help of
    small illustrative examples -- a number of challenges with the use
    of the $O$ notation, in particular in the multivariate case, that
    might be overlooked when reasoning informally.

    We put these considerations into practice by formalizing the $O$
    notation in the Coq proof assistant, and by extending an existing
    program verification framework, CFML, with support for a
    methodology enabling robust and modular proofs of asymptotic
    complexity bounds. We extend the existing framework of Separation
    Logic with Time Credits, which allows to reason at the same time
    about correctness and time complexity, and introduce negative time
    credits. Negative time credits increase the expressiveness of the
    logic, and enable convenient reasoning principles as well as
    elegant specifications. At the level of specifications, we show
    how asymptotic complexity specifications using $O$ can be
    integrated and composed within Separation Logic with Time
    Credits. Then, in order to establish such specifications, we
    develop a methodology that allows proofs of complexity in
    Separation Logic to be robust and carried out at a relatively high
    level of abstraction, by relying on two key elements: a mechanism
    for collecting and deferring constraints during the proof, and a
    mechanism for semi-automatically synthesizing cost expressions
    without loss of generality.

    We demonstrate the usefulness and practicality of our approach on
    a number of increasingly challenging case studies. These include
    algorithms whose complexity analysis is relatively simple (such as
    binary search, which is nonetheless out of the scope of many
    automated complexity analysis tools) and data structures (such as
    Okasaki's binary random access lists). In our most challenging
    case study, we establish the correctness and amortized complexity
    of a state-of-the-art incremental cycle detection algorithm: our
    methodology scales up to highly non-trivial algorithms whose
    complexity analysis intimately depends on subtle functional
    invariants, and furthermore makes it possible to formally verify
    OCaml code which can then actually be used as part of real world
    programs.",
  paper = "Guen19.pdf"
}
```

— **axiom.bib** —

```
@article{Gure12,
  author = "Gurevich, Yuri",
  title = {{What Is An Algorithm?}},
  journal = "LNCS",
  volume = "7147",
  pages = "31-42",
  year = "2012",
  abstract =
    "We attempt to put the title problem and the Church-Turing thesis into
    a proper perspective and to clarify some common misconceptions
    related to Turings analysis of computation. We examine two approaches
    to the title problem, one well-known among philosophers and another
    among logicians.",
  paper = "Gure12.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Gure18,
  author = "Gurevich, Yuri",
  title = {{Church-Turing Thesis Cannot Possibly Be True}},
  year = "2018",
  link =
  "\url{https://www.microsoft.com/en-us/research/video/church-turing-thesis-cannot-possibly-be-true/}",
  abstract =
    "The thesis asserts this: If an algorithm A computes a partial
    function f from natural numbers to natural numbers then f is partially
    recursive, i.e., the graph of f is recursively enumerable.

    The thesis has been formulated in 1930s. The only algorithms at the
    time were sequential algorithms. Sequential algorithms were
    axiomatized in 2000. This axiomatization was used in 2008 to prove the
    thesis for sequential algorithms, i.e., for the case where A ranges
    over sequential algorithms.

    These days, in addition to sequential algorithms, there are parallel
    algorithms, distributed algorithms, probabilistic algorithms, quantum
    algorithms, learning algorithms, etc.

    The question whether the thesis is true in full generality is actively
    discussed from 1960s. We argue that, in full generality, the thesis
    cannot possibly be true.",
  paper = "Gure18.pdf",
  keywords = "printed"
}
```

———-

1.2.8 H

— **axiom.bib** —

```
@misc{Hahn04,
  author = "Hahnle, Reiner and Wallenburg, Angela",
  title = {{Using a Software Testing Technique to Improve Theorem Proving}},
  year = "2004",
  link = "\url{http://www.cse.chalmers.se/~angelaw/papers/lic.pdf}",
  comment = "paper follows thesis in file",
  abstract =
    "Most efforts to combine formal methods and software testing go in the
    direction of exploiting formal methods to solve testing problems, most
    commonly test case generation.  Here we take the reverse viewpoint and
    show how the technique of partition testing can be used to improve a
    formal proof technique (induction for correctness of loops).  We first
    compute a partition of the domain of the induction variable, based on
    the branch predicates in the program code of the loop we wish to
    prove.  Based on this partition we derive mechanically a partitioned
    induction rule, which is (hopefully) easier to use than the standard
    induction rule.  In particular, with an induction rule that is
    tailored to the program to be verified, less user interaction can be
    expected to be required in the proof.  We demonstrate with a number of
    examples the effectiveness of our method.",
  paper = "Hahn04.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Hale07,
  author = "Hales, Thomas C.",
  title = {{The Jordan Curve Theorem, Formally and Informally}},
  journal = "Amer. Math. Monthly",
  volume = "114",
  number = "10",
  pages = "882-894",
  year = "2007"
}
```

———-

— **axiom.bib** —

```
@article{Hale10,
  author = "Hales, Thomas C. and Harrison, John and McLaughlin, Sean
            and Nipkow, Tobias and Obua, Steven and Zumkeller,
    Roland",
  title = {{A Revision of the Proof of the Kepler Conjecture}},
  journal = "Discrete and Computational Geometry",
```

```
  volume = "44",
  number = "1",
  pages = "1-34",
  year = "2010",
  abstract =
    "The Kepler conjecture asserts that no packing of congruent balls
    in three-dimentional Euclidean space has density greater than that
    of the face-centered cubic packing. The original proof, announced
    in 1998 and published in 2006, is long and complex. The process of
    revision and review did not end with the publication of the proof.
    This article summarizes the current status of a long-term
    initiative to reorganize the original proof into a more
    transparent form and to provide a greater level of certification
    of the correctness of the computer code and other details of the
    proof. A final part of this article lists errata in the original
    proof of the Keper conjecture.",
  paper = "Hale10.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Hale18,
  author = "Hales, Thomas C.",
  title = {{A Review of the Lean Theorem Prover}},
  year = "2018",
  link = "\url{htts://jiggerwit.wordpress.com/2018/09/18/a-review-of-the-lean-theorem-prover}",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Hall90,
  author = "Hall, Anthony",
  title = {{7 Myths of Formal Methods}},
  journal = "IEEE Software",
  volume = "7",
  number = "5",
  pages = "11-19",
  year = "1990",
  abstract =
    "Formal methods are difficult, expensive, and not widely useful,
    detractors say. Using a case study and other real-world examples, this
    article challenges such common myths.",
  paper = "Hall90.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hall94,
  author = "Hall, Condelia V.",
  title =
    {{Using Hindley-Milner Type Inference to Optimize List Representation}},
  booktitle = "LFP'94 Lisp and Functional Programming",
  publisher = "ACM",
  year = "1994",
  pages = "162-172",
  abstract =
    "Lists are a pervasive data structure in functional programs.  The
    generality and simplicity of their structure makes them expensive.
    Hindley-Mllner type inference and partial evaluation are all that is
    needed to optimise this structure, yielding considerable improvements
    in space and time consumption for some interesting programs.  This
    framework is applicable to many data types and their optimised
    representations, such as lists and parallel implementations of bags,
    or arrays and quadtrees.",
  paper = "Hall94.pdf, printed"
}
```

———-

— **axiom.bib** —

```
@book{Hami78,
  author = "Hamilton, A.G.",
  title = {{Logic for Mathematicians}},
  year = "1978",
  publisher = "Cambridge University Press",
  isbn = "0-521-29291-3",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Hamm18,
  author = "Hammack, Richard",
  title = {{Book of Proof}},
  publisher = "Hammack, Richard",
  year = "2018",
  paper = "Hamm18.pdf",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Hant76,
  author = "Hantler, Sidney L. King, James C.",
  title = {{An Introduction to Proving the Correctness of Programs}},
  journal = "ACM Computing Surveys",
  volume = "8",
  number = "3",
  pages = "331-353",
  year = "1976",
  abstract =
    "This paper explains, in an introductory fashion, the method of
    specifying the correct behavior of a program by the use of
    input/output assertions and describes one method for showing that the
    program is correct with respect to those assertions.  An initial
    assertion characterizes conditions expected to be true upon entry to
    the program and a final assertion characterizes conditions expected to
    be true upon exit from the program.  When a program contains no
    branches, a technique known as symbolic execution can be used to show
    that the truth of the initial assertion upon entry guarantees the
    truth of the final assertion upon exit.  More generally, for a program
    with branches one can define a symbolic execution tree.  If there is
    an upper bound on the number of times each loop in such a program may
    be executed, a proof of correctness can be given by a simple traversal
    of the (finite) symbolic execution tree.

    However, for most programs, no fixed bound on the number of times each
    loop is executed exists and the corresponding symbolic execution trees
    are infinite.  In order to prove the correctness of such programs, a
    more general assertion structure must be provided.  The symbolic
    execution tree of such programs must be traversed inductively rather
    than explicitly.  This leads naturally to the use of additional
    assertions which are called ``inductive assertions.''",
  paper = "Hant76.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@mastersthesis{Harg02a,
  author = "Hargreaves, G.",
  title = {{Interval analysis in MATLAB}},
  school = "University of Manchester, Dept. of Mathematics",
  year = "2002"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Harp85,
```

```
  author = "Harper, Robert",
  title = {{Aspects of the Implementation of Type Theory}},
  school = "Cornell University",
  year = "1985",
  comment = "TR 85-675",
  abstract =
    "This thesis is about building an automated programming logic. For our
    purposes an automated programming logic consists of
    \begin{itemize}
    \item A formal system for reasoning about programs
    \item A proof development environment which includes, at least, an
    editor for the construction of proofs in the logic
    \item Mechanized decision methods to assist in the proof development
    process
    \item A library mechanism for managing collections of theorems
    \end{itemize}",
  paper = "Harp85.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Harp89,
  author = "Harper, Robert and Mitchell, John C. and Moggi, Eugenio",
  title = {{Higher-Order Modules and the Phase Distinction}},
  booktitle = "Symp. on Principles of Programming Languages POPL'17",
  publisher = "ACM Press",
  year = "1989",
  comment = "CMU-CS-89-198",
  link = "\url{https://www.cs.cmu.edu/~rwh/papers/phase/tr.pdf}",
  pages = "341-354",
  abstract =
    "In earlier work, we used a typed function calculus, XML, with
    dependent types to analyze several aspects of the Standard ML type
    system. In this paper , we introduce a refinement of XML with a clear
    compile-time/run-time phase distinction, and a direct compile-time
    type checking algorithm. The calculus uses a finer separation of
    types into universes than XML and enforces the phase distinction using
    a nonstandard equational theory for module and signature
    expressions. While unusual from a type-theoretic point of view, the
    nonstandard equational theory arises naturally from the well-known
    Grothendieck construction on an indexed category.",
  paper = "Harp89.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Harp92,
  author = "Harper, Robert",
```

```
  title = {{Constructing Type Systems over an Operational Semantics}},
  journal = "J. Symbolic Computation",
  volume = "14",
  pages = "71-84",
  year = "1992",
  abstract =
    "Type theories in the sense of Martin-Lof and the NuPRL system are
    based on taking as primitive a type-free programming language
    given by an operational semantics, and defining types as partial
    equivalence relations on the set of closed terms. The construction
    of a type system is based on a general form of inductive
    definition that may either be taken as acceptable in its own
    right, or further explicated in terms of other patterns of
    induction. One suc account, based on a general theory of
    inductively defined relations, was given by Allen. An alternative
    account, based on an essentially set theoretic argument, is
    presented.",
  paper = "Harp92.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Harp93,
  author = "Harper, Robert and Honsell, Furio and Plotkin, Gordon",
  title = {{A Framework for Defining Logics}},
  journal = "J. ACM",
  volume = "40",
  number = "1",
  year = "1993",
  pages = "143-184",
  abstract =
    "The Edinburgh Logical Framework (LF) provides a means to define (or
    present) logics. It is based on a general treatment of syntax, rules,
    and proofs by means of a typed $\lambda$-calculus with dependent
    types. Syntax is treated in a style similar to, but more general than,
    Martin-Lof's system of arities. The treatment of rules and proofs
    focuses on his notion of a judgment. Logics are represented in LF via
    a new principle, the judgments as types principle, whereby each
    judgment is identified with the type of its proofs. This allows for a
    smooth treatment of discharge and variable occurence conditions and
    leads to a uniform treatment of rules and proofs whereby rules are
    viewed as proofs of higher-order judgments and proof checking is
    reduced to type checking. The practical benefit of our treatment of
    formal systems is that logic-independent tools, such as proof editors
    and proof checkers, can be constructed.",
  paper = "Harp93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Harp93a,
  author = "Harper, Robert and Mitchell, John C.",
  title = {{On the Type Structure of Standard ML}},
  journal = "Transactions on Programming Languages and Systems",
  publisher = "ACM",
  volume = "15",
  number = "2",
  pages = "211-252",
  year = "1993",
  abstract =
    "Standard ML is a useful programming module facility. One notable
    feature of the core expression language of ML is that it is implictly
    typed: no explicit type information need be supplied by the
    programmer.  In contrast, the module language of ML is explicitly
    typed; in particular, the types of parameters in parametric
    modules must be supplied by the programmer. We study the type
    structure of Standard ML by giving an explicitly-typed,
    polymmorphic function calculus that captures many of the essential
    aspects of both the core and module language. In this setting,
    implicitly-type core language expressions are reguarded as a
    convenient short-hand for an explicitly-typed counterpart in our
    function calculus. In contrast to the Girard-Reynolds polymorphic
    calculus, our function calculus is predicative: the type system
    may be built up by induction on type levels. We show that, in a
    precise sense, the language becomes inconsistent if restrictions
    imposed by type levels are relaxed. More specifically, we prove
    that the important programming features of ML cannot be added to
    any impredicative language, such as the Girard-Reynolds calcus,
    without implicitly assuming a type of all types.",
  paper = "Harp93a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Harp94,
  author = "Harper, Robert and Lillibridge, Mark",
  title = {{A Type-Theoretic Approach to Higher-Order Modules with Sharing}},
  booktitle = "POPL'21 Principles of Programming Languages",
  year = "1994",
  publisher = "ACM Press",
  abstract =
    "The design of a module system for constructing and maintaining
    large programs is a difficult task that raises a number of theoretical
    and practical issues.  A fundamental issue is the management of the
    flow of information between program units at compile time via the
    notion of an interface.  Experience has shown that fully opaque
    interfaces are awkward to use in practice since too much
    information is hidden, and that fully transparent interfaces lead to
    excessive interdependencies, creating problems for maintenance
```

```
    and separate compilation.  The ``sharing'' specications of Standard ML
    address this issue by allowing the programmer to specify equational
    relationships between types in separate modules, but are not
    expressive enough to allow the programmer complete control over
    the propagation of type information between modules.

    These problems are addressed from a type-theoretic viewpoint by
    considering a calculus based on Girard's system $F_\omega$.  The calculus
    differs from those considered in previous studies by relying exclusively
    on a new form of weak sum type to propagate information at
    compile-time, in contrast to approaches based on strong sums which
    rely on substitution.  The new form of sum type allows for the
    specification of equational, as well as type and kind, information
    in interfaces.  This provides complete control over the propagation
    of compile-time information between program units and is
    sufficient to encode in a straightforward way most uses of type
    sharing specifications in Standard ML.  Modules are treated as
    ``first-class'' citizens, and therefore the system supports
    higher-order modules and some object-oriented programming idioms: the
    language may be easily restricted to ``second-class' modules found in
    ML-like languages.",
  paper = "Harp94.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Harp97,
  author = "Harper, Robert and Stone, Christopher",
  title = {{An Interpretation of Standard ML in Type Theory}},
  type = "technical report",
  institution = "Carnegie Mellon University",
  number = "CMU-CS-97-147",
  year = "1997",
  abstract =
    "We define an interpretation of Standard ML into type theory. The
    interpretation takes the form of a set of elaboration rules
    reminiscent of the static semantics given in {\sl The Definition
    of Standard ML}. In particular, the elaboration rules are given in
    a relational style, exploiting indeterminancy to avoid
    over-commitment to specific implementation techniques. Elaboration
    consists of identifier scope resolution, type checking and type
    inference, expansion of derived forms, pattern compilation,
    overloading resolution, equality compilation, and the coercive
    aspects of signature matching.

    The underlying type theory is an explicitly-typed
    $\lambda$-calculus with product, sum, function, and recursive
    types, together with module types derived from the translucent sum
    formalism of Harper and Lillibridge. Programs of the type theory
    are given a type-passing dynamic semantics compatible with
    constructs such as polymorphic equality that rely on type analysis
```

```
    at run-time.",
  paper = "Harp97.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Harp12,
  author = "Harper, Robert",
  title = {{Type Theory Foundations}},
  year = "2012",
  link = "\url{https://www.cs.uoregon.edu/research/summerschool/summer12/videos/Harper1_0.mp4}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Harp16,
  author = "Harper, Robert",
  title = {{Practical Foundations for Progamming Languages}},
  publisher = "Cambridge University Press",
  year = "2016",
  isbn = "978-1-107-15030-0",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Harp18,
  author = "Harper, Robert",
  title = {{Computational Type Theory}},
  year = "2018",
  link =
  "\url{http://www.cs.uoregon.edu/research/summerschool/summer18/topics.php}",
  comment = "OPLSS 2018"
}
```

———-

— **axiom.bib** —

```
@misc{Harp18a,
  author = "Harper, Robert",
  title = {{Computational Type Theory Lectures}},
  year = "2018",
  link = "\url{http://www.youtube.com/watch?v=LEOSSLizYUI}",
```

```
  comment = "OPLSS 2018"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Harr96a,
  author = "Harrison, John",
  title = {{HOL Light: A Tutorial Introduction}},
  booktitle = "First Int. Conf. on Formal Methods in Computer-Aided Design",
  publisher = "unknownn",
  pages = "265-269",
  year = "1996"
}
```

———-

— **axiom.bib** —

```
@article{Harr09a,
  author = "Harrison, John",
  title = {{A Formalized Proof of Dirichlet's Theorem on Primes in
            Arithmetic Progression}},
  journal = "J. Formaliz. Reason.",
  volume = "2",
  number = "1",
  pages = "63-83",
  year = "2009"
}
```

———-

— **axiom.bib** —

```
@article{Harr09b,
  author = "Harrison, John",
  title = {{Formalizing an Analytic Proof of the Prime Number Theorem}},
  journal = "J. Automated Reasoning",
  volume = "43",
  pages = "243-261",
  year = "2009"
}
```

———-

— **axiom.bib** —

```
@misc{Harr13,
  author = "Harrison, John",
  title = {{A Survey of Automated Theorem Proving}},
```

```
  year = "2013",
  link = "\url{https://www.lektorium.tv/lecture/14805}"
}
```

———-

— **axiom.bib** —

```
@article{Hart63,
  author = "Hartmanis, J. and Stearns, R.E.",
  title = {{On the Computational Complexity of Algorithms}},
  journal = "Trans. American Mathematical Society",
  volume = "117",
  pages = "285-306",
  year = "1963",
  paper = "Hart63.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Hart95,
  author = "Hartmanis, Juris",
  title = {{On Computational Complexity and the Nature of Computer
           Science}},
  year = "1995",
  comment = "Turing Award Lecture",
  paper = "Hart95.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Harv19,
  author = "Harvey, David and van der Hoeven, Joris",
  title = {{Integer Multiplication in Time O(n log n)}},
  link = "\url{https://hal.archives-ouvertes.fr/hal.02070778/document}",
  year = "2019",
  abstract =
    "We present an algorithm that computes the product of two
    n-bit intgers in O(n log n) bit operations",
  paper = "Harv19.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Harv20,
  author = "Harvey, David",
  title = {{An Exponent One-Fifth Algorithm for Deterministic Integer
            Factorisation}},
  year = "2020",
  link = "\url{https://arxiv.org/pdf/2010.05450.pdf}",
  abstract =
    "Hittmeir recently presented a deterministic algorithm that
    provably computes the prime factorisation of a positive integer
    $N$ in $N^{2/9+o(1)}$ bit operations. Prior to this breakthrough,
    the best known complexity bound for this problem was
    $N^{1/4+o(1)}$, a result going back to the 1970s. In this paper we
    push Hittmeir's techniques further, obtaining a rigorous,
    deterministic factoring algorithm with complexity $N^{1/5+o(1)}$.",
  paper = "Harv20.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Hava95,
  author = "Havas, George and Majewski, Bohdan and Matthews, K.R.",
  title = {{Extended GCD Algorithms}},
  type = "technical report",
  institution = "University of Queensland",
  number = "TR0302",
  year = "1995",
  abstract =
    "Extended gcd calculation has a long history and plays an
    important role in computational number theory and linear
    algebra. Recent results have shown that finding optimal
    multipliers in extended gcd calculations is difficult. We study
    algorithms for finding good multipliers and present new algorithms
    with improved performance. We present a well-performing algorithm
    which is based on lattice basis reduction methods and may be
    formally analyzed. We also give a relatively fast algorithm with
    moderate performance.",
  paper = "Hava95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Hava97,
  author = "Havas, George and Majewski, Bohdan",
  title = {{Extended GCD Algorithms}},
  type = "technical report",
  institution = "University of Queensland",
  number = "TR0325",
```

```
  year = "1997",
  abstract =
    "Given an integer vector of $n$ positive number numbers
    $a=\vert a_i \vert^n_{i=1}$ the extended gcd problem asks for an
    integer vector $x$ of length $n$ such that
    \[xa^T=\sum_{i=1}^n x_ia_i = gcd(a_1,a_2,\ldots,a_n)\]

    For many applications it is vital that some measure of $x$,
    $\vert{x}\vert$ is small. We have proved, however, that if we choose
    either the max norm or the zero matric the question of finding
    $x$ such that $\vert{x}\vert$ is smaller than some positive constant
    $K$ is NP-complete. We conjecture that the questions remains
    NP-complete for other norms.

    In the light of these results we hae proposed two approximation
    algorithms. Their respective complexities are
    $O(n^2 log(max_i\{a_i\}))$ and $O(n^4 log(max_i\{a_i\}))$.
    Theoretical analysis of the algorithms leads
    to unsatisfactory bounds on the quality of the solution. Thus here
    we undertake a practical study of the methods, where their
    performance is matched against optimal solutions.",
  paper = "Hava97.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Hava98,
  author = "Havas, George and Majewski, Bohdan S. and Matthews, Keith R.",
  title = {{Extended GCD and Hermite Normal Form Algorithms via
           Lattice Basis Reduction}},
  journal = "Experimental Mathematics",
  volume = "7",
  number = "2",
  pages = "125-136",
  year = "1998",
  abstract =
    "Extended gcd calculation has a long history and plays an
    important role in computational number theory and linear
    algebra. Recent results have shown that finding optimal
    multipliers in extended gcd calculations is difficult. We present
    an algorithm which uses lattice basis reduction to produce small
    integer multipliers $x_1,\ldots,x_m$ for the equation
    $s=gcd(x_1,\ldots,x_m)-x_1s_1+\ldots+x_ms_m$ where
    $s_1,\ldots,s_m$ are given integers. The method generalises to
    produce small unimodular transformation matrices for computing the
    hermite normal form of an integer matrix.",
  paper = "Hava98.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Hayn87,
  author = "Haynes, Christopher T. and Friedman, Daniel P.",
  title = {{Embedding Continuations in Procedural Objects}},
  type = "technical report",
  institution = "Indiana University",
  number = "213",
  year = "1987",
  abstract =
    "Continuations, when available as first-class objects, provide a
    general control abstraction in programming languages. They
    liberate the programmer from specific control structures,
    increasing programming language extensibility. Such continuations
    may be extended by embedding them in procedural objects. This
    technique is first used to restore a fluid environment when a
    continuation object is invoked. We then consider techniques for
    constraining the power of continuations in the interest of
    security and efficiency. Domain mechanisms, which create dynamic
    barriers for enclosing control, are implemented using
    fluids. Domains are then used to implement an unwind-protect
    facility in the presence of first-class continuations. Finally, we
    present two mechanisms, wind-unwind and dynamic-wind, that
    generalizes unwind-protect.",
  paper = "Hayn87.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Hear71,
  author = "Hearn, Anthony C.",
  title = {{Applications of Symbol Manipulation in Theoretical Physics}},
  journal = "Communications of the ACM",
  volume = "14",
  number = "8",
  pages = "511-516",
  year = "1971",
  paper = "Hear71.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Hear72,
  author = "Hearn, Anthony C.",
  title = {{An Improved Non-Modular Polynomial GCD Algorithm}},
  journal = "ACM SIGSAM Bulletin",
  volume = "23",
```

```
  pages = "10-15",
  year = "1972",
  abstract =
    "An improved non-modular algorithm for the calculation of the
    greatest common divisor of two multivariate polynomials is
    presented.",
  paper = "Hear72.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Hear73,
  author = "Hearn, Anthony C.",
  title = {{REDUCE-2 Users Manual}},
  comment = "Computing Physics Group",
  publisher = "University of Utah",
  year = "1973"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hear74,
  author = "Hearn, Anthony C.",
  title = {{A Mode Analysing Algebraic Manipulation Program}},
  booktitle = "Proc. 1974 annual ACM Conference. Vol 2",
  publisher = "ACM",
  year = "1974"
}
```

———-

— **axiom.bib** —

```
@article{Hear79,
  author = "Hearn, Anthony C.",
  title = {{Non-modular computation of polynomial GCDs using Trial Division}},
  journal = "LNCS",
  volume = "72",
  pages = "227-239",
  year = "1979",
  abstract =
    "This paper describes a new algorithm for the determination of the
    GCD of two multivariate polynomials by non-modular means.",
  paper = "Hear79.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hear93,
  author = "Hearn, Anthony C. and Schufer, Eberhard",
  title = {{An Order-sorted Approach to Algebraic Computation}},
  booktitle = "DISCO 1993",
  year = "1993",
  pages = "134-144",
  publisher = "Springer",
  abstract =
    "This paper presents the prototype design of an algebraic
    computation system that manipulates algebraic quantities as
    generic objects using order-sorted algebra as the underlying
    model. The resulting programs have a form that is closely related
    to the algorithmic description of a problem, but with the security
    of full type checking in a compact, natural style.",
  paper = "Hear93.pdf",
  keywords = "printed, axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Hebi07,
  author = "Hebisch, Waldemar",
  title = {{FriCAS Project Statement}},
  year = "2007",
  link = "\url{http://www.math.uni.wroc.ps/~hebish/fricas/fricas-reg.html}",
  abstract =
    "Problem Statement: FriCAS a fork of Axiom project. Its starting
    point is wh-sandbox branch of the Axiom project. Axiom project
    tried to use literate programming methodology and switch emphasis
    from code to documentation. In practice that meants that almost
    all code is wrapped in so called pamphlet files and must be
    extracted by noweb tool before use in build process. This causes
    significant difficulties during developement, starting from having
    long file names, problems of tracking errors to exact source
    locations and complicationss in Makefiles. Literate programming
    practice in Axiom produced little documentation, but makes sources
    harder to read for programmer. FriCAS will use traditional
    methodology for new developement and gradually convert other files
    back to traditional form (back because Axiom was originally
    developed using traditional methodology, and only during
    transition to open source project files where mechanically
    converted to ``literate'' form). Axiom project evolved very
    slowly, many simple fixes were not applied for long time
    (years). FriCAS will use lightweight developement, allowing much
    faster evolution.

    Short term planned new technical developments: Axiom system is
    written in multiple languages, main two called Boot and Spad are
    specific to Axiom. Boot and Spad are translated to Common
```

```
    Lisp. Currently Axiom is fully functional when working on top of
    Gnu Common Lisp (GCL). In wh-sandbox branch core functionlity
    works also with other Lisp implementations, and FriCAS should
    fully support other Lisp implementations. This should increase
    acceptance of FriCAS, because on some platforms GCL is hard to
    build, on some does not work at all  and also Lisp developers
    frequently find other Lisp implementation preferable. The Spad
    language used for mathematical code in Axiom is strongly
    typed. Axiom system caches type information between builds --
    during build it uses type information from previous build. This
    makes modifications to Axiom code very difficult, because after
    changing type in source old type information is used in many
    cases, causing spurious type errors. FriCAS will use only type
    information from sources eliminationg such problems. Many modern
    mathematical codes to get better speed are written in C or
    C++. FriCAS will interface to such codes to gain speed
    advantage. Currently Axiom system offers its own user interface,
    but it is somewhat dated -- mostly text based,graphics has old
    looks and feel. FriCAS add hoos which make adding alternative user
    interfaces easier.

    Longer term plans: The compiler for Spad language included in
    Axiom is buggy and has serious performance problems -- new
    compiler is needed. New mathematical algorithms. Assertion
    support.",
}
```

———-

— **axiom.bib** —

```
@misc{Hebi16,
  author = "Hebisch, Waldemar",
  title = {{Integration in terms of exponential integrals and
            incomplete gamma functions}},
  year = "2016",
  link = "\url{http://www.math.uni.wroc.pl/~hebisch/other/icms.pdf}",
  comments = "slides",
  paper = "Hebi16.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Hebi20,
  author = "Hebisch, Waldemar",
  title = {{History of FriCAS}},
  year = "2020",
  link = "\url{http://fricas.sourceforge.net/history.html}",
  keywords = "axiomref, DONE"
```

```
}
```

———-

— **axiom.bib** —

```
@book{Heij67,
  author = "Heijenoort, Jean van",
  title = {{From Frege to Godel}},
  publisher = "Harvard University Press",
  isbn = "0-674-32449-8",
  year = "1967",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Hema13,
  author = "Hemann, Jason and Friedman, Daniel P.",
  title = {{uKanren: A Minimal Functional Core for Relational Programming}},
  year = "2013",
  link = "\url{webyrd.net/scheme-2013/papers/HemannMuKanren2013.pdf}",
  abstract =
    "This paper presents $\mu$Kanren, a minimalist language in the
    miniKanren family of relational (logic) programming languages. Its
    implementation comprises fewer than 40 lines of Scheme. We
    motivate the need for a minimalist miniKanren language, and
    iteratively develop a complete search strategy. Finally, we
    demonstrate that through sufficient user-level features one
    regains much of the expressiveness of other miniKanren
    languages. In our opinion its brevity and simple semantics make
    $\mu$Kanren uniquely elegant.",
  paper = "Hema13.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Hema14,
  author = "Hemann, Jason and Friedman, Daniel",
  title = {{Write the Other Half of Your Program}},
  year = "2014",
  link = "\url{https://www.youtube.com/watch?v=RG9fBbQrVOM}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hend76,
  author = "Henderson, Peter and Morris Jr., James H.",
  title = {{A Lazy Evaluator}},
  booktitle = "3rd Symp. on Principles of Programming Languages",
  publisher = "ACM",
  pages = "95-103",
  year = "1976",
  abstract =
    "A different way to execute pure LISP programs is presented. It
    delays the evaluation of parameters and list structures without
    ever having to perform more evaluation steps than the usual
    method. Although the central idea can be found in earlier work
    this paper is of interest since it treats a rather well-known
    language and works out an algorithm which avoids full
    substitution. A partial correctness proof using Scott-Strachey
    semantics is sketched in a later section.",
  paper = "Hend76.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Heng89,
  author = "Henglein, Friedrich",
  title = {{Polymorphic Type Inference and Semi-Unification}},
  school = "Rutgers",
  year = "1989",
  link =
  "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.388.1275&rep=rep1&type=pdf}",
  abstract =
    "In the last ten years declaration-free programming languages with
    a polymorphic typing discipline (ML, B) have been developed to
    approximate the flexibility and conciseness of dynamically typed
    languages (LISP, SETL) while retaining the safety and execution
    efficiency of conventional statically typed languages (Algol68,
    Pascal). These polymorphic languages can be type checked at
    compile time, yet allow functions whose arguments range over a
    variety of types.

    We investigate several polymorphic type systems, the most powerful
    of which, termed Milner-Mycroft Calculus, extends the so-called
    let-polymorphism found in, e.g. ML with a polymorphic typing rule
    for recursive defiitions. We show that semi-unification, the
    problem of solving inequalities over first-order terms,
    characterizes type checking in the Milner-Mycroft Calculus to
    polynomial time, even in the restricted case were nested
    definitions are disallowed. This permits us to extend some
    infeasibility results for related combinatorial problems to type
    inference and to correct several claims and statements in the
```

```
    literature.

    We prove the existence of unique most general solutions of term
    inequalities, called most general semi-unifiers, and present an
    algorithm for computing them that terminates for all known inputs
    due to a novel ‘‘extended occurs check’’. We conjecture this
    algorithm to be uniformly terminating even though, at present,
    general semi-unification is not known to be decidable. We prove
    termination of our algorithm for a restricted case of
    semi-unification that is of independent interest.

    Finally, we offer an explanation for the apparent practicality of
    polymorphic type inference in the face of theoretical
    intractability results.",
  paper = "Heng89.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Heng94,
  author = "Henglein, Fritz",
  title = {{Dynamic Typing: Syntax and Proof Theory}},
  journal = "Science of Computer Programming",
  volume = "23",
  pages = "197-230",
  year = "1994",
  abstract =
    "We present the {\sl dynamically typed $\lambda$-calculus}, an
    extension of the statically typed $\lambda$-calculus with a
    special type Dyn and explicit {\sl dynamic type coercions}
    corresponding to run-time type tagging and type check-and-untag
    operations. Programs in run-type typed languages can be
    interpreted in the dynamically typed $\lambda$-calculus via a
    nondeterministic {\sl completion process} that inserts explicit
    coercions and type declarations such that a well-typed term
    results.

    We characterize when two different completions of the same
    run-time typed programs are {\sl coherent} with an equational
    theory that is independent of the underlying
    $\lambda$-theory. This theory is refined by orienting some
    equations to define {\sl safety} and {\sl minimality} of
    completions. Intuitively, a safe completion is one that does not
    produce an error at run-time which another completion would have
    avoided, and a minimal completion is a safe completion that
    executes fewest tagging and check-and-untag operations amongst all
    safe completions.

    We show that every untyped $\lambda$-term has a safe completion at
    any type and that it is unique modulo a suitable congruence
    relation. Furthermore, we present a rewriting system for
```

```
    generating minimal completions. Assuming strong normalization of
    this rewriting system we show that every $\lambda I$-term has a
    minimal completion at any type, which is furthermore unique modulo
    equality in the dynamically typed $\lambda$-calculus.",
  paper = "Heng94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Henk94,
  author = "Henke, F.W. von and Dold, A. and Ruess, H. and Schwier, D.
            and Strecker, M.",
  title = {{Construction and Deduction Methods for the Formal
            Development of Software}},
  journal = "LNCS",
  volume = "1009",
  year = "1994",
  abstract =
    "In this paper we present an approach towards a framework based on
    the type theory ECC (Extended Calculus of Constructions) in which
    specifications, programs and operators for modular development by
    stepwise refinement can be formally described and reasoned
    about. We show that generic software development steps can be
    expressed as higher-order functions and demonstrate that proofs
    about their asserted effects can be carried out in the underlying
    logical calculus.

    For transformations requiring syntactic manipulations of objects,
    a two-level system comprising a Meta- and an Object-level is
    provided, and it is shown how transformations can be formalized
    that faithfully represent operators on the object level.",
  paper = "Henk94.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Henk96,
  author = "Henke, F.W. von and Luther, M. and Pfeifer, H. and Ruess, H.
            and Schwier, D. and Strecker, M. and Wagner, M.",
  title = {{The TYPELAB Specification and Verification Environment}},
  journal = "LNCS",
  volume = "1101",
  pages = "604-607",
  year = "1996",
  paper = "Henk96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Henn90,
  author = "Hennicker, Rolf",
  title = {{Context Induction: A Proof Principle for Behavioural
            Abstractions}},
  booktitle = "DISCO 1990",
  year = "1990",
  pages = "101-110",
  abstract =
    "An induction principle, called context induction, is presented
    which is appropriate for the verification of behavioural
    properties of abstract data types. The usefulness of the proof
    principle is documented by several applications: the verification
    of behavioural theorems over a behavioural specification, the
    verification of behavioural implementations and the verification
    of ‘‘forget=restrict-identify’’ implementations.

    In particular it is shown that behavioural implementations and
    ‘‘forget-restrict-identify’’ implementations (under certain
    assumptions) can be characterized by the same context condition,
    i.e. (under the given assumptions) both concepts are
    equivalent. This leads to the suggestion to use context induction
    as a uniform proof method for correctness proofs of formal
    implementations.",
  paper = "Henn90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Hera09,
  author = "Heras, Jonathan and Pascual, Vico and Rubio, Julio",
  title = {{Using Open Mathematical Documents to Interface Computer
            Algebra and Proof Assistant Systems}},
  journal = "LNCS",
  volume = "5625",
  year = "2009",
  abstract =
    "Mathematical Knowledge can be encoded by means of Open
    Mathematical Documents (OMDoc) to interface both Computer Algebra
    and Proof Assistant systems. In this paper, we show how a unique
    OMDoc structure can be used to dynamically generate, both a
    Graphical User Interface for a Computer Algebra system and a
    script for a Proof Assistant. This generic approach has been made
    concrete through a first prototype interfacing the Kenzo Computer
    Algebra system with the ACL2 Theorem Prover, both based on the
```

```
    Common Lisp programming language. An OMDoc repository has been
    developed allowing the user to customize the application in an
    easy way.",
  paper = "Hera09.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Hera13,
  author = "Heras, Jonathan and Komendantskaya, Ekaterina",
  title = {{ML4PG in Computer Algebra Verification}},
  journal = "LNCS",
  volume = "7961",
  year = "2013",
  abstract =
    "ML4PG is a machine-learning extension that provides statical
    proof hints during the process of Coq/SSReflect proof
    development. In this paper, we use ML4PG to find proof patterns in
    the CoqEAL library -- a library that was devised to verify the
    correctness of Computer Algebra algorithms. In particular, we use
    ML4PG to help us in the formalisation of an efficient algorithm to
    computer the inverse of triangular matrices.",
  paper = "Hera13.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Herd20,
  author = "Herda, Michal",
  title = {{The Common Lisp Condition System}},
  publisher = "Apress",
  year = "2020",
  isbn = "978-1-4842-6133-0",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Herk88,
  author = "Herken, Rolf",
  title = {{The Universal Turing Machine}},
  publisher = "Oxford Science Publications",
  year = "1988",
  isbn = "0-19-853774-3",
```

```
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Herm07,
  author = "Herman, David",
  title = {{Functional Pearl: The Great Escape}},
  booktitle = "ICFP'07",
  publisher = "ACM",
  isbn = "1-59593-815-2",
  year = "2007",
  abstract =
    "Filinski showed that callcc and a single mutable reference cell
    are sufficient to express the delimited control operators shift
    and reset. However, this implementation interacts poorly with
    dynamic bindings like exception handlers. We present a variation
    on Filinski's encoding of delimited continuations that behaves
    appropriately in the presence of exceptions and give an
    implementation in Standard ML of New Jersey. We prove the encoding
    correct with respect to the semantics of delimited dynamic binding.",
  paper = "Herm07.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Herr73,
  author = "Herrlich, Horst and Strecker, G.E.",
  title = {{Category Theory: An Introduction}},
  year = "1973",
  publisher = "Allyn and Bacon"
}
```

———-

— **axiom.bib** —

```
@article{Hetz12,
  author = "Hetzl, Stefan",
  title = {{Project Presentation: Algorithmic Structuring and
            Compression of Proofs (ASCOP)}},
  journal = "LNCS",
  volume = "7362",
  year = "2012",
  abstract =
    "Computer-generated proofs are typically analytic, i.e. they
    essentially consist only of formulas which are present in the
```

```
    theorem that is shown. In contrast, mathematical proffs written by
    humans almost never are: they are highly structured due to the use
    of lemmas.

    The ASCOP project aims at developing algorithms and software which
    structure and abbreviate analytic proofs by computing useful
    lemmas. These algorithms will be based on recent groundbreaking
    results establishing a new connection between proof theory and
    formal language theory. This connecion allows the application of
    efficient algorithms based on formal grammars to structure and
    compress proofs.",
  paper = "Hetz12.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Heyt56,
  author = "Heyting, A.",
  title = {{Intuitionism: An Introduction}},
  publisher = "North Holland",
  year = "1956",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Hick12,
  author = "Hickey, Rich",
  title = {{The Language of the System}},
  link = "\url{https://www.youtube.com/watch?v=ROor6_NGIWU}",
  year = "2012",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Hick12a,
  author = "Hickey, Rich",
  title = {{Clojure for Lisp Programmers Part 1}},
  link = "\url{https://www.youtube.com/watch?v=cPNkH-7PRTk}",
  year = "2012",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Hick12b,
  author = "Hickey, Rich",
  title = {{Clojure for Lisp Programmers Part 2}},
  link = "\url{https://www.youtube.com/watch?v=7mbcYxHoOOnM}",
  year = "2012",
  keywords = "DONE"
}
```

————-

— **axiom.bib** —

```
@misc{Hick15,
  author = "Hickey, Rich",
  title = {{Clojure, Made Simple}},
  link = "\url{https://www.youtube.com/watch?v=VSdnJDO-xdg}",
  year = "2015",
  keywords = "DONE"
}
```

————-

— **axiom.bib** —

```
@misc{Hick16,
  author = "Hickey, Rich",
  title = {{Spec-ulation}},
  link = "\url{https://www.youtube.com/watch?v=oyLBGkS5ICk}",
  year = "2016",
  keywords = "DONE"
}
```

————-

— **axiom.bib** —

```
@misc{Hick17,
  author = "Hickey, Rich",
  title = {{Effective Programs - 10 Years of Clojure}},
  link = "\url{https://www.youtube.com/watch?v=2V1FtfBDsLU}",
  year = "2017",
  keywords = "DONE"
}
```

————-

— **axiom.bib** —

```
@misc{Hick18,
  author = "Hickey, Rich",
  title = {{Maybe Not}},
  link = "\url{https://www.youtube.com/watch?v=YR5WdGrpoug}",
  year = "2018",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@book{Hilb52,
  author = "Hilbert, David and Cohn-Vossen, S.",
  title = {{Geometry and the Imagination}},
  publisher = "Chelsea Publishing Company",
  year = "1952",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Hind69,
  author = "Hindley, R.",
  title = {{The Principal Type-Scheme of an Object in Combinatory Logic}},
  journal = "Trans. AMS",
  volume = "146",
  year = "1969",
  pages = "29-60",
  paper = "Hind69.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Hind97,
  author = "Hindley, J. Roger",
  title = {{Basic Simple Type Theory}},
  publisher = "Cambridge",
  year = "1997",
  isbn = "0-521-05422-2",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Hoar71,
  author = "Hoare, Charles Antony Richard",
  title = {{Proof of a Program: FIND}},
  journal = "Communications of the ACM",
  volume = "14",
  number = "1",
  year = "1971",
  abstract =
    "A proof is given of the correctness of the algorithm 'Find.'
    First, an informal description is given of the purpose of the
    program and the method used. A systematic technique is described
    for constructing the program proof during the process of coding
    it, in such a way as to prevent the intrusion of logical
    errors. The prrof of termination is treated as a separate
    exercie. Finally, some conclusions related to general programming
    methodology are drawn.",
  paper = "Hoar71.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Hoar72a,
  author = "Hoare, C.A.R",
  title = {{Proof of Correctness of Data Representations}},
  journal = "Acta Informatica",
  volume = "1",
  pages = "271-281",
  year = "1972",
  abstract =
    "A powerful method of simplifying the proofs of program
    correctness is suggested; and some new light is shed on the
    problem of functions with side-effects.",
  paper = "Hoar72a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Hoar74,
  author = "Hoare, C.A.R",
  title = {{Hints on Programming Language Design}},
  journal = "Computer Systems Reliability",
  volume = "20",
  pages = "505-534",
  year = "1974",
  link = "\url{http://flint.cs.yale.edu/cs428/doc/HintsPL.pdf}",
```

```
  abstract =
    "This paper presents the view that a programming language
    is a tool that should assist the programmer in the most difficult
    aspects of his art, namely program design, documentation, and
    debugging. It discusses the objective criteria for evaluating a
    language design and illustrates them by application to language
    features of both high-level languages and machine-code
    programming. It concludes with an annotated reading list,
    recomended for all intending language designers.",
  paper = "Hoar74.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Hoar81,
  author = "Hoare, Charles Antony Richard",
  title = {{The Emperor's Old Clothes}},
  journal = "CACM",
  volume = "24",
  number = "2",
  pages = "75-83",
  year = "1981",
  abstract =
    "The author recounts his experiences in the implementation,
    design, and standardization of computer programming languages, and
    issues a warning for the future.",
  paper = "Hoar18.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@book{Hoar85,
  author = "Hoare, C.A.R and Shepherdson, J.C.",
  title = {{Mathematical Logic and Programming Languages}},
  year = "1985",
  publisher = "Prentice-Hall",
  isbn = "0-13-561465-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Hoar87,
```

```
  author = "Hoare, Charles Antony Richard",
  title = {{An Overview of Some Formal Methods for Program Design}},
  journal = "Computer",
  year = "1987",
  volume = "20",
  number = "9",
  abstract =
    "The code of a computer program is a formal text, describing
    precisely the actions of a computer executing that program. As in
    other branches of engineering, the progress of its implementation
    as well as its eventual quality can be promoted by additional
    design documents, formalized before starting to write the final
    code. These preliminary documents may be expressed in a variety of
    notations suitable for different purposes at different stages of a
    project, from capture of requirements through design and
    implementation, to delivery and long-term maintenance. These
    notations are derived from mathematics, and include algebra,
    logic, functions, and procedures. The connection between the
    notations is provided by mathematical calculation and proof.

    This article introduces and illustrates a selection of formal
    methods by means of a single recurring example, the design of a
    program to compute the greatest common divisor of two positive
    numbers. It is hoped that some of the conclusions drawn from
    analysis of this simple example will apply with even greater force
    to software engineering projects on a more realistic scale.",
  paper = "Hoar87.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Hoar95,
  author = "Hoare, Tony",
  title = {{Unification of Theories: A Challenge for Computing Science}},
  journal = "Lecture Notes in Computer Science",
  volume = "1130",
  year = "1995",
  publisher = "Springer",
  abstract =
    "Unification of theories is the long-standing goal of the natural
    sciences; and modern physics offers a spectacular paradigm of its
    achievement. The structure of modern mathematics has also been
    determined by its great unifying theories -- topology, algebra and
    the like. The same ideals and goals are shared by researchers and
    students of theorectical computing science.",
  paper = "Hoar95.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Hoar96,
  author = "Hoare, C.A.R",
  title = {{How did software get so reliable without proof?}},
  journal = "LNCS",
  volume = "1051",
  year = "1996",
  abstract =
    "By surveying current software engineering practice, this paper
    reveals that the techniques employed to achieve reliability are little
    different from those which have proved effective in all other branches
    of modern engineering: rigorous management of procedures for design
    inspection and review; quality assurance based on a wide range of
    targeted tests; continuous evolution by removal of errors from
    products already in widespread use; and defensive programming, among
    other forms of deliberate over-engineering. Formal methods and proof
    play a small direct role in large scale programming; but they do
    provide a conceptual framework and basic understanding to promote the
    best of current practice, and point directions for future
    improvement.",
  paper = "Hoar96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Hoar03,
  author = "Hoare, Tony",
  title = {{The Verifying Compiler: A Grand Challenge for Computing
            Research}},
  journal = "Journal of the ACM",
  volume = "50",
  number = "1",
  pages = "63-69",
  year = "2003",
  link = "\url{https://vimeo.com/39256698}",
  abstract =
    "This contribution proposes a set of criteria that distinguish a
    grand challenge in science or engineering from the many other
    kinds of short-term or long-term research problems that engage the
    interest of scientists and engineers. As an example drawn from
    Computer Science, it revives an old challenge: the contruction and
    application of a verifying compiler that guarantees correctness of
    a program before running it.",
  paper = "Hoar03.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Hoar04,
  author = "Hoare, C.A.R",
  title = {{Unified Theories of Programming}},
  year = "2004",
  link = "\url{https://fi.ort.edu.uy/innovaportal/file/20124/1/04-hoard_unified_theories.pdf}",
  abstract =
    "Professional practice in a mature engineering discipline is based
    on relevant scientific theories, usually expressed in the language
    of mathematics. A mathematical theory of programming aims to
    provide a similar basis for specification, design and
    implementation of computer programs. The theory can be presented
    in a variety of styles, including
    \begin{enumerate}
    \item Denotational, relating a program to a specification of its
    observable properties and behaviour
    \item Algebraic, providing equations and inequalities for
    comparison, transformation and optimisation of designs and
    programs.
    \item Operational, describing individual steps of a possible
    mechanical implementation
    \end{enumerate}

    This paper presents simple theories of sequential
    non-deterministic programming in each of these three styles; by
    deriving each presentation from its predecessor in a cyclic
    fashion, mutual consistency is assured.",
  paper = "Hoar04.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Hodg95,
  author = "Hodges, Wilfrid",
  title = {{The Meaning of Specifications I: Domains and Initial Models}},
  journal = "Theoretical Computer Science",
  volume = "192",
  issue = "1",
  year = "1995",
  pages = "67-89",
  abstract =
    "This is the first of a short series of papers intended to provide one
    common semantics for several different types of specification
    language, in order to allow comparison and translations. The
    underlying idea is that a specification describes the behaviour of a
    system, depending on parameters. We can represent this behaviour as a
    functor which acts on structures representing the parameters, and
    which yields a structure representing the behaviour. We characterise
    in domain-theoretic terms the class of functors which could in
```

```
    principle be specified and implemented; briefly, they are the functors
    which preserve directed colimits and whose restriction to finitely
    presented structures is recursively enumerable. We also characterise
    those functors which allow specification by initial semantics in
    universal Horn classes with finite vocabulary; these functors consist
    of a free functor (i.e. left adjoint of a forgetful functor) followed
    by a forgetful functor. The main result is that these two classes of
    functor are the same up to natural isomorphism.",
  paper = "Hodg95.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hoei04,
  author = "Hoeij, Mark van and Monagan, Michael",
  title = {{Algorithms for Polynomial GCD Computation over Algebraic
           Function Fields}},
  booktitle = "Proc. ISSAC 04",
  year = "2004",
  isbn = "1-58113-827-X",
  link = "\url{http://www.cecm.sfu.ca/personal/mmonagan/papers/AFGCD.pdf}",
  abstract =
    "Let $L$ be an algebraic function field in $k \ge 0$ parameters
    $t_1,\ldots,t_k$. Let $f_1$, $f_2$ be non-zero polynomials in
    $L[x]$. We give two algorithms for computing their gcd. The first, a
    modular GCD algorithm, is an extension of the modular GCD algorithm
    for Brown for {\bf Z}$[x_1,\ldots,x_n]$ and Encarnacion for {\bf
    Q}$(\alpha[x])$ to function fields. The second, a fraction-free
    algorithm, is a modification of the Moreno Maza and Rioboo algorithm
    for computing gcds over triangular sets. The modification reduces
    coefficient grownth in $L$ to be linear.  We give an empirical
    comparison of the two algorithms using implementations in Maple.",
  paper = "Hoei04.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Hoev20,
  author = "van der Hoeven, Joris",
  title = {{Overview of the Mathemagix Type System}},
  year = "2020",
  link = "\url{https://www.texmacs.org/joris/mmxtyping/mmxtyping.html}",
  abstract =
    "The goal of the Mathematix project is to develop a new and free
    software for computer algebra and computer analysis, based on a
    strongly typed and compiled language. In this paper, we focus on
    the underlying type system of this language, which allows for
    heavy overloading, including parameterized overloading with
```

```
    parameters in so called ‘‘categories’’. The exposition is informal
    and aims at giving the reader an overview of the main concepts,
    ideas and differences with existing languages. In a forthcoming
    paper, we intend to describe the formal semantics of the type
    system in more details.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hohl85,
  author = "Hohlfeld, Bernhard",
  title = {{Correctness Proofs of the Implementation of Abstract Data
            Types}},
  booktitle = "European COnference on Computer Algebra",
  publisher = "Springer",
  pages = "446-447",
  year = "1985",
  comment = "LNCS 204",
  paper = "Hohl85.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Holz11,
  author = "Holzl, Johannes and Heller, Armin",
  title = {{Three Chapters of Measure Theory in Isabelle / HOL}},
  booktitle = "Interactive Theorem Proving",
  publisher = "Springer",
  pages = "135-151",
  year = "2011"
}
```

———-

— **axiom.bib** —

```
@misc{HOPLxx,
  author = "unknown",
  title = {{Scratchpad II}},
  year = "2020",
  link = "\url{https://hopl.info/showlanguage2.prx?exp=566}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Horg93,
  author = "Horgan, John",
  title = {{The Death of Proof}},
  year = "1993",
  publisher = "Scientific American",
  paper = "Horg93.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Horo11,
  author = "Horozal, Fulya and Iacob, Alin and Jucovschi, Constantin
            and Kohlhase, Michael and Rabe, Florian",
  title = {{Combining Source, Content, Presentation, Narration, and
            Relational Presentation}},
  journal = "LNCS",
  volume = "6824",
  year = "2011",
  abstract =
    "In this paper, we try to bridge the gap between different
    dimensions / incarnations of mathematical knowledge: MKM
    representation formats (content), their human-oriented languages
    (source, presentation), their narrative linearizations
    (narration), and relational presentations used in the semantic
    web. The central idea is to transport solutions from software
    engineering to MKM regarding the parallel interlinked maintenance
    of the different incarnations. We show how the integration of
    these incarnations can be utilized to enrich the authoring and
    viewing processes, and we evaluate our infrastructure on the LATIN
    Logic Atlas, a modular library of logic formalizations, and a set
    of computer science lecture notes written in STEX -- a modular,
    semantic variant of LATEX.",
  paper = "Horo11.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Horo12,
  author = "Horozal, Fulya and Kohlhase, Michael and Rabe, Florian",
  title = {{Extending MKM Formats at the Statement Level}},
  journal = "LNCS",
  volume = "7362",
  year = "2012",
  abstract =
    "Successful representation and markup languages find a good
```

```
    balance between giving the user freedom of expression, enforcing
    the fundamental semantic invariants of the modeling framework, and
    allowing machine support for the underlying semantic
    structures. MKM formats maintain strong invariants while trying to
    be foundationally unconstrained, which makes the induced design
    problem particularly challenging.

    In this situation, it is standard practice to define a minimal
    core language together with a scripting/macro facility for
    syntactic extensions that map into the core language. In practice,
    such extension facilities are either fully unconstrained (making
    invariants and machine support difficult) or limited to the object
    level (keeping the statement and theory levels fixed).

    In this paper we develop a general methodology for extending MKM
    representation formats at the statement level. We show the utility
    (and indeed necessity) of statement-level extensions by
    redesigning the OMDoc format into a minimal, regular core language
    (strict OMDoc) and an extension (pragmatic OMDoc) that maps into
    strict OMDoc.",
  paper = "Horo12.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Horo15,
  author = "Horozal, Fulya and Rabe, Florian",
  title = {{Formal Logic Definitions for Interchange Languages}},
  journal = "LNCS",
  volume = "9150",
  year = "2015",
  abstract =
    "System integration often requires standardized interchange
    languages, via which systems can exchange mathematical
    knowledge. Major examples are the MathML-based markup languages
    and TPTP. However, these languages standardize only the syntax of
    the exchanged knowledge, which is insufficient when the involved
    logics are complex or numerous. Logical frameworks, on the other
    hand, allow representing the logics themselves (and are thus aware
    of the semantics), but they abstract from the concrete syntax.

    Maybe surprisingly, until recently, state-of-the-art logical
    frameworks were not quite able to adequately represent logics
    commonly used in formal systems. Using a recent extension of the
    logical framework LF, we show how to give concise formal
    definitions of the logics used in TPTP. We can also formally
    define translations and combinations between the various TPTP
    logics. This allows us to build sematics-aware tool support such
    as type-checking TPTP content.

    While our presentation focuses on the current TPTP logics, our
```

```
    approach can be easily extended to other logics and interchange
    languages. In particular, our logic representations can be used
    with both TPTP and MathML. Thus, a single definition of the
    semantics can be used with either interchange syntax.",
  paper = "Horo15.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Howe87,
  author = "Howe, Douglas J.",
  title = {{The Computational Behaviour of Girard's Paradox}},
  institution = "Cornell University",
  year = "1987",
  link = "\url{https://ecommons.cornell.edu/handle/1813/6660}",
  number = "TR 87-820",
  abstract =
    "In their paper ``Type'' Is Not a Type, Meyer and Reinhold argued that
    serious pathologies can result when a type of all types is added to a
    programing language with dependent types. Central to their argument is
    the claim that by following the proof of Girard's paradox it is
    possible to construct in their calculus $\lambda^{\tau \tau}$ a term
    having a fixed-point property. Because of the tremendous amount of
    formal detail involved, they were unable to establish this claim. We
    have made use of the Nuprl proof development system in constructing a
    formal proof of Girard's paradox and analysing the resulting term. We
    can show that the term does not have the desired fixed-point property,
    but does have a weaker form of it that is sufficient to establish some
    of the results of Meyer and Reinhold. We believe that the method used
    here is in itself of some interest, representing a new kind of
    application of a computer to a problem in symbolic logic."
}
```

———-

— **axiom.bib** —

```
@misc{Howe69,
  author = "Howard, W.H.",
  title = {{The Formulae-as-Types Notion of Construction}},
  year = "1969",
  link = "\url{http://www.dcc.fc.up.pt/~acm/howard.pdf}",
  abstract =
    "The following consists of notes which were privately circulated in
    1969. Since they have been referred to a few times in the literature,
    it seems worth while to publish them. They have been rearranged for
    easier reading, and some inessential corrections have been made.

    The ultimate goal was to develop a notion of construction suitable for
    the interpretation of intuitionistic mathematics. The notion of
```

```
    construction developed in the notes is certainly too crude for that,
    so the use of the word construction is not very appropriate. However,
    the terminology has been kept in order to preserve the original title
    and also to preserve the character of the notes. The title has a
    second defect; namely, a type should be regarded as a abstract object
    whereas a formula is the name of a type.

    In Part I the ideas are illustrated for the intuitionistic
    propositional calculus and in Part II (page 6) they are applied to
    Heyting arithmetic.",
  paper = "Howe69.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Hrba99,
  author = "Hrbacek, Karel and Jech, Thomas",
  title = {{Introduction to Set Theory}},
  publisher = "Marcel Dekker, Inc",
  year = "1999",
  isbn = "0-8247-7915-0",
  paper = "Hrba99.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Huda92,
  author = "Hudak, Paul and Jones, Simon Peyton and Wadler, Philip and
            Boutel, Brian and Fairbairn, Jon and Fasel, Joseph and
            Guzman, Maria M. and Hammond, Kevin and Hughes, John and
            Johnsson, Thomas and Kieburtz, Dick and Nikhil, Rishiyur and
            Patrain, Will and Peterson, John",
  title = {{Report on the Programming Language Haskell, a non-strict
           functional language version 1.2}},
  journal = "ACM SIGPLAN Notices",
  volume = "27",
  number = "5",
  year = "1992",
  pages = "1-164",
  abstract =
    "Some half dozen persons have written technically on combinatory
    logic, and most of these, including ourselves, have published
    something erroneous. Since some of our fellow sinners are among the
    most careful and competent logicians on the contemporary scene, we
    regard this as evidence that the subject is refractory. Thus fullness
    of exposition is necessory for accurary; and excessive condensation
    would be false economy here, even more than it is ordinarily."
}
```

———-

— **axiom.bib** —

```
@misc{Huda99,
  author = "Hudak, Paul and Peterson, John and Fasel, Joseph H.",
  title = {{A Gentle Introduction to Haskell 98}},
  year = "1999",
  link = "\url{https://www.haskell.org/tutorial/haskell-98-tutorial.pdf}",
  paper = "Huda99.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Huds19,
  author = "Hudson, Andrew Dana",
  title = {{A Priest, a Rabbi, and a Robot Walk Into a Bar}},
  year = "2019",
  link =
  "\url{https://slate.com/technology/future-tense-fiction-priest-rabbi-robot-bar-andrew-hudson.html}",
  abstract =
    "A new short story looks at how artificial intelligence
     could support, and distort, faith."
}
```

———-

— **axiom.bib** —

```
@book{Huet91,
  author = "Huet, Gerard and Plotkin, G.",
  title = {{Logical Frameworks}},
  publisher = "Cambridge University",
  year = "1991"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Huet00,
  author = "Huet, Gerard and Saibi, Amokrane",
  title = {{Constructive Category Theory}},
  booktitle = "Proof, Language, and Interaction: Essays in Honour of
               Robin Milner",
  publisher = "MIT Press",
  pages = "235-275",
  year = "2000"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Hugh12,
  author = "Hughes, John",
  title = {{Monads and all that}},
  year = "2012",
  link = "\url{https://www.cs.uoregon.edu/research/summerschool/summer12/curriculum.html}",
  keywords = "DONE"

}
```

———-

— **axiom.bib** —

```
@misc{Hugh19,
  author = "Hughes, John",
  title = {{How to Specify it!}},
  year = "2019",
  link = "\url{https://www.dropbox.com/s/tx2b84kae4bw1p4/paper.pdf}",
  abstract =
    "Property-based testing tools test software against a
    specification, rather than a set of examples. This tutorial paper
    presents five generic approaches to writing such specifications
    (for purely functional code). We discuss costs, benefits, and
    bug-finding power of each approach, with reference to a simple
    example with eight buggy variants. The lessons learned should help
    the reader to develope effective property-based tests in the future.",
  paper = "Hugh19.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@techreport{Hutt96,
  author = "Hutton, Graham and Meijer, Erik",
  title = {{Monadic Parser Combinators}},
  type = "technical report",
  institution = "University of Nottingham",
  number = "MOTTCS-TR-96-4",
  year = "1996",
  link = "\url{http://raw.githubusercontent.com/drewc/smug/master/doc/monparsing.org}",
  abstract =
    "In functional programming, a popular approach to building
    recursive descent parsers is to model parsers as functions, and to
```

```
    define higher-order functions (or combinators) that implement
    grammar constructions such as sequencing, choice, and
    repetition. Such parsers form an instance of a monad, an algebraic
    structure from mathematics that has proved useful for addressing a
    number of computational problems. The purpose of this artice is to
    provide a step-by-step tutorial on the monadic approach to
    building functional parsers, and to explain some of the benefits
    that result from exploiting monads. No prior knowledge of parser
    combinators or of monads is assumed. Indeed, this article can also
    be viewed as a first introduction to the use of monads in
    programming.",
  paper = "Hutt96.txt"
}
```

———-

— **axiom.bib** —

```
@article{Hutt99,
  author = "Hutton, Graham",
  title = {{A Tutorial on the Universality and Expressiveness of
           Fold}},
  journal = "J. Functional Programming",
  volume = "9",
  number = "4",
  pages = "355-372",
  year = "1999",
  abstract =
    "In functional programming, {\sl fold} is a standard operator that
    encapsulates a simple pattern of recursion for processing
    lists. This article is a tutorial on two key aspects of the fold
    operator for lists. First of all, we emphasize the use of the
    universal property of fold both as a proof principle that avoids
    the need for inductive proofs, and as a definition principle that
    guides the transformation of recursive functions into definitions
    using fold. Secondly, we show that even though the pattern of
    recursion encapsulated by fold is simple, in a language with
    tuples and functions as first-class values the fold operator has
    greater expressive power than might first be expected.",
  paper = "Hutt99.pdf",
  keywords = "printed"
}
```

———-

1.2.9 I

— **axiom.bib** —

```
@book{IBMx78,
  author = "IBM",
```

```
  title = {{LISP/370 Program Description / Operations Manual}},
  publisher = "IBM Research",
  year = "1978",
  comment = "SH20-2076-0"
}
```

———-

— **axiom.bib** —

```
@article{Ianc12,
  author = "Iancu, Mihnea and Rabe, Florian",
  title = {{Management of Change in Declarative Languages}},
  journal = "LNCS",
  volume = "7362",
  year = "2012",
  abstract =
    "Due to the high degree of interconnectedness of formal
    mathematical statements and theories, human authors often have
    difficulties anticipating and tracking the effects of a change in
    large bodies of symbolic mathematical knowledge. Therefore, the
    automation of change management is desirable. But while computers
    can in principle detect and propagate changes automatically, this
    process must take the semantics of the underlying mathematical
    formalism into account. Therefore, concrete management of change
    solutions are difficult to realize.

    The MMT language was designed as a generic declarative language
    that captures universal structural features while avoiding a
    commitment to a particular formalism. Therefore, it provides a
    promising framework for the systematic study of changes in
    declarative languages. We leverage this framework by providing a
    generic change management solution at the MMT level, which can be
    instantiated for arbitrary specific languages.",
  paper = "Ianc12.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Inne19,
  author = "Innes, Sean and Uu, Nicolas",
  title = {{Tic Tak Types}},
  booktitle = "Int. Workshop on Type Driven Development",
  publisher = "ACM",
  year = "2019",
  abstract =
    "Tic-Tac-Toe is a simple, familiar, classic game enjoyed by
    many. This pearl is designed to give a flavour of the world of
    dependent types to the uninitiated functional programmer. We cover
    a journey from Tic-Tak-Terrible implementations in the harsh world
```

```
    of virtually untyped {\sl Strings}, through the safe haven of
    vectors that know their own length, and into a Tic-Tac-Titanium
    version that is too strongly typed for its own good. Along the way
    we discover something we knew all along: types are great, but in
    moderation. This lesson is quickly put to use in a more complex
    recursive version.",
  paper = "Inne19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Ioak00,
  author = "Ioakimidis, Nikolaos I.",
  title = {{Derivation of Feasibility Conditions in Engineering
            Problems under Parametric Inequality Constraints with
    Classical Fourier Elimination}},
  journal = "Int. J. Numerical Methods Eng.",
  volume = "48",
  number = "11",
  pages = "1583-1599",
  year = "2000",
  abstract =
    "Fourier (or Motzkin or even Fourier-Motzkin) elimiation is the
    classical and equally old analogue of Gaussian elimination for the
    solution of linear equations in the case of linear
    inequalities. Here this approach and two standard improvements are
    applied to two engineering problems (involving numerical
    integration in fracture mechanics as well as finite differences in
    heat transfer in the parametric case) with linear inequality
    constraints. The results obtained by a solver of systems of
    inequalities concern the feasibility conditions (quantifier-free
    formulae), so that the satisfaction of the original system of
    linear inequality constraints can be possible. We use the computer
    algebra system Maple V and a related elementary procedure for
    Fourier elimination. The results constitute an extension of
    already available applications of computer algebra software to the
    classical approximate-numerical methods traditionally employed in
    enginerring, and are also related to computational elimination
    techniques in computer algebra and applied logic.",
  paper = "Ioak00.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Isab11,
  author = "Unknown",
```

```
  title = {{Isabelle}},
  link = "\url{http://www.cl.cam.ac.uk/research/hvg/Isabelle/index.html}",
  year = "2011"
}
```

———-

1.2.10 J

— **axiom.bib** —

```
@article{Jaff93,
  author = "Jaffe, Arthur and Quinn, Frank",
  title = {{``Theoretical Mathematics'': Towards a Cultural Synthesis
            of Mathematics and Theoretical Physics}},
  journal = "Bull. of the American Mathematical Society",
  volume = "29",
  number = "1",
  pages = "1-13",
  year = "1993",
  abstract =
    "Is specualative mathematics dangerous? Recent interactions
    between physics and mathematics pose the question with some force:
    traditional mathematical norms discourage speculation, but it is
    the fabric of theoretical physics. In practice there can be
    benefits, but there can also be unpleasant and destructive
    consequences. Serious caution is required, and the issue should be
    considered before, rather than after, obvious damage occurs. With
    the hazards carefully in mind, we propose a framework that should
    allow a healthy and positive role for speculation.",
  paper = "Jaff92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Jame81,
  author = "James, G. and Kerber, A.",
  title = {{The Representation Theory of the Symmetric Group}},
  journal = "Encycl. of Math. and its Appl.",
  volume = "16",
  algebra = "\newline\refto{package REP1 RepresentationPackage1}",
  publisher = "Cambr. Univ. Press",
  year = "1981"
}
```

———-

— **axiom.bib** —

```
@book{Jamm66,
  author = "Jammer, Max",
  title = {{The Conceptual Development of Quantum Mechanics}},
  year = "1996",
  publisher = "McGraw-Hill"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Jask08,
  author = "Jaskelioff, Mauro and Ghani, Neil and Hutton, Graham",
  title = {{Modularity and Implementation of Mathematical Operational
            Semantics}},
  publisher = "Elsevier",
  year = "2008",
  link = "\url{http://www.cs.nott.ac.uk/~pszgmh/modular.pdf}",
  abstract =
    "Structural operational semantics is a popular technique for
    specifying the meaning of programs by means of inductive
    clauses. One seeks syntactic restrictions on those clauses so that
    the resulting operational semantics is well-behaved. This approach
    is simple and concrete but it has some drawbacks. Turi pioneered a
    more abstract categorical treatment based upon the idea that
    operational semantics is essentially a distribution of syntax over
    behaviour. In this article we take Turi's approach in two new
    directions. Firstly, we show how to write operational semantics as
    modular components and how to combine such components to specify
    complete languages. Secondly, we show how the categorical nature
    of Turi's operational semantics makes it ideal for implementation
    in a functional programming language such as Haskell.",
  paper = "Jask08.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Jebe93,
  author = "Jebelean, Tudor",
  title = {{Improving the Multiprecision Euclidean Algorithm}},
  booktitle = "DISCO 1993",
  year = "1993",
  pages = "45-58",
  publisher = "Springer",
  abstract =
    "We improve the implementation of Lehmer-Euclid algorithm for
    multiprecision integer GCD computations by partial consequence
    computation on pairs of double digits, enhanced conditions for
    exiting the partia consequence computatin, and approximative GCD
    computation. The combined effect of these improvements is an
```

```
    experimentally measured speed-up by a factor of 2 over the
    currently used implementation.",
  paper = "Jebe93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Jedy13,
  author = "Jedynak, Wojciech and Biernacka, Malgorzata and
            Biernacki, Dariusz",
  title = {{An Operational Foundation for the Tactic Language of Coq}},
  booktitle = "Proc. 15th Symp. on Principles and Practices of
               Declarative Programming",
  publisher = "ACM",
  pages = "25-36",
  year = "2013",
  isbn = "978-1-4503-2154-9",
  abstract =
    "We introduce a semantic toolbox for Ltac, the tactic language of
    the popular Coq proof assistant. We present three formats of
    operational semantics, each of which has its use in the practice
    of tactic programming: a big-step specification in the form of
    natural semantics, a model of implementation in the form of an
    abstract machine, and a small-step characterization of computation
    in the form of reduction semantics. The three semantics are
    provably equivalent and have been obtained via off-the-shelf
    derivation techniques of the functional correspondence and the
    syntactic correspondence. We also give examples of Ltac programs
    and discuss some of th eissues that the formal semantics help to
    clarify.

    With this work we hope to enhance the operational understanding of
    Ltac as well as to set up a framework to reason about Coq scripts
    and to build tools supporting tactic programming based on rigorous
    semantics.",
  paper = "Jedy13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Jeff10,
  author = "Jeffrey, David J. and Rich, Albert D.",
  title = {{Reducing Expression Size Using Rule-Based Integration}},
  journal = "LNCS",
  volume = "6167",
  year = "2010",
  abstract =
```

```
    "This paper describes continuing progress on the development of a
    repository of transformation rules relevant to indefinite
    integration. The methodology, however, is not restricted to
    integration. Several optimization goals are being pursued,
    including achieving the best form for the output, reducing the
    size of the repository while retaining its scope, and minimizing
    the number of steps required for the evaluation process. New
    optimizations for expression size are presented.",
  paper = "Jeff10.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@book{Jeff81,
  author = "Jeffrey, Richard",
  title = {{Formal Logic: Its Scope and Limits}},
  publisher = "McGraw-Hill",
  year = "1981",
  isbn = "0-07-032321-6",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Jenk18,
  author = "Jenkins, Kris",
  title = {{Communicating with Types}},
  year = "2018",
  link = "\url{https://vimeo.com/302682323}",
  abstract =
    "Modern type systems have come a long way from the days of C and
    Java. Far from being nit-pickers that berate us for making
    mistakes, type systems like the ones found in Haskell, PureScript
    and Elm form a language in themselves. A language for expressing
    high-level ideas about our software to our colleagues and to the
    computer. A design language. In this talk, we'll take a look at
    the way the right kind of type signatures let us talk about
    software. We'll survey how to state our assumptions about the
    domain we're coding in and how each part fits together. We'll show
    how it can highlight problems, and surface opportunities for
    reuse. And most importantly, we'll see how types can help you
    communicate to your coworkers, and to future maintainers, with
    little effort. You've probably heard the phrase ``programs are
    meant to be read by humans and only incidentally for computers to
    execute''. Come and see how a modern type system is about
    communicating ideas to humans, and only incidentally about proving
    correctness.",
```

```
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Jenk75,
  author = "Jenks, Richard D.",
  title = {{Course Outline: Yale University, New Haven}},
  journal = "SIGSAM Bulletin",
  volume = "9",
  number = "3",
  pages = "9-10",
  publisher = "ACM",
  year = "1975",
  paper = "Jenk75.pdf",
  keywords = "axiomref, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Jenk85,
  author = "Jenks, R. and Trager, B. and Watt, S.M. and Sutor, R.S.",
  title = {{Scratchpad II Programming Language Manual}},
  year = "1985",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Jens75,
  author = "Jensen, Kathleen and Wirth, Niklaus",
  title = {{PASCAL User Manual and Report}},
  publisher = "Springer-Verlag",
  year = "1975",
  isbn = "0-387-90144-2",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Jone80,
  author = "Jones, Neil D. and Schmidt, David A.",
  title = {{Compiler Generation from Denotational Semantics}},
```

```
  journal = "LNCS",
  volume = "54",
  pages = "70-93",
  year = "1980",
  abstract =
    "A methodology is described for generating provably correct
    compilers from denotational definitions of programming
    languages. An application is given to produce compilers into STM
    code (an STM or state transition machine is a flow-chart-like
    program, low-level enough to be translated into efficient code on
    conventional computers). First, a compiler $\phi:LAMC\rightarrow
    STM$ from a lambda calculus dialect is defined. Any denotational
    defintion $\Delta$ of language $L$ defines a map
    $\over{\rightarrow}{\Delta}:L\rightarrow LAMC$, so
    $\over{\rightarrow}{\Delta}\circ \phi$ compiles $L$ into STM
    code. Correctness follows from the correctness of $\phi$.

    The algebraic framework of Morris, ADJ, etc. is used. The set of
    STMs is given an algebraic structure so any
    $\over{\rightarrow}{\Delta} \circ \phi$ may be specified by giving
    a derived operator on STM for each syntax rule of $L$.

    This approach yields quite redundant object programs, so the paper
    ends by describing two flow analytic optimization methods. The
    first analyzes an already-produced STM to obtain information about
    its runtime behaviour which is used to optimize the STM. The
    second analyzer the generated compiling scheme to determine
    runtime properties of object programs in general which a compiler
    can use to produce less redundant STMs.",
  paper = "Jone80.pdf"
}
```

———-

— axiom.bib —

```
@article{Jone80a,
  author = "Jones, Neil D. and Madsen, Michael",
  title = {{Attribute-Influenced LR Parsing}},
  journal = "LNCS",
  volume = "54",
  pages = "393-407",
  year = "1980",
  abstract =
    "Methods are described which make it possible, when given an
    arbitrary attribute grammar (or AG),
    \begin{enumerate}
    \item to analyze the AG to determine which of its attributes may
    be computed during LR parsing,
    \item to augment the parser with instructions and data structures
    to compute many attributes during parsing,
    \item to use attribute values to assist the parsing process
    (e.g. to use symbol table information to decide whether P(X) is an
```

```
    array element or a function call).
    \end{enumerate}",
  paper = "Jone80a.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Jone87,
  author = "Jones, Simon Peyton",
  title = {{The Implementation of Functional Programming Languages}},
  publisher = "Simon and Schuster",
  year = "1987",
  isbn = "0-13-453333-X",
  paper = "Jone87.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Jone93,
  author = "Jones, Neil D. and Gomard, Carsten K. and Sestoft, Peter",
  title = {{Partial Evaluation and Automatic Program Generation}},
  year = "1993",
  publisher = "Prentice Hall",
  link = "\url{www.itu.di/~setoft/pebook/jonesgomardsestoft-letter.pdf}",
  paper = "Jone93.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Jone00,
  author = "Jones, Simon Peyton",
  title = {{Implementing Functional Languages: A Tutorial}},
  publisher = "University of Glasgow",
  year = "2000",
  link = "\url{https://www.microsoft.com/en-us/research/wp-content/uploads/1992/01/student.pdf}",
  paper = "Jone00.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Joua90,
  author = "Jouannaud, Jean-Pierre and Kirchner, Claude",
  title = {{Solving Equations in Abstract Algebras: A Rule-based Survey of
```

```
          Unification}},
  year = "1990",
  publisher = "Universite do Paris-Sud"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Joua90a,
  author = "Jouannaud, Jean-Pierre",
  title = {{Completion modulo Associativity, Commutativity and
           Identity (ACI)}},
  booktitle = "DISCO 1990",
  year = "1990",
  pages = "111-120",
  publisher = "Springer",
  abstract =
    "Rewriting with associativity, commutativity and identity has been
    an open problem for along time. In a recent paper [PBW89], Baird,
    Peterson and Wilkerson introduced the notion of constrained
    rewriting, to avoid the problem of non-termination inherent to the
    use of identities. We build up on this idea in two ways: by giving
    a complete set of rules for completion modulo these axioms; by
    showing how to build appropriate orderings for proving termination
    of constrained rewriting modulo associativity, commutativity and
    identity.",
  paper = "Joua90a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Joua91,
  author = "Jouannaud, Jean Pierre and Okada, Mitsuhiro",
  title = {{A Computation Model for Executable Higher-order Algebraic
           Specification Languages}},
  booktitle = "Symposium on Logic in Computer Science",
  pages = "350-361",
  isbn = "081862230X",
  year = "1991",
  abstract =
    "The combination of (polymorphically) typed lambda-calculi with
    first-order as well as higher-order rewrite rules is considered. The
    need of such a combination for exploiting the benefits of
    algebraically defined data types within functional programming is
    demonstrated. A general modularity result, which allows as particular
    cases primitive recursive functionals of higher types, transfinite
    recursion of higher types, and inheritance for all types, is
    proved. The class of languages considered is first defined, and it is
```

```
    shown how to reduce the Church-Rosser and termination (also called
    strong normalization) properties of an algebraic functional language
    to a so-called principal lemma whose proof depends on the property to
    be proved and on the language considered. The proof of the principal
    lemma is then sketched for various languages. The results allows
    higher order rules defining the higher-order constants by a certain
    generalization of primitive recursion. A prototype of such primitive
    recursive definitions is provided by the definition of the map
    function for lists.",
  paper = "Joua91.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Jung18,
  author = "Jung, Ralf and Jourdan, Jacques-Henri and
            Krebbers, Robbert and Dreyer, Derek",
  title = {{RustBelt: Securing the Foundations of the Rust Programming
            Language}},
  booktitle = "POPL '18",
  publisher = "ACM",
  year = "2018",
  abstract =
    "Rust is a new systems programming language that promises to
    overcome the seemingly fundamental tradeoff between high-level
    safety guarantees and low-level control over resource
    management. Unfortunately, none of Rust's safety claims have been
    formally proven, and there is good reason to question whether they
    actually hold. Specifically, Rust employs a string,
    ownership-based, type system, but then extends the expressive
    power of this core type system through libraries that internally
    use unsafe features. In this paper, we give the first formal (and
    machine-checked) safety proof for a language representing a
    realistic subset of Rust. Our proof is extensible in the sense
    that, for each new Rust library that uses unsafe features, we can
    say what verification condition it must satisfy in order for it to
    be deemed a safe extension to the language. We have carried out
    this verification for some of the most important libraries that
    are used throughout the Rust ecosystem.",
  paper = "Jung18.pdf",
  keywords = "printed"

}
```

———-

1.2.11 K

— **axiom.bib** —

```
@misc{Kaha04,
  author = "Kahan, William",
  title = {{On the Cost of Floating-Point Computation Without
            Extra-Precise Arithmetic}},
  year = "2004",
  link = "\url{https://people.eecs.berkeley.edu/~wkahan/Qdrtcs.pdf}",
  abstract =
    "Current benchmarks give the impression that computation costs no
    more than the time consumed, and perhaps the memory occupied, only
    because we can measure these so easily. What about the costs of
    maintenance (when hardware or operating systems change), of
    development (algorithm design, tests, proofs of correctness), and
    of misleading results? Solving a quadratic equation provides a
    relatively easily understood case study of the way all costs get
    inflated when arithmetic precision rather higher than the
    precision of the data and the accuracy demanded of results is
    unavailable or, worse, unusable because of lapses in the design
    and/or implementation of widely used programming languages. Then
    costs are inflated by the trickery required to devise portable
    programs that compute roots at least about as accurately as the
    data deserve, and to prove that they are that accurate, and to
    test them. This trickery will be illustrated by a MATLAB program
    designed to get results of high quality from diverse versions of
    MATLAB on the two most popular kinds of hardware, PCs and
    Macs. Included is a test program and some of its results. Only a
    small part of the program needs higher precision arithmetic not
    too slow. There it would cost far less than the consequences of
    doing without.",
  paper = "Kaha04.pdf"
}
```

———-

— axiom.bib —

```
@article{Kahr94,
  author = "Kahrs, S. and Sannella, D. and Tarlecki, A.",
  title = {{Interfaces and Extended ML}},
  journal = "SIGPLAN Notices",
  volume = "29",
  number = "8",
  pages = "111-118",
  year = "1994",
  abstract =
    "This is a position paper giving our views on the uses and makeup
    of module interfaces. The position espoused is inspired by our
    work on the Extended ML (EML) formal software development
    framework and by ideas in the algebraic foundations of
    specification and formal development. The present state of
    interfaces in EML is outlined and set in the context of plans for
    a more general EML-like framework with axioms in interfaces taken
    from an arbitrary logical system formulated as an
    {\sl institution}. Some more speculative plans are sketched
```

```
    concerning the simultaneous use of multiple institutions in
    specification and development.",
  paper = "Kahr94.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Kahr95,
  author = "Kahrs, Stefan",
  title = {{On the Static Analysis of Extended ML}},
  type = "technical report",
  institution = "Lab for Foudations of Comp Sci. Univ. Edinburgh",
  number = "Research Note",
  year = "1995",
  abstract =
    "This is a short note describing differences in static analysis of
    EML, as defined in [KST94] and SML, as defined in [MTH90] and
    [MT91]. It is intended for use by people who are building an EML
    parser/type-checker by modifying an existing SML compiler.",
  paper = "Kahr95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Kahr97,
  author = "Kahrs, Stefan and Sannella, Donald and Tarlecki, Andrzej",
  title = {{The Definition of Extended ML}},
  journal = "Theoretical Computer Science",
  volume = "173",
  pages = "445-484",
  year = "1997",
  abstract =
    "This document formally defines the syntax and semantics of the
    Extended ML language. It is based on the published semantics of
    Standard ML in an attempt to ensure compatibility between the two
    languages.",
  paper = "Kahr97.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Kahr98,
```

```
  author = "Kahrs, Stefan and Sannella, Donald",
  title = {{Reflections on the Design of a Specification Language}},
  journal = "LNCS",
  volume = "1382",
  pages = "154-170",
  year = "1998",
  abstract =
    "We reflect on our experiences from work on the design and
    semantic underpinnings of Extended ML, a specification language
    which supports the specification and formal development of
    Standard ML programs. Our aim is to isolate problems and issues
    that are intrinsic to the general enterprise of designing a
    specification language for use with a given programming language.
    Consequently the lessons learned go far beyond our original aim of
    designing a specification language for ML.",
  paper = "Kahr98.pdf",
  keywords = "printed, DONE"
}
```

———-

— axiom.bib —

```
@misc{Kali14,
  author = "Kaliszyk, Cezary and Urban, Josef",
  title = {{Learning-Assisted Automated Reasoning with Flyspeck}},
  comment = "arXiv:1211.7012v3",
  year = "2014",
  abstract =
    "The considerable mathematical knowledge encoded by the Flyspeck
    project is combined with external automated theorem provers (ATPs)
    and machine-learning premise selection methods trained on the
    Flyspeck proofs, producing an AI system capable of proving a wide
    range of mathematical conjectures automatically. The performance
    of this architecture is evaluated in a bootstrapping scenario
    emulating the development of Flyspeck from axioms to the last
    theorem, each time using only the previous theorems and proofs. It
    is shown that 39\% of the 14185 theorems could be proved in a
    push-button mode (without any high-level advice and user
    interaction) in 30 seconds of real time on a fourteen CPU
    workstation.

    The necessary work involves: (i) an implementation of sound
    translations of the HOL Light logic to ATP formalisms: untyped
    first-order, polymorphic typed first-order, and typed first-order,
    (ii) export of the dependency information from HOL Light and ATP
    proofs for the machine learners, and (iii) choice of suitable
    representations and methods for learning from previous proofs, and
    their integration as advisors with HOL Light. This work is
    described and discussed here, and an initial analysis of the body
    of proofs that were found fully automatically is provided.",
  paper = "Kali14.pdf",
  keywords = "printed"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Kalo18,
  author = "Kalorkoti, K.",
  title = {{Introduction to Computer Algebra}},
  comment = "Course Notes",
  year = "2018",
  keywords = "shelf, printed"
}
```

———-

— **axiom.bib** —

```
@incollection{Kalt83a,
  author = "Kaltofen, E.",
  title = {{Factorization of Polynomials}},
  booktitle = "Computer Algebra - Symbolic and Algebraic Computation",
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "95-113",
  abstract =
    "Algorithms for factoring polynomials in one or more variables over
    various coefficient domains are discussed. Special emphasis is given
    to finite fields, the integers, or algebraic extensions of the
    rationals, and to multivariate polynomials with integral coefficients.
    In particular, various squarefree decomposition algorithms and Hensel
    lifting techniques are analyzed. An attempt is made to establish a
    complete historic trace for todays methods. The exponential worst
    case complexity nature of these algorithms receives attention.",
  paper = "Kalt83a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kalt84,
  author = "Kaltofen, E.",
  title = {{A Note on the Risch Differential Equation}},
  booktitle = "International Sympoium on Symbolic and Algebraic
               Manipulation",
  pages = "359-366",
  publisher = "Springer",
  year = "1984",
  comment = "LNCS 174",
```

```
  abstract =
    "This paper relates to the technique of integrating a function in
    a purely transcendental regular elementary Liouville extension by
    prescribing degree bounds for the transcendentals and then solving
    linear systems over the constants. The problem of finding such
    bounds explicitly remains yet to be solved due to the so-called
    third possibilities in the estimates for the degrees given in
    R. Risch's original algorithm.

    We prove that in the basis case in which we have only exponentials
    of rational functions, the bounds arising from the third
    possibilities are again degree bounds of the inputs. This result
    provides an algorithm for solving the differential equation
    $y^\prime =f^\prime y= g$ in $y$ where $f$, $g$, and $y$ are
    rational functions over an arbitrary constant field. This new
    algorithm can be regarded as a direct generalization of the
    algorithm by E. Horowitz for computing the rational part of the
    integral of a rational function (i.e. $f^\prime = 0$), though its
    correctness proof is quite different.",
  paper = "Kalt84.pdf"
}
```

———-

— axiom.bib —

```
@inproceedings{Kalt85f,
  author = "Kaltofen, Erich and Rolletschek, Heinrich",
  title = {{Arithmetic in Quadratic Fields with Unique Factorization}},
  booktitle = "Research Contributions from the Euro. Conf. on Comp. Alg.",
  series = "Lecture Notes in Computer Science Volume 204",
  volume = "2",
  pages = "279-288",
  year = "1985",
  isbn = "0-387-15983-5 (vol. 1),0-387-15984-3 (vol. 2)",
  abstract =
    "In a quadratic field $\mathbb{Q}(\sqrt{D})$, $D$ a squarefree
    integer, with class number 1 any algebraic integer can be
    decomposed uniquely into primes but for only 21 domains Euclidean
    algorithms are known. We prove that for $D \le -19$ even remainder
    sequences with possibly non-decreasing norms cannot determine the
    GCD of arbitrary inputs. We then show how to compute the greatest
    common divisor of the algebraic integers in any fixed
    $\mathbb{Q}(\sqrt{D})$ with class number 1 in $O(S^2)$ binary
    steps where $S$ is the number of bits needed to encode the
    inputs. We also prove that in any domain the computation of the
    prime factorization of an algebraic integer can be reduced in
    polynomial-time to factoring its norm into rational primes. Our
    reduction is based on a constructive version of a theorem by
    A. Thue. Finally we present another GCD algorithm for complex
    quadratic fields based on a short lattice vector construction.",
  paper = "Kalt85f.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt99a,
  author = "Kaltofen, E. and Monagan, M.",
  title = {{On the Genericity of the Modular Polynomial {GCD} Algorithm}},
  booktitle = "Proc. 1999 Internat. Symp. Symbolic Algebraic Comput.",
  year = "1999",
  pages = "59--66",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/99/KaMo99.pdf}",
  paper = "Kalt99a.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Kama05,
  author = "Kamareddine, Fairouz and Laan, Twan and Nederpelt, Rob",
  title = {{A Modern Perspective on Type Theory}},
  comment = "Applied Logic Series 29",
  publisher = "Kluwer Academic Publishers",
  isbn = "1-4020-2335-9",
  year = "2005",
  paper = "Kama05.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Kami12,
  author = "Kaminski, Paul",
  title = {{The Role of Autonomy in DoD Systems}},
  type = "Task Force Report",
  institution = "Defense Science Board, Dept. of Defense",
  year = "2012",
  link = "\url{https://fas.org/irp/agency/dod/dsb/autonomy.pdf}",
  comment = "verification, validation, and trust"
}
```

———-

— **axiom.bib** —

```
@techreport{Kane90,
  author = "Kanellakis, Paris C. and Mairson, Harry G. and Mitchell, John C.",
  title = {{Unification and ML Type Reconstruction}},
  link = "\url{ftp://ftp.cs.brown.edu/pub/techreports/90/cs90-26.pdf}",
```

```
  institution = "Brown University",
  year = "1990",
  number = "CS-90-26",
  abstract =
    "We study the complexity of type reconstruction for a core fragment of
    ML with lambda abstraction, function application, and the polymorphic
    {\bf let} declaration. We derive exponential upper and lower bounds on
    recognizing the typable core ML expressions. Our primary technical
    tool is unification of succinctly represented type expressions. After
    observing that core ML expressions, of size $n$, can be typed in
    DTIME($s^n$), we exhibit two different families of programs whose
    principal types grow exponentially. We show how to exploit the
    expressiveness of the {\bf let}-polymorphism in these constructions to
    derive lower bounds on deciding typability: one leads naturally to
    NP-hardness and the other to DTIME($2^{n^k}$)-hardness for each integer
    $k\ge 1$. Our generic simulation of any exponential time Turing
    Machine by ML type reconstruction may be viewed as a nonstandard way
    of computing with types. Our worse-case lower bounds stand in contrast
    to practical experience, which suggests that commonly used algorithms
    for type reconstruction do not slow compilation substantially.",
  paper = "Kane90.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kapl80,
  author = "Kaplan, Marc A. and Ullman, Jeffrey D.",
  title = {{A Scheme for the Automatic Inference ov Variable Types}},
  journal = "J. ACM",
  volume = "27",
  number = "1",
  pages = "128-145",
  year = "1980",
  abstract =
    "In this paper an algorithm for the determination of run-time
    types in a programming language requiring no type declarations is
    presented. It is demonstrated that this algorithm is superior to
    other published algorithms in the sense that it produces stronger
    assertions about the set of possible types for variables than do
    other known algorithms. In fact this algorithm is shown to be the
    best possible algorithm from among all those that use the same set
    of primitive operators.",
  paper = "Kapl80.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Katz75,
  author = "Katz, Shmuel and Manna, Zohar",
  title = {{A Closer Look at Termination}},
  journal = "Acta Informatica",
  volume = "5",
  pages = "333-352",
  year = "1975",
  abstract =
    "Several methods for proving that computer programs terminate are
    presented and illustrated. The methods considered involve (a)
    using the 'no-infinitely-descending-chain' property of
    well-founded sets (Floyd's approach), (b) bounding a counter
    associated with each loop ({\sl loop} approach), (c) showing that
    some exit of each loop must be taken ({\sl exit} approach), or (d)
    inducting on the structure of the data domain (Burstall's
    approach). We indicate the relative merit of each method for
    proving termination or non-termination as an integral part of an
    automatic verification system.",
  paper = "Katz75.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Kauf97,
  author = "Kaufman, Matt and Moore, J Strother",
  title = {{An Industrial Strength Theorem Prover for a Logic Based on
            Common Lisp}},
  year = "1997",
  link = "\url{web.eecs.umich.edu/~bchandra/courses/papers/Kaufmann_ACL2.pdf}",
  abstract =
    "ACL2 is a re-implemented extended version of Boyer and Moore's
    Nqthm and Kaufmann's Pc-Nqthm intended for large scale
    verification projects. This paper deals primarily with how we
    scaled up Nqthm's logic to an ``industrial strength'' programming
    language -- namely, a large applicative subset of Common Lisp --
    while preserving the use of total functions within a logic. This
    makes it possible to run formal models efficiently while keeping
    the logic simple. We enumerate many other important features of
    ACL2 and we briefly summarize two industrial applications: a model
    of the Motorola CAP digital signal processing chip and the proof
    of the correctness of the kernel of the floating point division
    algorithm on the AMD$5_k$86 microprocessor by Advanced Micro
    Devices, Inc.",
  paper = "Kauf97.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Kayx19,
  author = "Kay, Alan",
  title = {{Alan Kay Speaks at ATLAS Institute}},
  year = "2019",
  link = "\url{https://www.youtube.com/watch?v=nOrdzDaPYV4}"
}
```

———-

— **axiom.bib** —

```
@misc{Keen14,
  author = "Keenen, David C.",
  title = {{To Dissect a Mockingbird: A Graphical Notation for the
            Lambda Calculus with Animated Reduction}},
  year = "2014",
  link = "\url{http://dkeenan.com/Lambda}",
  abstract =
    "The lambda calculus, and the closely related theory of
    combinators, are important in the foundations of mathematics,
    logic and computer science. This paper provides an informal and
    entertaining introduction by means of an animated graphical
    notation.",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Kerb10,
  author = "Kerber, Manfred",
  title = {{Proofs, Proofs, Proofs, and Proofs}},
  journal = "LNCS",
  volume = "6167",
  year = "2010",
  abstract =
    "In logic there is a clear concept of what constitutes a proof and
    what not. A proof is essentially defined as a finite sequence of
    formulae which are either axioms or derived by proof rules from
    formulae earlier in the sequence. Sociologically, however, it is
    more difficult to say what should constitute a proof and what
    not. In this paper we will look at different forms of proofs and
    try to clarify the concept of proof in the wider meaning of the
    term. This has implications on how proofs should be represented
    formally.",
  paper = "Kerb10.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kfou88,
  author = "Kfoury, A.J. and Tiuryn, J. and Utzyczyn, P.",
  title = {{A Proper Extension of ML with an Effective Type-Assignment}},
  booktitle = "POPL 88",
  year = "1988",
  pages = "58-69",
  abstract =
    "We extend the functional language ML by allowing the recursive calls
    to a function F on the right-hand side of its definition to be at
    different types, all generic instances of the (derived) type of F on
    the left-hand side of its definition. The original definition of ML
    does not allow this feature. This extension does not produce new types
    beyond the usual universal polymorphic types of ML and satisfies the
    properties already enjoyed by ML: the principal-type property and the
    effective type-assignment property.",
  paper = "Kfou88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kfou93,
  author = "Kfoury, A. J. and Tiuryn, J. and Urzyczyn, P.",
  title = {{The Undecidability of the Semi-unification Problem}},
  journal = "Information and Computation",
  volume = "102",
  number = "1",
  year = "1993",
  pages = "83-101",
  abstract =
    "The Semi-Unification Problem (SUP) is a natural generalization of
    both first-order unification and matching. The problem arises in
    various branches of computer science and logic. Although several
    special cases of SUP are known to be decidable, the problem in general
    has been open for several years. We show that SUP in general is
    undecidable, by reducing what we call the ``boundedness problem'' of
    Turing machines to SUP. The undecidability of this boundedness problem
    is established by a technique developed in the mid-1960s to prove
    related results about Turing machines.",
  paper = "Kfou93.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Kicz91,
  author = "Kiczales, Gregor and des Rivieres, Jim and Bobrow, Daniel G.",
  title = {{The Art of the Metaobject Protocol}},
```

```
  publisher = "MIT Press",
  year = "1991",
  isbn = "0-262-61074-4",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kife91,
  author = "Kifer, Michael and Wu, James",
  title = {{A First-order Theory of Types and Polymorphism in Logic
           Programming}},
  booktitle = "Proc Sixth Annual IEEE Symp. on Logic in Comp. Sci.",
  year = "1991",
  pages = "310-321",
  abstract =
    "A logic called typed predicate calculus (TPC) that gives declarative
    meaning to logic programs with type declarations and type inference is
    introduced. The proper interaction between parametric and inclusion
    varieties of polymorphism is achieved through a construct called type
    dependency, which is analogous to implication types but yields more
    natural and succinct specifications. Unlike other proposals where
    typing has extra-logical status, in TPC the notion of type-correctness
    has precise model-theoretic meaning that is independent of any
    specific type-checking or type-inference procedure. Moreover, many
    different approaches to typing that were proposed in the past can be
    studied and compared within the framework of TPC. Another novel
    feature of TPC is its reflexivity with respect to type declarations;
    in TPC, these declarations can be queried the same way as any other
    data. Type reflexivity is useful for browsing knowledge bases and,
    potentially, for debugging logic programs.",
  paper = "Kife91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Kirk89,
  author = "Kirkerud, Bjorn",
  title = {{Object-Oriented Programming With Simula}},
  year = "1989",
  series = "International Computer Science Series",
  publisher = "Addison-Wesley"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kiss18,
  author = "Kiss, Csongor and Eisenbach, Susan and Field, Tony and
            Jones, Simon Peyton",
  title = {{Higher-order Type-level Programming in Haskell}},
  booktitle = "Proc. ACM Programming Languages",
  year = "2018",
  publisher = "ACM",
  abstract =
    "Type family applications in Haskell must be fully saturated. This
    means that all type-level functions have to be first-order,
    leading to code that is both messy and long winded. In tis paper
    we detail an extension to GHC that removes this restriction. We
    augment Haskell's existing type arrow, $\rightarrow$, with an
    unmatchable arrow $\twoheadrightarrow$, that supports partial
    application of type families without compromising soundness. A
    soundness proof is provided. We show how the techniques described
    can lead to substantial code-size reduction (circa 80\%) in the
    type-level logic of commonly-used type-level libraries whilst
    simultaneously improving code quality and readability.",
  paper = "Kiss18.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Klee65,
  author = "Kleene, S.C. and Vesley, R.E.",
  title = {{The Foundations of Intuitionistic Mathematics}},
  publisher = "North-Holland Publishing Company",
  year = "1965",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@techreport{Klop90,
  author = "Klop, J. W.",
  title = {{Term Rewriting Systems}},
  institution = "Stichting Methematisch Centrum",
  year = "1990",
  number = "CS-R9073",
  abstract =
    "Term Rewriting Systems play an important role in various areas, such
    as abstract data type specifications, implementations of functional
    programming languages and automated deduction. In this chapter we
    introduce several of the basic concepts and facts for
    TRSs. Specifically, we discuss Abstract Reduction Systems; general
    Term Rewriting Systems including an account of Knuth-Bendix completion
```

```
    and (E- )unification; orthogonal TRSs and reduction strategies;
    strongly sequential orthogonal TRS. Finally some extended rewrite
    formates are introduced: Conditional TRSs and Combinatory Reduction
    Systems. The emphasis throughout the paper is on providing information
    of a syntactic nature."
}
```

———-

— **axiom.bib** —

```
@misc{Knil12,
  author = "Knill, Oliver",
  title = {{A Multivariate Chinese Remainder Theorem}},
  link = "\url{https://arxiv.org/abs/1206.5114}",
  year = "2012",
  abstract =
    "Using an adaptation of Qin Jiushao's method from the 13th
    century, it is possible to prove that a system of linear modular
    equations $a(i,1)x(i)+a(i,n)x(n)=b(i)\bmod m(i),i=1,\ldots,n$ has
    integer solutions if $m(i)>1$ are pairwise relatively prime and in
    each row, at least one matrix element $a(i,j)$ is relatively prime
    to $m(i)$. The Chinese remainder theorem is a special case, where
    $A$ has only one column.",
  paper = "Knil12.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Knil18,
  author = "Knill, Oliver",
  title = {{Some Fundamental Theorems in Mathematics}},
  link = "\url{https://arxiv.org/pdf/1807.08416.pdf}",
  year = "2018",
  abstract =
    "An expository hitchhikers guide to some theorems in mathematics",
  paper = "Knil18.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Knut71,
  author = "Knuth, Donald",
  title =
    {{The Art of Computer Programming Vol. 2 (Seminumerical Algorithms)}},
  year = "1971",
  publisher = "Addison-Wesley"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Knut87,
  author = "Knuth, Donald E. and Larrabee, Tracy and Roberts, Paul M.",
  title = {{Mathematical Writing}},
  year = "1987",
  link = "\url{jmlr.csail.mit.edu/reviewing-papers/knuth_mathematical_writing.pdf}"
}
```

———-

— **axiom.bib** —

```
@misc{Knut19,
  author = "Knuth, Donald E.",
  title = {{Donald Knuth: Algorithms, Complexity, Life, and The Art of
            Computer Programmig}},
  link = "\url{https://www.youtube.com/watch?v=2Bd8fsXbST8}",
  year = "2019",
  comment = "Lex Fridman AI Podcast"
}
```

———-

— **axiom.bib** —

```
@article{Kohl08a,
  author = "Kohlhase, Michael and Muller, Christine and Rabe, Florian",
  title = {{Notations for Living Mathematical Documents}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
  abstract =
    "Notations are central for understanding mathematical
    discourse. Readers would like to read notations that transport the
    meaning well and prefer notations that are familiar to
    them. Therefore, authors optimze the choice of notations with
    respect to these two criteria, while at the same time trying to
    remain consistent over the document and their own prior
    publications. In print media where notations are fixed at
    publication time, this is an over-constrained problem. In living
    documents notations can be adapted at reading time, taking reader
    preferences into account.

    We present a representational infrastructure for notations in
    living mathematical documents. Mathematical notations can be
    defined declaratively. Author and reader can extensionally define
```

```
    the set of available notation definitions at arbitrary document
    levels, and they can guide the notation selection function via
    intensional annotations.

    We give an abstract specification of notation definitions and the
    flexible rendering algorithms and show their coverage on
    paradigmatic examples. We show how to use this framework to render
    OPENMATH and Content-MathML to Presentation-MathML, but the
    approach extends to arbitrary content and presentation formats. We
    discuss prototypical implementations of all aspects of the
    rendering pipeline.",
  paper = "Kohl08a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kohl13,
  author = "Kohlhase, Michael and Mance, Felix and Rabe, Florian",
  title = {{A Universal Machine for Biform Theory Graphs}},
  journal = "LNCS",
  volume = "7961",
  year = "2013",
  abstract =
    "Broadly speaking, there are two kinds of semantics-aware
    assistant systems for mathematics: proof assistants express the
    semantic in logic and emphasize deduction, and computer algebra
    systems express the semantics in programming languages and
    empahsize computation. Combining the complementary strengths of
    both approaches while mending their complementary weaknesses has
    been an important goal of the mechanized mathematics community for
    some time.

    We pick up on the idea of biform theories and interpret it in the
    MMT/OMDOC framework which introduced the foundations-as-theories
    approach, and can thus represent both logics and programming
    languages as theories. This yields a formal, modular framework of
    biform theory graphs which mixes specifications and implemenations
    sharing the module system and typing information.

    We present automated knowledge management work flows that
    interface to existing specification/programming tools and enable
    an OPENMATH Machine, that operationalizes biform theories,
    evaluating expressions by exhaustively applying the
    implementations of the respective operators. We evaluate the new
    biform framework by adding implementations to the OPENMATH
    standard content dictionaries.",
  paper = "Kohl13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Korn07,
  author = "Kornilowicz, Artur",
  title = {{A Proof of the Jordan Curve Theorem via the Brouwer Fixed
            Point Theorem}},
  journal = "Mechanized Mathematics and Its Applications",
  volume = "6",
  number = "1",
  pages = "33-40",
  year = "2007"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kote16,
  author = "Kotelnikov, Evgenii and Kovacs, Laura and Reger, Giles and
            Voronkov, Andrei",
  title = {{The Vampire and the FOOL}},
  booktitle = "SIGPLAN Conf. on Certified Programs and Proofs",
  year = "2016",
  publisher = "ACM",
  pages = "37-48",
  abstract =
    "This paper presents new features recently implemented in the
    theorem prover Vampire, namely support for first-order logic with
    a first class boolean sort (FOOL) and polymorphic arrays. In
    addition to having a first class boolean sort, FOOL also contains
    if-then-else and let-in expressions. We argue that presented
    extensions facilitate reasoning-based program analysis, both by
    increasing the expressivity of first-order reasoners and by gains
    in efficiency.",
  paper = "Kote16.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Kote18,
  author = "Kotelnikov, Evgenii",
  title = {{Automated Theorem Proving with Extensions of First-Order Logic}},
  school = "Chalmers",
  year = "2018",
  abstract =
    "Automated theorem provers are computer programs that check
    whether a logical conjecture follows from a set of logical
    statements. The conjecture and the statements are expressed in the
    language of some formal logic, such as first-order logic. Theorem
```

```
    provers for first-order logic have been used for automation in
    proof assistants, verification of programs, static analysis of
    networks, and other purposes. However, the efficient usage of
    these provers remains challenging. One of the challenges is the
    complexity of translating domain problems to first-order
    logic. Not only can such translation be cumbersome due to semantic
    differences between the domain and the logic, but it might
    inadvertently result in problems that provers cannot easily handle.

    The work presented in the thesis addresses this challenge by
    developing an extension of first-order logic named FOOL. FOOL
    contains syntactical features of programming languages and more
    expressive logics, is friendly for translation of problems from
    various domains, and can be efficiently suported by existing
    theorem provers. We describe the syntax and semantics of FOOL and
    present a simple translation from FOOL to plain first-order
    logic. We describe an efficient clausal normal form transformation
    algorithm for FOOL and based on it implement a support for FOOL in
    the Vampire theorem prover. We illustrate the efficient use of
    FOOL for program verification by describing a concise encoding of
    next state relations of imperative programs in FOOL. We show a
    usage of features of FOOL in problems of static analysis of
    networks. We demonstrate the efficiency of automated theorem
    proving in FOOL with an extensive set of experiments. In these
    experiments we compare the performance of Vampire on a large
    collection of problems from various sources translated to FOOL and
    ordinary first-order logic. Finally, we fix the syntax for FOOL in
    TPTP, the standard language of first-order theorem provers.",
  paper = "Kote18.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@article{Kova04,
  author = "Kovacs, Laura and Jebelean, Tudor",
  title = {{Practical Aspects of Imperative Program Verification in
            Theorema}},
  journal = "Analele Universitatu din Timisoara",
  volume = "XXXXIX",
  number = "2",
  year = "2004",
  abstract =
    "Approaching the problem of imperative program verification from a
    practical point of view has certain implications concerning: the style
    of specifications, the programming language which is used, the help
    provided to the user for finding appropriate loop invariants, the
    theoretical frame used for formal verification, the language used
    for expressing generated verification theorems as well as the database
    of necessary mathematical knowledge, and finally the proving power,
    style and language. The Theorema system (www. theorema.org) has
```

```
    certain capabilities which make it appropriate for such a practical
    approach. Our approach to imperative program verification in Theorema
    is based on Hoare Logic and the Weakest Precondition Strategy. In
    this paper we present some practical aspect of our work, as well as a
    novel method for verification of pro- grams that contain loops, namely
    a method based on recurrence equation solvers for generating the
    necessary loop invariants and termination terms automatically.  We
    have tested our system with numerous examples, some of these example
    are presented at the end of the paper.",
  paper = "Kova04.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Kova04a,
  author = "Kovacs, Laura and Jebelean, Tudor",
  title = {{Automated Generation of Loop Invariants by Recurrence
            Solving in Theorema}},
  year = "2004",
  abstract =
    "Most of the properties established during program verification are
    either invariants or depend crucially on invariants. The effectiveness
    of automated verification of (imperative) programs is therefore
    sensitive to the ease with which invariants, even trivial ones, can be
    automatically deduced. We present a method for invariant generation
    that relies on combinatorial techniques, namely on recurrence solving
    and variable elimination. The effectiveness of the method is
    demonstrated on examples.",
  paper = "Kova04a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Kova07,
  author = "Kovacs, Laura",
  title = {{Automated Invariant Generation by Algebraic Techniques for
            Imperative Program Verification in Theorema}},
  school = "Johannes-Kepler University Linz",
  year = "2007",
  abstract =

    "This thesis presents algebraic and combinatorial approaches for
    reasoning about imperative loops with assignments, sequencing and
    conditionals.

    A certain family of loops, called P-solvable , is defined for which
```

```
    the value of each program variable can be expressed as a polynomial of
    the initial values of variables, the loop counter, and some new
    variables where there are algebraic dependencies among the new
    variables.  For such loops, a systematic method is developed for
    generating polynomial invariants.  Further, if the bodies of these
    loops consist only of assignments and conditional branches, and test
    conditions in the loop and conditionals are ignored, the method is
    shown to be complete for some special cases.  By completeness we mean
    that it generates a set of polynomials from which, under additional
    assumptions for loops with conditional branches, any polynomial
    invariant can be derived. Many non-trivial algorithms working on
    numbers can be naturally implemented using P-solvable loops.

    By combining advanced techniques from algorithmic combinatorics,
    symbolic summation, computer algebra and computational logic, a
    framework is developed for generating polynomial invariants for
    imperative programs operating on numbers.

    Exploiting the symbolic manipulation capabilities of the computer
    algebra system Mathematica , these techniques are implemented in a new
    software package called Aligator .  By using several combinatorial
    packages developed at RISC, Aligator includes algorithms for solving
    special classes of recurrence relations (those that are either
    Gosper-summable or C-finite) and generating polynomial dependencies
    among algebraic exponential sequences. Using Aligator , a complete set
    of polynomial invariants is successfully generated for numerous
    imperative programs working on numbers.

    The automatically obtained invariant assertions are subsequently used
    for proving the partial correctness of programs by generating
    appropriate verification conditions as first-order logical formulas.
    Based on Hoare logic and the weakest precondition strategy, this
    verification process is supported in an imperative verification
    environment implemented in the Theorema system.  Theorema is
    convenient for such an integration given that it is built on top of
    the computer algebra system Mathematica and includes automated methods
    for theorem proving in predicate logic, domain specific reasoning and
    proving by induction.",
  paper = "Kova07.pdf"
}
```

———-

— axiom.bib —

```
@inproceedings{Kova08,
  author = "Kovacs, Laura",
  title = {{Reasoning Algebraiclly About P-Solvable Loops}},
  booktitle = "Int. Conf. on Tools and Algorithms for the Construction
               and Analysis of Systems",
  pages = "249-264",
  year = "2008",
  abstract =
```

```
    "We present a method for generating polynomial invariants for a
    subfamily of imperative loops operating on numbers, called the
    P-solvable loops. The method uses algorithmic combinatorics and
    algebraic techniques. The approach is shown to be complete for some
    special cases. By completeness we mean that it generates a set of
    polynomial invariants from which, under additional assumptions, any
    polynomial invariant can be derived. These techniques are implemented
    in a new software package Aligator written in Mathematica and
    successfully tried on many programs implementing interesting
    algorithms working on numbers.",
  paper = "Kova08.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Kova13,
  author = "Kovacs, Laura and Voronkov, Andrei",
  title = {{First-Order Theorem Proving and Vampire}},
  year = "2013",
  link = "\url{http://www.cse.chalmers.se/~laurako/pub/CAV13_Kovacs.pdf}",
  abstract =
    "In this paper we give a short introduction in first-order theorem
    proving and the use of the theorem prover Vampire. We discuss the
    superposition calculus and explain the key concepts of saturation
    and redundancy elimination, present saturation algorithms and
    preprocessing, and demonstrate how these concepts are implemented
    in Vampire. Further, we also cover more recent topics and features
    of Vampire designed for advanced applications, including
    satisfiability checking, theory reasoning, interpolation,
    consequence elimination, and program analysis.",
  paper = "Kova13.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Kova16,
  author = "Kovacs, Laura",
  title = {{Symbolic Computation and Automated Reasoning for Program
            Analysis}},
  journal = "LNCS",
  volume = "9681",
  pages = "20-27",
  year = "2016",
  abstract =
    "This talk describes how a combination of symbolic computation
    techniques with first-order theorem proving can be used for solving
```

```
    some challenges of automating program analysis, in particular for
    generating and proving properties about the logically complex parts of
    software. The talk will first present how computer algebra methods,
    such as Grbner basis computation, quantifier elimination and
    algebraic recurrence solving, help us in inferring properties of
    program loops with non-trivial arithmetic. Typical properties inferred
    by our work are loop invariants and expressions bounding the number of
    loop iterations. The talk will then describe our work to generate
    first-order properties of programs with unbounded data structures,
    such as arrays. For doing so, we use saturation-based first-order
    theorem proving and extend first-order provers with support for
    program analysis. Since program analysis requires reasoning in the
    combination of first-order theories of data structures, the talk
    also discusses new features in first-order theorem proving, such as
    inductive reasoning and built-in boolean sort. These extensions allow
    us to express program properties directly in first-order logic and
    hence use further first-order theorem provers to reason about program
    properties.",
  paper = "Kova16.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Kowa63,
  author = "Kowalsky, Hans Joachim",
  title = {{Linear Algebra}},
  year = "1963",
  publisher = "Walter de Gruyter",
  comment = "(German)"
}
```

———-

— **axiom.bib** —

```
@article{Kowa84,
  author = "Kowalski, R.",
  title = {{The Relation Between Logic Programming and Logic
            Specification}},
  journal = "Phil. Trans. R. Soc. Lond.",
  volume = "312",
  pages = "345-351",
  year = "1984",
  link = "\url{www.doc.ic.ac.uk/~rak/papers/logic%20programming%20and%20specification.pdf}",
  abstract =
    "Formal logic is widely accepted as a program specification
    language in computing science. It is ideally suited to the
    representation of knowledge and the description of problems
    without regard to the choice of programming language. Its use as a
```

```
    specification language is compatible not only with conventional
    programming languages but also with programming languages based
    entirely on logic itself. In this paper I shall investigate the
    relation that holds when both programs and program specifications
    are expressed in formal logic.

    In many cases, when a specification {\sl completely} defines the
    relations to be computed, there is no syntactic distinction
    between specification and program. Moreover the same mechanism
    that is used to execute logic programs, namely automated
    deduction, can also be used to execute logic specifications. Thus
    all relations defined by complete specifications are
    executable. The only difference between a complete specification
    and a program is one of efficiency. A program is more efficient
    than a specification.",
  paper = "Kowa84.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Koze94,
  author = "Kozen, Dexter and Landau, Susan and Zippel, Richard",
  title = {{Decomposition of Algebraic Functions}},
  journal = "LNCS",
  volume = "877",
  pages = "80-92",
  year = "1994",
  abstract =
    "Functional decomposition -- whether a function $f(x)$ can be
    written as a composition of functions $g(h(x))$ in a nontrivial
    way -- is an important primitive in symbolic computation
    systems. The problem of univariate polynomial decomposition was
    shown to have an efficient solution by Kozen and Landau.
    Dickerson and von zur Gathen gave algorithms for certain
    multivariate cases. Zippel showed how to decompose rational
    functions. In this paper, we address the issue of decomposition of
    algebraic functions. We show that the problem is related to
    univariate resultants in algebraic function fields, and in fact
    can be reformulated as a problem of resultant decomposition. We
    characterize all decompositions of a given algebraic function up
    to isomorphism, and give an exponential time algorithm for finding
    a nontrivial one if it exists. The algorithm involves genus
    calculations and constructing transcendental generators of fields
    of genus zero.",
  paper = "Koze94.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Kran07,
  author = "Krantz, Steven G.",
  title = {{The Proof is in the Pudding}},
  year = "2007",
  link = "\url{https://www.math.wustl.edu/~sk/books/proof.pdf}",
  paper = "Kran07.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kreb11,
  author = "Krebbers, Robbert and Spitters, Bas",
  title = {{Computer Certified Efficient Exact Reals in Coq}},
  journal = "LNCS",
  volume = "6824",
  year = "2011",
  abstract =
    "Floating point operations are fast, but require continuous effort
    on the part of the user in order to ensure that the results are
    correct. This burden can be shifted away from the user by
    providing a library of exact analysis in which the computer
    handles the error estimates. We provide an implementation of the
    exact real numbers in the Coq proof assistant. This improves on
    the earlier Coq-implementation by O'Connor in two ways: we use
    dyadic rationals built from the machine integers and we optimize
    computation of power series by using approximate
    division. Moreover, we use type classes for clean mathematical
    interfaces. This appears to be the first time that type classes
    are used in heavy computation. We obtain over a 100 times speed up
    of the basic operations and indications for improving the Coq system.",
  paper = "Kreb11.pdf"
}
```

———-

— **axiom.bib** —

```
@inbook{Krei98,
  author = "Kreitz, Christoph",
  title = {{Program Synthesis}},
  booktitle = "Automated Deduction -- A Basis for Applications (Vol III)",
  publisher = "Springer",
  year = "1998",
  chapter = "5",
  pages = "105-134",
  isbn = "978-94-017-0437-3",
  comment = "Applied Logic Series, volume 10",
  paper = "Krei98.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Kreix11,
  author = "Kreitz, Christoph and Rahli, Vincent",
  title = {{Introduction to Classic ML}},
  type = "technical report",
  institution = "Cornell University",
  year = "2011",
  paper = "Krei11.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Krip80,
  author = "Kripke, Saul A.",
  title = {{Naming and Necessity}},
  publisher = "Harvard University Press",
  year = "1980",
  isbn = "0-674-59846-6",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Kubu18,
  author = "unknown",
  title = {{Algebras We Love}},
  year = "2018",
  link = "\url{https://kubuszok.com/2018/algebras-we-love}",
  keywords = "printed, DONE"
}
```

———-

1.2.12 L

— **axiom.bib** —

```
@book{Laka76,
  author = "Lakatos, Imre",
  title = {{Proofs and Refutations}},
  year = "1976",
```

```
  publisher = "Cambridge University Press",
  isbn = "978-1-107-53405-6",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Lama19,
  author = "Lamagna, Edmund A.",
  title = {{Computer Algebra: Concepts and Techniques}},
  publisher = "CRC Press",
  year = "2019",
  isbn = "978-1-138-09314-0",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Lamb16,
  author = "Lambe, Larry A.",
  title = {{An Algebraic Study of the Klein Bottle}},
  journal = "Homotopy and Related Structures",
  volume = "11",
  number = "4",
  pages = "885-891",
  year = "2016",
  abstract =
    "We use symbolic computation (SC) and homological perturbation
    (HPT) to compute a resolution of the integers $\mathbb{Z}$ over
    the integer group ring of $G$, the fundamental group of the Klein
    bottle. From this it is easy to read off the homology of the Klein
    bottle as well as other information.",
  paper = "Lamb16.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Lamp19,
  author = "Lample, Guillaume and Charton, Francois",
  title = {{Deep Learning for Symbolic Mathematics}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1912.01412.pdf}",
  link = "\url{https://www.youtube.com/watch?v=0_sHHG5_lr8}",
  abstract =
```

```
    "Neural networks have a reputation for being better at solving
    statistical or approximate problems than at performing
    calculations or working with symbolic data. In this paper, we show
    that they can be surprisingly good at more elaborated tasks in
    mathematics, such as symbolic integration and solving differential
    equations. We propose a syntax for representing mathematical
    problems, and methods for generating large datasets that can be
    used to train sequence-to-sequence models. We achieve results that
    outperform commercial Computer Algebra Systems such as Matlab or
    Mathematica.",
  paper = "Lamp19.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Lamp99,
  author = "Lamport, Leslie and Paulson, Lawrence C.",
  title = {{Should Your Specification Language Be Typed?}},
  journal = "ACM Trans. on Document Formatting",
  volume = "21",
  number = "3",
  year = "1999",
  pages = "502-526",
  abstract =
    "Most specification languages have a type system. Type systems are
    hard to get right, and getting them wrong can lead to
    inconsistencies. Set theory can serve as the basis for a
    specification language without types. This possibility, which has
    been widely overlooked, offers many advantages. Untyped set theory
    is simple and is more flexible than any simple typed
    formalism. Polymorphism, overloading, and subtyping can make a
    type system more powerful, but at the cost of increased
    complexity, and such refinements can never attain the flexibility
    of having no types at all. Typed formalisms have advantages too,
    stemming from the power of mechanical type checking. While types
    serve little purpose in hand proofs, they do help with mechanized
    proofs. In the absence of verification, type checking can catch
    errors in specifications. It may be possible to have the best of
    both worlds by adding typing annotations to an untyped
    specification language. We consider only specification languages,
    not programming languages",
  paper = "Lamp99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Lamp09,
  author = "Lamport, Leslie",
  title = {{The PlusCal Algorithm Language}},
  year = "2009",
  link = "\url{https://lamport.azurewebsites.net/pubs/pluscal.pdf}",
  abstract =
    "Algorithms are different from programs and should not be
    described with programming languages. The only simple alternative
    to programming languages has been pseudo-code. PlusCal is an
    algorithm language that an be used right now to replace
    pseudo-code, for both sequential and concurrent algorithms. It is
    based on the TLA+ specification language, and a PlusCal algorithm
    is automatically translated to a TLA+ specification that can be
    checked with the TLC model checker and reasoned about formally.",
  paper = "Lamp09.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Lamp88,
  author = "Lampson, B. and Burstall, R.",
  title = {{Pebble, a Kernel Language for Modules and Abstract Data Types}},
  journal = "Information and Computation",
  volume = "76",
  pages = "278-346",
  year = "1988",
  abstract =
    "A small set of constructs can simulate a wide variety of apparently
    distinct features in modern programming languages.  Using a kernel
    language called Pebble based on the typed lambda calculus with
    bindings, declarations, dependent types, and types as compile time
    values, we show how to build modules, interfaces and implementations,
    abstract data types, generic types, recursive types, and unions.
    Pebble has a concise operational semantics given by inference rules.",
  paper = "Lamp88.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Lams00,
  author = "Lamsweerde, Axel van",
  title = {{Formal Specification: A Roadmap}},
  booktitle = "The Future of Software Engineering",
  publisher = "ACM Press",
  year = "2000",
  abstract =
    "Formal specifications have been a focus of software engineering
```

```
    research for many years and have been applied in a wide variety of
    settings. Their industrial use is still limited but has been steadily
    growing. After recalling the essence, role, usage, and pitfalls of
    formal specification, the paper reviews the main specification
    paradigms to date and discuss their evaluation criteria. It then
    provides a brief assessment of the current strengths and weaknesses of
    today's formal specification technology. This provides a basis for
    formulating a number of requirements for formal specification to
    become a core software engineering activity in the future.",
  paper = "Lams00.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Lanc13,
  author = "Lanczos, Cornelius",
  title = {{Cornelius Lanczos (1893-1974) about Mathematics}},
  year = "2013",
  link = "\url{https://www.youtube.com/watch?v=avSHHi9QCjA}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Land66,
  author = "Landin, P.J.",
  title = {{The Next 700 Programming Languages}},
  journal = "Communications of the ACM",
  volume = "9",
  pages = "157-166",
  year = "1966",
  abstract =
    "A family of unimplemented computing languages is described that
    is intended to span differences of application areas by a unified
    framework. This framework dictates the rules about the uses of
    user-coined names, and the conventions about characterizing
    functional relationships. Within this framework the design of a
    specific language splits into two independent parts. One is the
    choice of written appearances of programs (or more generally,
    their physical representation). The other is the choice of the
    abstract entities (such as numbers, character-strings, lists of
    them, functional relations among them) that can be referred to in
    the language.

    The system is biased towards ``expressions'' rather than
    ``statements''. It includes a nonprocedural (purely functional)
    subsystem that aims to expand the class of users' needs that can
```

```
    be met by a single print-instruction, without sacrificing the
    important properties that make conventional right-hand-side
    expressions easy to construct and understand.",
  paper = "Land66.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@book{Lang05,
  author = "Lang, Serge",
  title = {{Algebra}},
  publisher = "Springer",
  year = "2005",
  series = "Graduate Texts in Mathematics",
  isbn = "978-0387953854"
}
```

———-

— **axiom.bib** —

```
@InCollection{Laue82,
  author = "Lauer, M.",
  title = {{Computing by Homomorphic Images}},
  booktitle = "Computer Algebra: Symbolic and Algebraic Computation",
  pages = "139-168",
  year = "1982",
  publisher = "Springer",
  isbn = "978-3-211-81684-4",
  abstract =
    "After explaining the general technique of Computing by homomorphic
    images, the Chinese remainder algorithm and the Hensel lifting
    construction are treated extensively. Chinese remaindering is first
    presented in an abstract setting. Then the specialization to Euclidean
    domains, in particular $\mathbb{Z}$, $\mathbb{K}[y]$, and
    $\mathbb{Z}[y_1,\ldots,y_n]$ is treated. The lifting construction
    is first also presented in an abstract form from which Hensel's
    Lemma derives by specialization. After introducing Zassenhaus'
    quadratic lifting construction, again, the case of $\mathbb{Z}$
    and $\mathbb{X}[y_1,\ldots,y_r]$ is considered. For both techniques,
    Chinese remaindering as well as the lifting algorithms, a complete
    computational example is presented and the most frequent application
    is discussed.",
}
```

———-

— **axiom.bib** —

```
@article{Lawv69,
  author = "Lawvere, F. William",
  title = {{Adjointness in Foundations}},
  journal = "Dialectica",
  volume = "23",
  number = "3/4",
  year = "1969",
  pages = "281-296",
  paper = "Lawv69.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Lawv06,
  author = "Lawvere, F. William",
  title = {{Adjointness in Foundations: Author's Commentary}},
  journal = "Theory and Applications of Categories",
  volume = "16",
  pages = "1-16",
  year = "2006",
  paper = "Lawv06.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Laza79,
  author = "Lazard, Daniel",
  title = {{Systems of Algebraic Equations}},
  booktitle = "International Symbposium on Symbolic and Algebraic
               Manipulation",
  pages = "88-94",
  publisher = "Springer",
  year = "1979",
  comment = "LNCS 72",
  paper = "Laza79.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Lean19,
  author = "Lean",
  title = {{Lean Chat Room}},
  link = "\url{https://leanprover.zulipchat.com}",
  year = "2019"
```

```
}
```

———-

— **axiom.bib** —

```
@book{Lear15,
  author = "Leary, Christopher C. and Kristiansen, Lars",
  title = {{A Friendly Introduction to Mathematical Logic}},
  publisher = "Milne Library, Geneseo",
  year = "2015",
  isbn = "978-1-942341-32-1",
  paper = "Lear15.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Lebo98,
  author = "Coen, Ethan and Coen, Joel",
  title = {{The Big Lebowski}},
  year = "1998",
  link = "\url{https://www.imdb.com/title/tt0118715}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Leex07,
  author = "Lee, Daniel K. and Crary, Karl and Harper, Robert",
  title = {{Towards a Mechanized Metatheory of Standard ML}},
  booktitle = "POPL'07",
  publisher = "ACM",
  year = "2007",
  abstract =
    "We present an internal language with equivalent expressive power to
    Standard ML, and discuss its formalization in LF and the
    machine-checked verification of its type safety in Twelf. The
    internal language is intended to serve as the target of elaboration in
    an elaborative semantics for Standard ML in the style of Harper and
    Stone. Therefore, it includes all the programming mechanisms
    necessary to implement Standard ML, including translucent modules,
    abstraction, polymorphism, higher kinds, references, exceptions,
    recursive types, and recursive functions. Our successful formalization
    of the proof involved a careful interplay between the precise
    formulations of the various mechanisms, and required the invention of
    new representation and proof techniques of general interest.",
  paper = "Leex07.pdf",
  keywords = "printed"
```

```
}
```

———-

— **axiom.bib** —

```
@inproceedings{Leis87,
  author = "Leiss, Hans",
  title = {{On Type Inference for Object-Oriented Programming Languages}},
  booktitle = "Int. Workshop on Computer Science Logic",
  year = "1987",
  pages = "151-172",
  abstract =
    "We present a type inference calculus for object-oriented programming
    languages. Explicit polymorphic types, subtypes and multiple
    inheritance are allowed. Class types are obtained by selection from
    record types, but not considered subtypes of record types. The subtype
    relation for class types reflects the (mathematically clean)
    properties of subclass relations in object-oriented programming to a
    better extend than previous systems did.

    Based on Mitchells models for type inference, a semantics for types is
    given where types are sets of values in a model of type-free lambda
    calculus. For the sublanguage without type quantifiers and subtype
    relation, automatic type inference is possible by extending Milners
    algorithm W to deal with a polymorphic fixed-point rule."
}
```

———-

— **axiom.bib** —

```
@book{Lemx68,
  author = "Lem, Stanislaw",
  title = {{His Master's Voice}},
  publisher = "MIT Press",
  year = "1968"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Lens84,
  author = "Lenstra, Arjen K.",
  title = {{Polynomial Factorizaion by Root Approximation}},
  booktitle = "International Symbposium on Symbolic and Algebraic
               Manipulation",
  pages = "272-276",
  publisher = "Springer",
  year = "1984",
```

```
  comment = "LNCS 174",
  abstract =
    "We show that a constructive version of the fundamental theorem of
    algebra [3], combined with the basis reduction algorithm from [1],
    yields a polynomial-time algorithm for factoring polynomials in
    one variable with rational coefficients.",
  paper = "Lens84.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Lepi16,
  author = "Lepigre, Rodolphe",
  title = {{Semantics and Implementation of an Extension of ML for
            Proving Programs}},
  year = "2016",
  school = "Universite Grenoble Alpes",
  link = "\url{http://lepigre.fr/these/manuscript_lepigre.pdf}",
  abstract =
    "In recent years, proof assistants have reached an impressive level
    of maturity. They have led to the certification of complex
    programs such as compilers and operating systems. Yet, using a
    proof assistant requires highly specialised skills and it remains
    very different from standard programming. To bridge this gap, we
    aim at designing an ML-style programming language with support for
    proofs of programs, combining in a single tool the flexibility of
    ML and the fine specification features of a proof assistant. In
    other words, the system should be suitable both for programming
    (in the strongly-typed, functional sense) and for gradually
    increasing the level of guarantees met by programs, on a by-need
    basis.

    We thus define and study a call-by-value language whose type
    system extends higher-order logic with an equality type over
    untyped programs, a dependent function type, classical logic and
    subtyping. The combination of call-by-value evaluation, dependent
    functions and classical logic is known to raise consistency
    issues. To ensure the correctness of the system (logical
    consistency and runtime safety), we design a theoretical framework
    based on Krivine's classical realisability. The construction of
    the model relies on an essential property linking the different
    levels of interpretation of types in a novel way.

    We finally demonstrate the expressive power of our system using
    our prototype implementation, by proving properties of standard
    programs like the map function on lists or the insertion sort.",
  paper = "Lepi16.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Lero92,
  author = "Leroy, Xavier",
  title = {{Polymorphic Typing of an Algorithmic Language}},
  type = "research report",
  institution = "INRIA",
  number = "N1778",
  year = "1992",
  abstract =
    "The polymorphic type discipline, as in the ML language, fits well
    within purely applicative languages, but does not extend naturally
    to the main feature of algorithmic languages: in-place update of
    data structures. Similar typing difficulties arise with other
    extensions of applicative languages: logical variables,
    communication channels, continuation handling. This work studies
    (in the setting of relational semantics) two new approaches to the
    polymorphic typing of these non-applicative features. The first
    one relies on a restriction of generalization over types (the
    notion of dangerous variables), and on a refined typing of
    functional values (closure typing). The resulting type system is
    compatible with the ML core language, and is the most expressive
    type systems for ML with imperative features so far. The second
    approach relies on switching to ``by-name'' sematics for the
    constructs of polymorphism, instead of the usual ``by-value''
    semanticcs. The resulting language differs from ML, but lends
    itself easily to polymorphic typing. Both approaches smoothly
    integrate non-applicative features and polymorphic typing.",
  paper = "Lero92.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Lero94,
  author = "Leroy, Xavier",
  title = {{Manifest Types, Modules, and Separate Compilation}},
  booktitle = "Principles of Programming Languages POPL'94",
  year = "1994",
  publisher = "ACM Press",
  pages = "109-122",
  abstract =
    "This paper presents a variant of the SML module system
    that introduces a strict distinction between abstract types
    and manifest types (types whose definitions are part of the
    module specification), while retaining most of the expressive
    power of the SML module system.  The resulting module
    system provides much better support for separate compilation.",
  paper = "Lero94.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Lero17,
  author = "Leroy, Xavier and Doligez, Damien and Frish, Alain and
            Garrigue, Jacques and Remy, Didier and Vouillon, Jerome",
  title = {{The OCaml System (release 4.06)}},
  publisher = "INRIA",
  year = "2017",
  link =
   "\url{https://caml.inria.fr/pub/distrib/ocaml-4.06/ocaml-4.06-refman.pdf}"
}
```

———-

— **axiom.bib** —

```
@book{Leve11,
  author = "Leveson, Nancy",
  title = {{Engineering a Safer World}},
  year = "2011",
  isbn = "978-0-262-01662-9",
  publisher = "MIT Press"
}
```

———-

— **axiom.bib** —

```
@misc{Lewi17a,
  author = "Lewis, Robert Y. and Wu, Minchao",
  title = {{A Bi-directional Extensible Ad Hoc Interface between Lean
            and Mathematica}},
  year = "2017",
  link = "\url{https://robertylewis.com/leanmm/lean_mm.pdf}",
  abstract =
    "We implement a user-extensible ad hoc connection between the Lean
    proof assistant and the computer algebra system Mathematica. By
    reflecting the syntax of each system in the other and providing a
    flexible interface for extending translation, our connection
    allows for the exchange of arbitrary information between the two
    systemss. We show how to make use of the Lean metaprogramming
    framework to verif certain Mathematica computations, so that the
    rigor of the proof assistant is not compromised. We also establish
    a connection in the opposite direction, using Mathematica to
    import and process proof terms from Lean.",
  paper = "Lewi17a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Lewi20,
  author = "Lewis, Robert Y.",
  title = {{The Lean Mathematical Library}},
  year = "2020",
  link = "\url{https://www.youtube.com/watch?v=fnRrwsaFUM8}"
}
```

———-

— **axiom.bib** —

```
@misc{Lica16,
  author = "Licata, Dan",
  title = {{A Functional Programmer's Guide to Homotopy Type Theory}},
  year = "2016",
  comment = "ICFP conference",
  link = "\url{https://www.youtube.com/watch?v=caSOTjr1z18}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Lica16a,
  author = "Licata, Dan",
  title = {{Computing with Univalence}},
  year = "2016",
  comment = "Institute for Advanced Study",
  link = "\url{https://www.youtube.com/watch?v=YDdDuq2AGw4}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Limo92,
  author = "Limongelli, C. and Temperini, M.",
  title = {{Abstract Specification of Structures and Methods in Symbolic
           Mathematical Computation}},
  journal = "Theoretical Computer Science",
  volume = "104",
  year = "1992",
  pages = "89-107",
  abstract =
    "This paper describes a methodology based on the object-oriented
```

```
    programming paradigm, to support the design and implementation of a
    symbolic computation system. The requirements of the system are
    related to the specification and treatment of mathematical
    structures. This treatment is considered from both the numerical and
    the symbolic points of view. The resulting programming system should
    be able to support the formal definition of mathematical data
    structures and methods at their highest level of abstraction, to
    perform computations on instances created from such definitions, and
    to handle abstract data structures through the manipulation of their
    logical properties. Particular consideration is given to the
    correctness aspects. Some examples of convenient application of the
    proposed design methodology are presented.",
  paper = "Limo92.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Limo93,
  author = "Limongelli, C. and Temperini, M.",
  title = {{On the Uniform Representation of Mathematical Data
            Structures}},
  booktitle = "DISCO 1993",
  year = "1993",
  pages = "319-330",
  publisher = "Springer",
  abstract =
    "Topics about the integration of the numeric and symbolic
    computation paradigms are discussed. Mainly an approach through a
    uniform representation of numbers and symbols is presented, that
    allows for the application of algebraic algorithms to numeric
    problems. The $p$-adic constructions is the basis of the unifying
    representation environment. An integrated version of the Hensel
    algorithm is presented, which is able to perform symbolic and
    numeric computations over instances of ground (concrete) and
    parametric structures, and symbolic computations over instances of
    abstract structures. Examples are provided to show how the
    approach outlined and the proposed implementation can treat both
    cases of symbolic and numeric computations. In the numeric case it
    is shown that the proposed extension of the Hensel Algorithm can
    allow for the exact manipulation of numbers. Moreover, such an
    extension avoids the use of simplification algorithms, since the
    computed results are already in simplified form.",
  paper = "Limo93.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Limx20,
  author = "Lim, Jay P. and Aanjaneya, Mridul and Gustafson, John
            and Nagarakatte, Santosh",
  title = {{A Novel Approach to Generate Correctly Rounded Math
            Libraries for New Floating Point Representations}},
  link = "\url{https://arxiv.org/pdf/2007.05344.pdf}",
  year = "2020",
  abstract =
    "Given the importance of floating-point (FP) performance in
    numerous domains, several new variants of FP and its alternatives
    have been proposed (e.g. Bfloat16, TensorFloat32, and
    Posits). These representations do not have correctly rounded math
    libraries. Further, the use of existing FP libraries for these new
    representations can produce incorrect results. This paper proposes
    a novel methodology for generating polynomial approximations that
    can be used to implement correctly rounded math libraries. Existing
    methods produce polynomials that approximate the real value of an
    elementary function $f(x)$ and experience wrong results due to
    errors in the approximation and due to rounding errors in the
    implementation. In contrast, our approach generates polynomials
    that approximate the correctly rounded value of $f(x)$ (i.e. the
    value of $f(x)$ rounded to the target representation). This
    methodology provides more margin to identify efficient polynomials
    that produce correctly rounded results for all inputs. We frame
    the problem of generating efficient polynomials that produce
    correctly rounded results as a linear programming problem. Our
    approach guarantees that we produce the correct result even with
    range reduction techniques. Using our approach, we have developed
    correctly rounded, yet faster, implementations of elementary
    functions for multiple target representations. Our Bfloat16
    library is 2.3x faster than the corresponding state-of-the-art
    while producing correct results for all inputs.",
  paper = "Limx20.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Linc92,
  author = "Lincoln, Patrick and Mitchell, John C.",
  title = {{Algorithmic Aspects of Type Inference with Subtypes}},
  booktitle = "POPL 92",
  pages = "293-304",
  year = "1992",
  abstract =
    "We study the complexity of type inference for programming languages
    with subtypes. There are three language variations that effect the
    problem: (i) basic functions may have polymorphic or more limited
    types, (ii) the subtype hierarchy may be fixed or vary as a result of
    subtype declarations within a program, and (iii) the subtype hierarchy
    may be an arbitrary partial order or may have a more restricted form,
    such as a tree or lattice. The naive algorithm for infering a most
```

```
    general polymorphic type, undervariable subtype hypotheses, requires
    deterministic exponential time. If we fix the subtype ordering, this
    upper bound grows to nondeterministic exponential time. We show that
    it is NP-hard to decide whether a lambda term has a type with respect
    to a fixed subtype hierarchy (involving only atomic type names). This
    lower bound applies to monomorphic or polymorphic languages. We give
    PSPACE upper bounds for deciding polymorphic typability if the subtype
    hierarchy has a lattice structure or the subtype hierarchy varies
    arbitrarily. We also give a polynomial time algorithm for the limited
    case where there are of no function constants and the type hierarchy
    is either variable or any fixed lattice.",
  paper = "Linc92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Lind88,
  author = "Lindsay, Peter A.",
  title = {{A Survery of Mechanical Support for Formal Reasoning}},
  journal = "Software Engineering Journal",
  volume = "3",
  number = "1",
  year = "1988",
  pages = "3-27",
  paper = "Lind88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Lint10,
  author = "Linton, S. and Hammond, K. and Konovalov, A. and Al Zain, A.D.
            and Trinder, P. and Horn, P.",
  title = {{Easy Compostion of Symbolic Computation Software: A New Lingua
           Franca for Symbolic Computation}},
  booktitle = "Proc. ISSAC 2010",
  publisher = "ACM",
  year = "2010",
  pages = "339-346",
  abstract =
    "We present the results of the first four years of the European
    research project SCIEnce (www.symbolic-computation.org),
    which aims to provide key infrastructure for symbolic computation
    research.  A primary outcome of the project is that we have developed
    a new way of combining computer algebra systems using the Symbolic
    Computation Software Composability Protocol (SCSCP), in which both
    protocol messages and data are encoded in the OpenMath format.  We
    describe SCSCP middleware and APIs, outline some implementations for
```

```
    various Computer Algebra Systems (CAS), and show how SCSCP-compliant
    components may be combined to solve scientific problems that can not
    be solved within a single CAS, or may be organised into a system for
    distributed parallel computations.",
  paper = "Lint10.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@misc{lion137,
  author = "Unknown",
  title = {{Thoughts in Free Time}},
  link = "\url{https://lion137.blogspot.com/2019/02/fundamental-algorithms-polynomial-gcd.html}",
  comment = "\url{https://github.com/lion137/Fundamental_Algorithsms}",
  year = "2019"
}
```

———-

— **axiom.bib** —

```
@article{Lipt94,
  author = "Lipton, Richard J.",
  title = {{Straight-Line Complexity and Integer Factorization}},
  journal = "LNCS",
  volume = "877",
  pages = "71-79",
  year = "1994",
  abstract =
    "We show that if polynomials with many rational roots have
    polynomial length straight-line complexity, then integer
    factorization is 'easy'",
  paper = "Lipt94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Lisk77a,
  author = "Liskov, Barbara and Zilles, Stephen",
  title = {{Programming with Abstract Data Types}},
  journal = "SIGPLAN Notices",
  volume = "9",
  number = "4",
  pages = "50-59",
  year = "1977",
  abstract =
```

```
    "The motivation behind the work in very-high-level languages is to
    ease the programming task by providing the programmer with a
    language containing primitives or abstractions suitable to his
    problem area. The programmer is then able to spend his effort in
    the right place; he concentrates on solving his problem, and the
    resulting program will be more reliable as a result. Clearly, this
    is a worthwhile goal.

    Unfortunately, it is very difficult for a designer to select in
    advance all the abstractions which the users of his language might
    need. If a language is to be used at all, it is likely to be used
    to solve problems which its designer did not envision, and for
    which the abstractions embedded in the language are not sufficient.

    This paper presents an approach which allows the set of built-in
    abstractions to be augmented when the need for a new data
    abstraction is discovered. This approach to the handling of
    abstraction is an outgrowth of work on designing a language for
    structured programming. Relevant aspects of this language are
    described, and examples of the use and definitions of abstraction
    are given.",
  paper = "Lisk77a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Lisk12,
  author = "Liskov, Barbara",
  title = {{Programming the Turing Machine}},
  year = "2012",
  link = "\url{https://www.youtube.com/watch?v=ibRar7sWulM}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Lohx01,
  author = "Loh, Andres and McBride, Conor and Swierstra, Wouter",
  title = {{A Tutorial Implementation of a Dependently Typed Lambda
            Calculus}},
  journal = "Foundations Informaticae",
  volume = "XXI",
  pages = "1001-1031",
  year = "2001",
  abstract =
    "We present the type rules for a dependently typed core calculus
    together with a straight-forward implementation in Haskell. We
    explicitly highlight the changes necessary to shift from a
```

```
    simply-typed lambda calculus to a dependently typed lambda
    calculus. We also describe how to extend our core language with
    data types and write several small example programs. The article
    is accompanied by an executable interpreter and example code that
    allows immediate experimentation with the system we describe.",
  paper = "Lohx01.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Lohx19,
  author = "Loh, Po-Shen",
  title = {{A Simple Proof of the Quadratic Formula}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1910.06709.pdf}",
  abstract =
    "This aarticle provides a simple proof of the quadratic formula,
    which also produces an efficient and natural method for solving
    general quadratic equations. The derivation is computationally
    light and conceptually natural, and has the potential to demystify
    quadratic equations for students worldwide.",
  paper = "Lohx19.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Long99,
  author = "Longley, John",
  title = {{Matching Typed and Untyped Realizability}},
  journal = "Theoretical Computer Science",
  volume = "23",
  number = "1",
  year = "1999",
  abstract =
    "Realizability interpretations of logics are given by saying what
    it means for computational objects of some kind to {\sl realize}
    logical formulae. The computational object in question might be
    drawn from an untyped universe of computation, such as a partial
    combinatory algebra, or they might be typed objects such as terms
    of a PCF-style programming language. In some instances, one can
    show that a particular untyped realizability interpretation
    matches a particular typed one, in the sense that they give the
    same set of realizable formulae. In this case, we have a very good
    fit indeed between the typed language and the untyped
    realizability model -- we refer to this condition as
    {\sl (constructive) logical full abstraction}.
```

```
    We give some examples of this situation for a variety of
    extensions of PCF. Of particular interest are some models that are
    logically fully abstract for typed languages including
    {\sl non-functional} features. Our results establish connections
    between what is computable in various programming languages and
    what is true inside various realizable toposes. We consider some
    examples of logical formulae to illustrate these ideas, in
    particular their application to exact real-number compatibility.",
  paper = "Long99.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Loos72,
  author = "Loos, Rudiger",
  title = {{Algebraic Algorithm Descriptions as Programs}},
  journal = "ACM SIGSAM Bulletin",
  volume = "23",
  year = "1972",
  pages = "16-24",
  abstract =
    "We propose methods for writing algebraic programs in an algebraic
    notation. We discuss the advantages of this approach and a specific
    example",
  paper = "Loos72.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Loos74,
  author = "Loos, Ruediger G. K.",
  title = {{Toward a Formal Implementation of Computer Algebra}},
  journal = "SIGSAM",
  volume = "8",
  number = "3",
  pages = "9-16",
  year = "1974",
  abstract =
    "We consider in this paper the task of synthesizing an algebraic
    system. Today the task is significantly simpler than in the pioneer
    days of symbol manipulation, mainly because of the work done by the
    pioneers in our area, but also because of the progress in other areas
    of Computer Science. There is now a considerable collection of
    algebraic algorithms at hand and a much better understanding of data
    structures and programming constructs than only a few years ago.",
  paper = "Loos74.pdf",
```

```
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@article{Loos76,
  author = "Loos, Rudiger",
  title = {{The Algorithm Description Language (ALDES) (report)}},
  journal = "ACM SIGSAM Bulletin",
  volume = "10",
  number = "1",
  year = "1976",
  pages = "14-38",
  abstract =
    "ALDES is a formalization of the method to describe algorithms used in
    Knuth's books. The largest documentation of algebraic algorithms,
    Collins' SAC system for Computer Algebra, is written in this
    language. In contrast to PASCAL it provides automatic storage
    deallocation. Compared to LISP equal emphasis was placed on efficiency
    of arithmetic, list processing, and array handling. To allow the
    programmer full control of efficiency all mechanisms of the system are
    accessible to him. Currently ALDES is available as a preprocessor to
    ANSI Fortran, using no additional primitives.",
  paper = "Loos76.pdf"
}
```

———-

— **axiom.bib** —

```
@inbook{Loos82,
  author = "Loos, R",
  title = {{Generalized Polynomial Remainder Sequences}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "115-138",
  abstract =
    "Given two polynomials over an integral domain, the problem is to
    compute their polynomial remainder sequence (p.r.s) over the same
    domain. Following Habischt, we show how certain powers of leading
    coefficients enter systematically all following remainders. The
    key tool is the subresultant chain of two polynomials. We study
    the primitive, the reduced and the improved subresultant p.r.s
    algorithm of Brown and Collins as basis for computing polynomial
    greatest common divisors, result and or Sturm sequences. Habicht's
    subresultant theorem allows new and simple proofs of many results
    and algorithms found in different ways in computer algebra.",
  paper = "Buch82.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@inbook{Loos82a,
  author = "Loos, R",
  title = {{Computing in Algebraic Extensions}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "173-188",
  abstract =
    "The aim of this chapter is an introduction to elementary
    algorithms in algebraic extensions, mainly over $\mathbb{Q}$ and,
    to some extent, over $GF(p)$. We will talk about arithmetic in
    $\mathbb{Q}(\alpha)$ and $GF(p^n)$ in Section 1 and some
    polynomial algorithms with coefficients from these domains in
    Section 2. Then, we will consider the field $K$ of all algebraic
    numbers over $\mathbb{Q}$ and show constructively that $K$ indeed
    is a field that multiple extensions can be replaced by single ones
    and the $K$ is algebraically closed, i.e. that zeros of algebraic
    number polynomials will be elements of $K$ (Section 4-6). For this
    purpose we develop a simple resultant calculus which reduces all
    operations on algebraic numbers to polynomial arithmetic on long
    integers together with some auxiliary arithmetic on rational
    intervals (Secion 3). Finally, we present some auxiliary algebraic
    number algorithms used in other chapters of this volume (Section
    7). This chapter does not include any special algorithms of
    algebraic number theory. For an introduction and survey with an
    extensive bibliography the reader is referred to Zimmer.",
  paper = "Buch82.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Loos92,
  author = "Loos, Rudiger and Collins, George E.",
  title = {{Revised Report on the ALgorithm Language ALDES}},
  publisher = "Institut fur Informatik",
  year = "1992"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Loux90,
  author = "Lou, Zhaohui",
  title = {{An Extended Calculus of Constructions}},
  school = "University of Edinburgh",
  year = "1990",
  abstract =
    "This thesis presents and studies a unifying theory of dependent
    types ECC Extended Calculus of Constructions. ECC integrates
    Coquand-Huet's (impredicative) calculus of constructions and
    Martin-Lof's (predicative) type theory with universes, and turns
    out to be a strong and expressive calculus for formalization of
    mathematics, structured proof development and program specification.

    The meta theory of ECC is studied and we show that the calculus
    has good meta-theoretic properties. The main proof theoretic
    result is the {\sl strong normalization theorem} which makes
    explicit the predicativity of the predicative universes. The
    strong normalization result shows the proof theoretic consistency
    of the calculus; in particular, it implies the consistency of the
    embedded intuitionistic higher-order logic and the decidability of
    the theory. The meta-theoretic results establish the theoretical
    foundations both for pragmatic applications in theorem proving and
    program specification and for computer implementations of the
    theory. ECC has been implemente in the proof development system
    LEGO developed by Pollack.

    In ECC, dependent $\Sigma$ types are non-propositional types
    residing in the predicative universes and propositions are lifted
    as higher-level types as well. This solves the known difficulty
    that adding strong $\Sigma$ types to an impredicative system
    results in logical paradox and enables $\Sigma$ types to be used
    to express the intuitionistic notion of subsets. $\Sigma$ types
    together with type universes hence provide useful abstraction and
    module mechanisms for abstract description of mathematical
    theories and basic mechanisms for program specification and
    adequate formalization of abstract mathematics (e.g. abstract
    algebras and notions in category theory). A notion of (abstract)
    mathematical theory can be described and leads to a promising
    approach to {\sl abstract reasoning} and {\sl structured
    reasoning}. Program specifications can be expressed by $\Sigma$
    types, using propositions in the embedded logic to describe
    program properties (for example, by an equality reflection result,
    computational equality can be modeled by the propositional
    Leibniz's equality definable in the theory). These developments
    allow comprehensive structuring of formal or rigorous development
    of proofs and programs.

    Also discussed is how the calculus can be understood
    set-theoretically. We explain an $\omega-Set$ (realizability)
    model of the theory. In particular, propositions can be
    interpreted as partial equivalence relations and the predicative
    type universes as corresponding to large set universes.",
  paper = "Loux90.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Loux08,
  author = "Lou, Zhaohui",
  title = {{Coercions in a Polymorphic Type System}},
  journal = "Math. Struct. in Comp. Science",
  volume = "18",
  pages = "729-751",
  year = "2008",
  abstract =
    "We incorporate the idea of coercive subtyping, a theory of
    abbreviation for dependent type theories, into the polymorphic
    type system in functional programming languages. The traditional
    type system with let-polymorphism is extended with argument
    coercions and function coercions, and a corresponding type
    inference algorithm is presented and proved to be sound and complete.",
  paper = "Loux08.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Loux12,
  author = "Lou, Zhaohui",
  title = {{Formal Semantics in Modern Type Theories with Coercive Semantics}},
  journal = "Linguistics and Philosophy",
  volume = "35",
  pages = "491-513",
  year = "2012",
  abstract =
    "In the formal semantics based on modern type theories, common
    nouns are interpreted as types, rather than as predicates on
    entities as in Montague's semantics. This brings about important
    advantages in linguistic interpretations but also leads to a
    limitation of expressive power because there are fewer operations
    on types as compared with those on predicates. The theory of
    coercive subtyping adequately extends the modern type theories
    and, as shown in this paper, plays a very useful role in making
    type theories more expressive for formal semantics. It not only
    gives a satisfactory solution to the basic problem of 'multiple
    categorisations' caused by interpreting common nouns as types, but
    provides a powerful formal framework to model interesting
    linguistic phenomena such as copredication, whose formal treatment
    has been found difficult in a Montagovian setting. In particular,
    we show how to formally introduce dot-types in a type theory with
    coercive subtyping and study some type-theoretic constructs that
    provide useful representational tools for reference transfers and
```

```
    multiple word meanings in formal lexical semantics.",
  paper = "Loux12.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Luca03,
  author = "Lucas, J.R.",
  title = {{Minds, Machines and Godel}},
  year = "2003",
  link =
  "\url{https://pdfs.semanticscholar.org/bde3/b731bf73ef6052e34c4465e57718c03b13f8.pdf}",
  paper = "Luca03.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Luck70,
  author = "Luckham, D.C. and Park, D.M.R and Paterson, M.S.",
  title = {{On Formalised Computer Programs}},
  journal = "J. of Computer and System Sciences",
  volume = "4",
  pages = "220-249",
  year = "1970",
  paper = "Luck70.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Luox96,
  author = "Luo, Zhaohui",
  title = {{Coercive Subtyping in Type Theory}},
  booktitle = "CSL'96 Euro. Ass. for Comput. Sci. Logic",
  year = "1996",
  abstract =
    "We propose and study coercive subtyping, a formal extension with
    subtyping of dependent type theories such as Martin-Lof's type theory
    [NPS90] and the type theory UTT [Luo94]. In this approach, subtyping
    with specied implicit coercions is treated as a feature at the level
    of the logical framework; in particular, subsumption and coercion are
    combined in such a way that the meaning of an object being in a
    supertype is given by coercive definition rules for the definitional
    equality. It is shown that this provides a conceptually simple and
    uniform framework to understand subtyping and coercion relations in
```

```
    type theories with sophisticated type structures such as inductive
    types and universes. The use of coercive subtyping in formal
    development and in reasoning about subsets of objects is discussed in
    the context of computer-assisted formal reasoning.",
  paper = "Luox96.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Luox11,
  author = "Luo, Zhaohui",
  title = {{Type-Theoretical Semantics with Coercive Subtyping}},
  year = "2011",
  comment = "Lecture Notes 2011 ESSLLI Summer School, Ljubljana,
             Slovenia",
  paper = "Luox11.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Luox13,
  author = "Luo, Zhaohui and Soloviev, Sergei and Xue, Tao",
  title = {{Coercive Subtyping: Theory and Implementation}},
  journal = "Information and Computation",
  year = "2013",
  volume = "223",
  pages = "18-42",
  link =
    "\url{https://hal-univ-tlse3.archives-ouvertes.fr/hal-01130574/document}",
  abstract =
    "Coercive subtyping is a useful and powerful framework of subtyping
    for type theories.  The key idea of coercive subtyping is subtyping
    as abbreviation. In this paper, we give a new and adequate formulation
    of T[C] , the system that extends a type theory T with coercive
    subtyping based on a set C of basic subtyping judgements, and show
    that coercive subtyping is a conservative extension and, in a more
    general sense, a definitional extension.  We introduce an intermediate
    system, the star-calculus T[C] , in which the positions that
    require coercion insertions are marked, and show that T[C] is a
    conservative extension of T and that T[C] is equivalent to T[C].
    This makes clear what we mean by coercive subtyping being a
    conservative extension, on the one hand, and amends a technical
    problem that has led to a gap in the earlier conservativity proof, on
    the other. We also compare coercive subtyping with the ordinary
    notion of subtyping  subsumptive subtyping, and show that the former
    is adequate for type theories with canonical objects while the latter
    is not. An improved implementation of coercive subtyping is done in
    the proof assistant Plastic.",
```

```
  paper = "Luox13.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Luth98,
  author = "Luther, Marko and Strecker, Martin",
  title = {{A Guided Tour through TYPELAB}},
  year = "1998",
  abstract =
    "This report gives a survey of TYPELAB, a specification and
    verification environment that integrates interactive proof
    development and automated proof search. TYPELAB is based on a
    constructive type theory, the Calculus of COnstructions, which can
    be understood as a combinations of a typed $\lambda$-calculus and
    an expressive higher-order logic. Distinctive features of the type
    system are dependent function types for modeling polymorphism and
    dependent record types for encoding specifications and
    mathematical theories. After presenting an extended example which
    demonstrates how program development by stepwise refinement of
    specifications can be carried out, the theory underlying the
    prover component of TYPELAB is described in detail. A calculus
    with metavariables and explicit substitutions is introduced, and
    the meta-theoretic properties of this calculus are
    analyzed. Furthermore, it is shown that this calculus provides an
    adequate foundation for automated proof search in fragments of the
    logic.",
  paper = "Luth98.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Lyxx15,
  author = "Ly, Kim Quyen",
  title = {{Formalization in Coq of Polynomial Interpretations on
            Rationals}},
  year = "2015",
  link = "\url{https://www.di.ens.fr/~quyen/publication/ly10.pdf}",
  paper = "Lyxx15.pdf",
  keywords = "printed"
}
```

———-

1.2.13 M

— **axiom.bib** —

```
@article{Ma90,
  author = "Ma, Keju and {von zur Gathen}, Joachim",
  title =
    {{Analysis of Euclidean Algorithms for Polynomials over Finite Fields}},
  journal = "J. Symbolic Computation",
  year = "1990",
  volume = "9",
  pages = "429-455",
  link = "\url{http://www.researchgate.net/publication/220161718_Analysis_of_Euclidean_Algorithms_for_Polynomial
  abstract = "
    This paper analyzes the Euclidean algorithm and some variants of it
    for computing the greatest common divisor of two univariate polynomials
    over a finite field. The minimum, maximum, and average number of
    arithmetic operations both on polynomials and in the ground field
    are derived.",
  paper = "Ma90.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Maxx90,
  author = "Ma, Kwan-Liu and Kessler, Robert R.",
  title = {{TICL -- A Type Inference System for Common Lisp}},
  journal = "Software -- Practice and Experience",
  volume = "20",
  number = "6",
  pages = "593-623",
  year = "1990",
  abstract =
    "Most current Common Lisp compilers generate more efficient code
    when supplied with data type information. However, in keeping with
    standard Lisp programming style, most programmers are reluctant to
    provide type information; they simply allow the run-time type
    system to managed the data types accordingly. To fill this gap, we
    have designed and implemented a type inference system for Common
    Lisp (TICL). TICL takes a Lisp program that has been annotated
    with a few type declarations, adds as many declarations as
    possible, and produces a type declared program. The compiler can
    then use this information to generate more efficient
    code. Measurements indicate that a 20 per cent speed improvement
    can generally be achieved.",
  paper = "Maxx90.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Mack01,
  author = "MacKenzei, Donald",
  title = {{Mechanizing Proof: Computing, Risk, and Trust}},
  year = "2001",
  publisher = "MIT Press",
  isbn = "0-262-13393-8",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Mack02,
  author = "Mackie, Ian and Pinto, Jorge Sousa",
  title = {{Encoding Linear Logic with Interaction Combinators}},
  journal = "Information and Computation",
  volume = "176",
  pages = "153-186",
  year = "2002",
  abstract =
    "The purpose of this paper is to demonstrate how Lafont's
    interaction combinators, a system of three symbols and six
    interaction rules, can be used to encode linear
    logic. Specifically, we give a translation of the multiplicative,
    exponential, and additive fragments of linear logic together with
    a strategy for cut-elimination which can be faithfully
    simulated. Finally, we show briefly how this encoding can be used
    for evaluating $\lambda$-terms. In addition to offering a very
    simple, perhaps the simplest, system of rewriting for linear logic
    and the $\lambda$-calculus, the interaction net implementation
    that we present has been shown by experimental testing to offer a
    good level of sharing in terms of the number of cut-elimination
    steps (resp. $\beta$-reduction steps). In particular it performs
    better than all extant finite systems of interaction nets.",
  paper = "Mack02.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Macl91,
  author = "MacLane, Saunders",
  title = {{Categories for the Working Mathematician}},
  publisher = "Springer",
  year = "1991",
  isbn = "0-387-98403-8",
  link = "\url{http://www.maths.ed.ac.uk/~aar/papers/maclanecat.pdf}",
```

```
  paper = "Macl91.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Macl92,
  author = "MacLane, Saunders",
  title = {{Sheaves in Geometry and Logic: A First Introduction to Topos
           Theory}},
  year = "1992",
  isbn = "978-0-387-97710-2",
  publisher = "Springer"
}
```

———-

— **axiom.bib** —

```
@misc{Maco15,
  author = "Macor, Jackson",
  title = {{A Brief Introduction to Type Theory and the Univalence
            Axiom}},
  year = "2015",
  comment = "U. Chicago REU",
  link = "\url{https://math.uchicago.edu/~may/REU2015/REUPapers/Macor.pdf}",
  abstract =
    "In this paper, we will introduce the basic concepts and notation
    of modern type theory in an informal manner. We will discuss
    functions, type formation, the nature of proof, and proceed to
    prove some basic results in sentential logic. These efforts will
    culminate in a proof of the axiom of choice in type theory. We
    will further introduce a few concepts in homotopy type theory, a
    modern invention which sees to provide a foundation of mathematics
    without ZFC type theory. We will conclude with the univalence
    axiom, an indispensable tool in homotopy type theory, and use it
    to prove a stronger version of the axiom of choice.",
  paper = "Maco15.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Macq84,
  author = "MacQueen, David",
  title = {{Modules for Standard ML}},
  booktitle = "Conf. on Lisp and Functional Programming",
  publisher = "ACM Press",
```

```
  year = "1984",
  pages = "198-207",
  abstract =
    "The functional programming language ML has been undergoing a thorough
    redesign during the past year, and the module facility described here
    has been proposed as part of the revised language, now called Standard
    ML. The design has three main goals: (1) to facilitate the structuring
    of large ML programs; (2) to support separate compilation and generic
    library units; and (3) to employ new ideas in the semantics of data
    types to extend the power of ML's polymorphic type system. It is based
    on concepts inherent in the structure of ML, primarily the notions of
    a declaration, its type signature, and the environment that it
    denotes.",
  paper = "Macq84.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Macq86,
  author = "MacQueen, David",
  title = {{Using Dependent Types to Express Modular Structure}},
  booktitle = "Principles of Programming Languages POPL'13",
  publisher = "ACM Press",
  pages = "277-286",
  year = "1986",
  link =
   "\url{https://people.mpi-sws.org/~dreyer/courses/modules/macqueen86.pdf}",
  paper = "Macq86.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Macq88,
  author = "MacQueen, David",
  title = {{An Implementation of Standard ML Modules}},
  booktitle = "Lisp and Functional Programming '88",
  year = "1988",
  pages = "212-223",
  publisher = "ACM Press",
  abstract =
    "Standard ML includes a set of module constructs that support
    programming in the large. These constructs extend ML's basic
    polymorphic type system by introducing the dependent types of Martin
    L/Sf's Intuitionistic Type Theory. This paper discusses the problems
    involved in implementing Standard ML's modules and describes a
    practical, efficient solution to these problems. The representations
    and algorithms of this implementation were inspired by a detailed
    formal semantics of Standard ML developed by Milner, Tofte, and
```

```
    Harper. The implementation is part of a new Standard ML compiler that
    is written ia Standard ML using the module system.",
  paper = "Macq88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Macq94,
  author = "MacQueen, David and Tofte, Mads",
  title = {{A Semantics for Higher-order Functors}},
  journal = "LNCS",
  volume = "788",
  pages = "409-423",
  year = "1994",
  abstract =
    "Standard ML has a module system that allows one to define parametric
    modules, called functors.  Functors are ``first-order,'' meaning
    that functors themselves cannot be passed as parameters or returned
    as results of functor applications.  This paper presents a semantics
    for a higher-order module system which generalizes the module system
    of Standard ML.  The higher-order functors described here are
    implemented in the current version of Standard ML of New Jersey and
    have proved useful in programming practice.",
  paper = "Macq94.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Macqxx,
  author = "MacQueen, David B.",
  title = {{Reflections on Standard ML}},
  year = "unknown",
  abstract =
    "Standard ML is one of a number of new programming languages
    developed in the 1980s that are seen as suitable vehicles for
    serious systems and applications programming. It offers an
    excellent ratio of expressiveness to language complexity, and
    provides competitive efficiency. Because of its type and module
    system, Standard ML manages to combine safety, security, and
    robustness with much of the flexibility of dynamically typed
    languages like Lisp. It is also has the most well-developed
    scientific foundation of any major language. Here I review the
    strengths and weaknesses of Standard ML and describe some of what
    we have learned through the design, implementation, and use of the
    language.",
  paper = "Macqxx.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Mads80,
  author = "Madsen, Ole Lehrmann",
  title = {{On Defining Semantics by means of Extended Attribute Grammars}},
  journal = "LNCS",
  volume = "54",
  pages = "259-299",
  year = "1980",
  paper = "Mads80.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mahb13,
  author = "Mahboubi, Assia",
  title = {{The Rooster and the Butterflies}},
  journal = "LNCS",
  volume = "7961",
  year = "2013",
  abstract =
    "This paper describes a machine-checked proof of the Jorden-Holder
    theorem for finite groups. This purpose of this description is to
    discuss the representation of the elementary concepts of finite
    group theory inside type theory. The design choices underlying
    these representations were crucial to the successful formalization
    of a complete proof of the Odd Order Theorem in the Coq system.",
  paper = "Mahb13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Maje94,
  author = "Majewski, Bohdan and Havas, George",
  title = {{The Complexity of Greatest Common Divisor Computations}},
  journal = "LNCS",
  volume = "877",
  pages = "184-193",
  year = "1994",
  abstract =
    "We study the complexity of expressing the greatest common divisor
    of $n$ positive numbers as a linear combination of the
    numbers. We prove the NP-completeness of finding an optimal set of
    multipliers with respect to either of $L_0$ metric or the
```

```
    $L_\infty$ norm. We present and analyze a new method for
    expressing the gcd of $n$ numbers as their linear combination and
    give an upper bound on the size of the largest multiplier
    produced by this method, which is optimal.",
  paper = "Maje94.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Makk07,
  author = "Makkai, Michael",
  title = {{Formal Specification of Computer Programs}},
  year = "2007",
  link = "\url{https://www.math.mcgill.ca/makkai/MATH3182007/31807552.pdf}",
  paper = "Makk07.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Mane76,
  author = "Manes, Ernest G.",
  title = {{Algebraic Theories}},
  publisher = "Springer",
  year = "1976",
  series = "Graduate Texts in Mathematics",
  isbn = "978-1-9860-1"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Mann68,
  author = "Manna, Zohar",
  title = {{Termination of Algorithms}},
  school = "Carnegie Mellon University",
  year = "1968",
  link = "\url{http://apps/dtic.mil/dtic/tr/fulltext/u2/670558.pdf}",
  abstract =
    "The thesis contains two parts which are self-contained units.

    In Part I we present several results on the relation between
    \begin{enumerate}
    \item the problem of termination and equivalence of programs and
    abstract programs, and
```

```
    \item the first order predicate calculus
    \end{enumerate}

    Part II is concerned with the relation between
    \begin{enumerate}
    \item the termination of interpreted graphs, and
    \item properties of well-ordered sets and graph theory
    \end{enumerate}",
  paper = "Mann68.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mann78,
  author = "Manna, Zohar and Waldinger, Richard",
  title = {{The Logic of Computer Programming}},
  journal = "IEEE Trans. on Software Engineering",
  volume = "4",
  number = "3",
  year = "1978",
  abstract =
    "Techniques derived from mathematical logic promise to provide an
    alternative to the conventional methodology for constructing,
    debugging, and optimizing computer programs. Ultimately, these
    techniques are intended to lead to the automation of many of the
    facets of the programming process.

    This paper provides a unified tutorial exposition of the logical
    techniques, illustrating each with examples. The strengths and
    limitations of each technique as a practial programming aid are
    assessed and attempts to implement these methods in experimental
    systems are discussed.",
  paper = "Mann78.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Mann78a,
  author = "Manna, Zohar and Waldinger, Richard",
  title = {{Is 'sometime' sometimes better than 'always'?}},
  journal = "CACM",
  volume = "21",
  number = "2",
  year = "1978",
  abstract =
    "This paper explores a technique for proving the correctness and
    termination of programs simultaneously. This approach, the
```

```
    {\sl intermittent-assertion method}, involves documenting the
    program with assertions that must be true at some time when
    control passes through the corresponding point, but that need not
    be true every time. The method, introduced by Burstall, promises
    to provide a valuable complement to the more conventional methods.

    The intermittent-assertion method is presented with a number of
    examples of correctness and termination proofs. Some of these
    proofs are markedly simpler than their conventional
    counterparts. On the other hand, it is shown that a proof of
    correctness or termination by any of the conventional techniques
    can be rephrased directly as a proof using intermittent
    assertions. Finally it is shown how the intermittent assertion
    method can be applied to prove the validity of program
    transformations and the correctness of continuously operating
    systems.",
  paper = "Mann78a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Mann80,
  author = "Manna, Zohar",
  title = {{Lectures on the Logic of Computer Programming}},
  publisher = "Society for Industrial and Applied Mathematics",
  year = "1980",
  paper = "Mann80.tgz",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Maox20,
  author = "Mao, Lei",
  title = {{Euclidean Algorithm}},
  link = "\url{https://leimao.github.io/blog/Euclidean-Algorithm}",
  year = "2020"
}
```

———-

— **axiom.bib** —

```
@misc{Mapl18,
  author = "Maplesoft",
  title = {{Maple Computer Algebra Software}},
```

```
  year = "2018",
  link = "\url{https://www.maplesoft.com}"
}
```

———-

— **axiom.bib** —

```
@book{Marc77,
  author = "Marcus, Daniel A.",
  title = {{Number Fields}},
  publisher = "Springer",
  year = "1977",
  isbn = "978-0387902791"
}
```

———-

— **axiom.bib** —

```
@article{Mart83,
  author = "Marti, Jed",
  title = {{The Little META Translator Writing System}},
  journal = "Software -- Practice and Experience",
  volume = "13",
  pages = "941-959",
  year = "1983",
  abstract =
    "The Little META Translator Writing System is a tool for
    implementing experimental compilers, interpreters, preprocessors
    and higher level translator writing systems on a
    microprocessor. This paper presents the salient features of the
    system through the implementation of three translators. Some
    knowledge of LISP is assumed.",
  paper = "Mart83.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mart73,
  author = "Martin-Lof, P.",
  title = {{An Intuitionistic Theory of Types}},
  booktitle = "Logic Colloquium 1973",
  publisher = "H.E. Rose and J.G. Shepherdson",
  pages = "73-118",
  year = "1973",
  abstract =
    "The theory of types with which we shall be concerned is intended to
```

```
    be a full scale system for foralizing intuitionistic mathematics as
    developed, for example, in the book by Bishop 1967. The language of
    the theory is richer than the language of first order predicate
    logic. This makes it possible to strengthen the axioms for existence
    and disjunction. In the case of existence, the possibility of
    strengthening the usual elimination rule seems first to have been
    indicated by Howard 1969, whose proposed axioms are special cases of
    the existential elimination rule of the present theory. Furthermore,
    there is a reflection principle which links the generation of objects
    and types and plays somewhat the same role for the present theory as
    does the replacement axiom for Zermelo-Frankel set theory.

    An earlier, not yet conclusive, attempt at formulating a theory of
    this kind was made by Scott 1970. Also related, although less closely,
    are the type and logic free theories of consructions of Kreisel 1962
    and 1965 and Goodman 1970.

    In its first version, the present theory was based on the strongly
    impredicative axiom that there is a type of all types whatsoever,
    which is at the same time a type and an object of that type. This
    axiom had to be abandoned, however, after it was shown to lead to a
    contradiction by Jean Yves Girard. I am very grateful to him for
    showing me his paradox. The change that it necessitated is so drastic
    that my theory no longer contains intuitionistic simple type theory as
    it originally did. Instead, its proof theoretic strength should be
    close to that of predicative analysis.",
  paper = "Mart73.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@inproceedings{Mart79,
  author = "Martin-Lof, P.",
  title = {{Constructive Mathematics and Computer Programming}},
  booktitle = "Proc. 6th Int. Congress for Logic, Methodology and
               Philosophy of Science",
  publisher = "North-Holland",
  year = "1979",
  pages = "153-179",
  abstract =
    "If programming is understood not as the writing of instructions for
    this or that computing machine but as the design of methods of
    computation that it is the computer's duty to execute (a difference
    that Dijkstra has referred to as the difference between computer
    science and computing science), then it no longer seems possible to
    distinguish the discipline of programming from constructive
    mathematics. This explains why the intuitionistic theory of types,
    which was originally developed as a symbolism for the precise
    codification of constructive mathematics, may equally well be viewed
    as a programming language. As such it provides a precise notation not
```

```
    only, like other programming languages, for the programs themselves
    but also for the tasks that the programs are supposed to
    perform. Moreover, the inference rules of the theory of types, which
    are again completely formal, appear as rules of correct program
    synthesis. Thus the correctness of a program written in the theory of
    types is proved formally at the same time as it is being
    synthesized.",
  paper = "Mart79.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Mart84,
  author = "Martin-Lof, P.",
  title = {{Constructive Mathematics and Computer Programming}},
  journal = "Phil. Trans. Royal Society London",
  volume = "312",
  year = "1984",
  pages = "501-518",
  link =
    "\url{https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.1984.0073}",
  abstract =
    "If programming is understood not as the writing of instructions for
    this or that computing machine but as the design of methods of
    computation that it is the computers duty to execute (a difference
    that Dijkstra has referred to as the difference between computer
    science and computing science), then it no longer seems possible to
    distinguish the discipline of programming from constructive
    mathematics.  This explains why the intuitionistic theory of types
    (Martin-Lof 1975 In Logic Colloquium 1973 (ed.  H. E. Rose and
    J. C. Shepherdson), pp.  73 - 118 .  Amsterdam: North- Holland), which
    was originally developed as a symbolism for the precise codification
    of constructive mathematics, may equally well be viewed as a
    programming language.  As such it provides a precise notation not
    only, like other programming languages, for the programs themselves
    but also for the tasks that the programs are supposed to perform.
    Moreover, the inference rules of the theory of types, which are again
    completely formal, appear as rules of correct program synthesis. Thus
    the correctness of a program written in the theory of types is proved
    formally at the same time as it is being synthesized.",
  paper = "Mart84.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Mart97a,
```

```
  author = "Martzloff, J.C.",
  title = {{A History of Chinese Mathematics}},
  publisher = "Springer Verlag",
  year = "1997"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Maur17,
  author = "Maurer, Luke and Downen, Paul and Ariola, Zena M.
            and Jones, Simon Peyton",
  title = {{Compiling without Continuations}},
  booktitle = "PLDI'17",
  publisher = "ACM",
  year = "2017",
  abstract =
    "Many fields of study in compilers give rise to the concept of a
     {\sl joint point} -- a place where different execution paths come
     together. Join points are often treated as functions or
     continuations, but we believe it is time to study them in their
     own right. We show that adding join points to a direct-style
     functional intermediate language is a simple but powerful change
     that allows new optimizations to be performed, including a
     significant improvement to list fusion. Finally, we report on
     recent work on adding join points to the intermediate language of
     the Glasgow Haskell Compiler.",
  paper = "Maur17.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Mcbr99,
  author = "McBride, Conor",
  title = {{Dependently Typed Functional Programs and their Proofs}},
  school = "University of Edinburgh",
  year = "1999",
  link = "\url{http://strictlypositive.org/thesis.pdf}",
  abstract =
    "Research in dependent type theories has, in the past, concentrated on
    its use in the presentation of theorems and theorem-proving. This
    thesis is concerned mainly with the exploitation of the computational
    aspects of type theory for programming, in a context where the
    properties of programs may readily be specified and established. In
    particular, it develops technology for programming with dependent
    inductive families of datatypes and proving those programs correct. It
    demonstrates the considerable advantage to be gained by indexing data
    structures with pertinent characteristic information whose soundness
    is ensured by typechecking, rather than human effort.
```

Type theory traditionally presents safe and terminating computation on
inductive datatypes by means of elimination rules which serve as
induction principles and, via their associated reduction behaviour,
recursion operators. In the programming language arena, these appear
somewhat cumbersome and give rise to unappealing code, complicated by
the inevitable interaction between case analysis on dependent types
and equational reasoning on their indices which must appear explicitly
in the terms. Thierry Coquand's proposal to equip type theory directly
with the kind of pattern matching notation to which functional
programmers have become used to over the past three decades offers a
remedy to many of these difficulties. However, the status of pattern
matching relative to the traditional elimination rules has until now
been in doubt. Pattern matching implies the uniqueness of identity
proofs, which Martin Hofmann showed underivable from the conventiaonal
definition of equality. This thesis shows that the adoption of this
uniqueness as axiomatic is sufficient to make pattern matching
admissible.

A datatype's elimination rule allows abstraction only over the whole
inductively defined family. In order to support pattern matching, the
application of such rules to specific instances of dependent families
has been systematised. The underlying analysis extends beyond
datatypes to other rules of a similar second order character,
suggesting they may have other roles to play in the specification,
verification and, perhaps, derivation of programs. The technique
developed shifts the specificity from the instantiation of the type's
indices into equational constraints on indices freely chosen, allowing
the elimination rule to be applied.

Elimination by this means leaves equational hypotheses in the
resulting subgoals, which must be solved if further progress is to be
made. The first-order unification algorithm for constructor forms in
simmple types presented in [McB96] has been extended to cover
dependent datatypes as well, yielding completely automated solution of
a class of problems which can be syntactically defined.

The justification and operation of these techniques requires the
machine to construct and exploit a standardised collection of
auxiliary lemmas for each datatype. This is greatly facilitated by two
technical developments of interest in their own right:
\begin{itemize}
\item a more convenient definition of equality, with a relaxed
formulation rule allowing elements of different types to be compared,
but nonetheless equivalent to the usual equality plus the axiom of
uniqueness
\item a type theory, OLEG, which incorporates incomplete objects,
accounting for their 'holes' entirely within the typing judgments and,
novelly, not requiring any notion of explicit substitution to manage
their scopes.
\end{itemize}

A substantial prototype has been implemented, extending the proof
assistant LEGO. A number of programs are developed by way of

```
    example. Chiefly, the increased expressivity of dependent datatypes is
    shown to capture a standard first-order unification algorithm within
    the class of structurally recursive programs, removing any need for a
    termination argument. Furthermore, the use of elimination rules in
    specifying the components of the program simplifies significantly its
    correctness proof.",
  paper = "Mcbr99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Mcbr20,
  author = "McBride, Conor",
  title = {{Epigram 2 - Autopsy, Obituary, Apology}},
  year = "2020",
  link = "\url{https://vimeo.com/428161108}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Mcca60a,
  author = "McCarthy, J.",
  title = {{Programs with Common Sense}},
  year = "1960",
  abstract =
    "Interesting work is being done in programming computers to solve
    problems which require a high degree of intelligence in
    humans. However, certain elementary verbal reasoning processes so
    simple that they can be carried out by any non-feeble-minded human
    have yet to be simulated by machine programs.

    This paper will discuss programs to manipulate in a suitable
    formal language (most likely a part of the predicate calculus)
    common instrumental statements. The basic program will draw
    immediate conclusions from a list of premises. These conclusions
    will be either declarative or imperative sentences. When an
    imperative sentence is deduced the program takes a corresponding
    action. These actions may include printing sentences, moving
    sentences on lists, and reinitiating the basic deduction process
    on these lists.

    Facilities will be provided for communication with humans in the
    system via manual intervention and display devices connected to
    the computer.",
  paper = "Mcca60a.pdf",
  keywords = "DONE"
```

```
}
```

———-

— **axiom.bib** —

```
@book{Mcca62,
  author = "McCarthy, John and Abrahams, Paul W. and Edwards, Daniel J.
            and Hart, Timothy P. and Levin, Michael I.",
  title = {{LISP 1.5 Programmer's Manual}},
  link = "\url{http://www.softwarepreservation.org/projects/LISP/book/LISP%201.5%20Programmers%20Manual.pdf}",
  publisher = "MIT Press",
  isbn = "0-262-13011-4",
  year = "1962"
}
```

———-

— **axiom.bib** —

```
@book{Mcco79,
  author = "McCorduck, Pamela",
  title = {{Machines Who Think}},
  year = "1979",
  publisher = "Freeman",
  keywords = "shelf, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Mcke14,
  author = "McKenna, Brian",
  title = {{Idris: Practical Dependent Types with Practical Examples}},
  year = "2014",
  link = "\url{https://www.youtube.com/watch?v=4i7KrG1Afbk}",
  abstract =
    "Dependent types turn types into a first-class language construct and
    allows types to be predicated upon values. Allowing types to be
    controlled by ordinary values allows us to prove many more properties
    about our programs and enables some interesting forms of
    metaprogramming. We can do interesting things like write a type-safe
    printf or verify algebraic laws of data structures - but while
    dependent types are quickly gaining in popularity, it can be tricky to
    find examples which are both useful and introductory.

    This talk will demonstrate some practical dependent typing examples
    (and not sized vectors) using Idris, a language designed for writing
    executable programs, rather than just proving theorems."
}
```

———-

— **axiom.bib** —

```
@article{Meie02,
  author = "Meier, Andreas and Sorge, Volker and Colton, Simon",
  title = {{Employing Theory Formationi to Guide Proof Planning}},
  journal = "LNCS",
  volume = "2385",
  year = "2002",
  abstract =
    "The invention of suitable concepts to characterise mathematical
    structures is one of the most challenging tasks for both human
    mathematicians and automated theorem provers alike. We present an
    approach where automatic concept formation is used to guide
    non-isomorphism proofs in the residue class domain. The main idea
    behind the proof is to automatically identify discriminants for
    two given structures to show that they are not
    isomorphic. Suitable discriminants are generated by a theory
    formation system; the overall proof is constructe by a proof
    planner with the additional support of traditional automated
    theorem provers and a computer algebra system.",
  paper = "Meie02.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Meij91,
  author = "Meijer, Erik and Fokkinga, Maarten and Paterson, Ross",
  title = {{Functional Programming with Bananas, Lenses, Envelopes and
           Barbed Wire}},
  link =
    "\url{http://eprints.eemcs.utwente.nl/7281/01/db-utwente-40501F46.pdf}",
  abstract = "
    We develop a calculus for lazy functional programming based on
    recursion operators associated with data type definitions. For these
    operators we derive various algebraic laws that are useful in deriving
    and manipulating programs. We shall show that all example functions in
    Bird and Wadler's ``Introduction to Functional Programming'' can be
    expressed using these operators.",
  paper = "Meij91.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Meilxx,
```

```
  author = "Meili, Mario",
  title = {{Polymorphic Lambda Calculus}},
  year = "unknown",
  link = "\url{http://wiki.ifs.hsr.ch/SemProgAnTr/files/mmeili_polymorphic_lambda_calculus_final.pdf}",
  paper = "Meilxx.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Miet15,
  author = "Mietek, Bak",
  title = {{Build Your Own Proof Assistant}},
  year = "2015",
  link = "\url{https://www.youtube.com/watch?v=7u-jx1UyRmc}"
}
```

———-

— **axiom.bib** —

```
@inbook{Mign82,
  author = "Mignotte, M.",
  title = {{Some Useful Bounds}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "259-264",
  abstract =
    "Some fundamental inequalities for the following values are
    listed: the determinant of a matrix, the absolute value of the
    roots of a polynomial, the coefficients of divisors of
    polynomials, and the minimal distance between the roots of a
    polynomial. These inequalities are useful for the analysis of
    algorithms in various areas of computer algebra.",
  paper = "Buch82.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Meye86,
  author = "Meyer, Albert R. and Reinhold, Mark B.",
  title = {{Type is not a type}},
  booktitle = "POPL 86",
  pages = "287-295",
  year = "1986",
```

```
  abstract =
    "A function has a dependent type when the type of its result
    depends upon the value of its argument. Dependent types originated in
    the type theory of intuitionistic mathematics and have reappeared
    independently in programming languages such as CLU, Pebble, and
    Russell. Some of these languages make the assumption that there exists
    a type-of-all-types which is its own type as well as the type
    of all other types. Girard proved that this approach is inconsistent
    from the perspective of intuitionistic logic. We apply Girard's
    techniques to establish that the type-of-all-types assumption creates
    serious pathologies from a programming perspective: a system using
    this assumption is inherently not normalizing, term equality is
    undecidable, and the resulting theory fails to be a conservative
    extension of the theory of the underlying base types. The failure of
    conservative extension means that classical reasoning about programs
    in such a system is not sound.",
}
```

———-

— **axiom.bib** —

```
@book{Meye88,
  author = "Meyer, Bertrand",
  title = {{Object-Oriented Software Construction}},
  year = "1988",
  publisher = "Prentice Hall",
  link = "\url{https://sophia.javeriana.edu.co/~cbustaca/docencia/POO-2016-01/documentos/Object%20Oriented%20Sof
  paper = "Meye88.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mham11,
  author = "Mhamdi, Tarek and Hasan, Osman and Tahar, Sofiene",
  title = {{Formalization of Entropy Measure in HOL}},
  booktitle = "Interactve Theorem Proving",
  publisher = "Springer",
  pages = "233-248",
  year = "2011"
}
```

———-

— **axiom.bib** —

```
@book{Mich76,
  author = "Michaelson, S. and Milner, R.",
  title = {{Automata Languages and Programming}},
```

```
  year = "1976",
  publisher = "Edinburgh University Press",
  isbn = "0-85224-308-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Mill06,
  author = "Mills, Bruce",
  title = {{Theoretical Introduction to Programming}},
  publisher = "Springer",
  year = "2006",
  isbn = "1-84628-021-4",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Miln94,
  author = "Milne, Peter",
  title = {{Classical Harmony: Rules of Inference and the Meaning of
            the Logical Constants}},
  journal = "Synthese",
  volume = "100",
  number = "1",
  pages = "49-94",
  year = "1994",
  abstract =
    "The thesis that, in a system of natural deduction, the meaning of
    a logical constant is given by some or all of its introduction and
    elimination rules has been developed recently by Dummett, Prawitz,
    Tennant, and others, by the addition of harmony constraints.
    Introduction and elimination rules for a logical constant must be
    in harmony. By deploying harmony constraints, these authors have
    arrived at logics no stronger than intuitionistic propositional
    logic. Classical logic, they maintain, cannot be justified from
    this proof-theoretic perspective. This paper argues that, while
    classical logic can be formulated so as to satisfy a number of
    harmony constraints, the meanings of the standard logical
    constants cannot all be given by their introduction and/or
    elimination rules; negation, in particular, comes under close
    scrutiny.",
  paper = "Miln94.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Miln73,
  author = "Milner, Robin",
  title = {{Models of LCF}},
  type = "technical report",
  institution = "Stanford Artificial Intelligence Laboratory",
  number = "STAN-CS-73-332",
  year = "1973",
  link = "\url{http://i.stanford.edu/pub/cstr/reports/cs/tr/73/332/CS-TR-73-332.pdf}",
  abstract =
    "LCF is a deductive system for computable functions proposed by
    D. Scott in 1969 in an unpublished memorandum. The purpose of the
    present paper is to demonstrate the soundness of the system with
    respect to certain models, which are partially ordered domains of
    continuous functions. This demonstration was supplied by Scott in
    his memorandum; the present paper is merely intended to make this
    work more accessible.",
  paper = "Miln73.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Miln90,
  author = "Milner, Robin and Torte, Mads and Harper, Robert",
  title = {{The Definition of Standard ML}},
  publisher = "Lab for Foundations of Computer Science, Univ. Edinburgh",
  link = "\url{http://sml-family.org/sml90-defn.pdf}",
  year = "1990",
  paper = "Miln90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Miln91,
  author = "Milner, Robin and Torte, Mads",
  title = {{Commentary on Standard ML}},
  publisher = "Lab for Foundations of Computer Science, Univ. Edinburgh",
  link = "\url{https://pdfs.semanticscholar.org/d199/16cbbda01c06b6eafa0756416e8b6f15ff44.pdf}",
  year = "1991",
  paper = "Miln91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Miln97,
  author = "Milner, Robin and Torte, Mads and Harper, Robert and
            MacQueen, David",
  title = {{The Definition of Standard ML (Revised)}},
  publisher = "Lab for Foundations of Computer Science, Univ. Edinburgh",
  link = "\url{http://sml-family.org/sml97-defn.pdf}",
  year = "1997",
  paper = "Miln97.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Mimr20,
  author = "Mimram, Samuel",
  title = {{Program = Proof}},
  publisher = "Polytechnique.fr",
  year = "2020",
  link = "\url{www.lix.polytechnique.fr/Labo/Samuel.Mimram/teaching/IFN551/course.pdf}",
  paper = "Mimr20.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Miol74,
  author = "Miola, Alfonso and Yun, David Y.Y.",
  title = {{Computational Aspects of Hensel-type Univariate Polynomial
            Greatest Common Divisor Algorithms}},
  journal = "ACM Sigplan Bulletin",
  year = "1974",
  abstract =
    "Two Hensel-type univariate polynomial Greatest Common Divisor
    (GCD) algorithms are presented and compared. The regular linear
    Hensel construction is shown to be generally more efficient than
    the Zassenhaus quadratic contstruction. The UNIGCD algorithm for
    UNIvariate polynomial GCD computations, based on the regular
    Hensel construction is then presented and compared with the
    Modular algorithm based on the Chinese Remainder Algorithm. From
    both an analytical and an experimental point of view, the UNIGCD
    algorithm is shown to be preferable for many common univariate GCD
    computations. This is true even for dense polynomials, which was
    considered to be the most suitable case for the application of the
    Modular algorithm.",
  paper = "Miol74.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mitc88a,
  author = "Mitchell, John C.",
  title = {{Polymorphic Type Inference and Containment}},
  journal = "Information and Computation",
  volume = "76",
  number = "2-3",
  year = "1988",
  pages = "211-249",
  abstract =
    "Type expressions may be used to describe the functional behavior of
    untyped lambda terms. We present a general semantics of polymorphic
    type expressions over models of untyped lambda calculus and give
    complete rules for inferring types for terms. Some simplified typing
    theories are studied in more detail, and containments between types
    are investigated.",
  paper = "Mitch88a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mitc91,
  author = "Mitchell, John C.",
  title = {{TYpe Inference with Simple Subtypes}},
  journal = "J. of Functional Programming",
  volume = "1",
  number = "3",
  year = "1991",
  pages = "245-285",
  abstract =
    "Subtyping appears in a variety of programming languages, in the form
    of the automatic coercion of integers to reals, Pascal subranges,
    and subtypes arising from class hierarchies in languages with
    inheritance. A general framework based on untyped lambda calculus
    provides a simple semantic model of subtyping and is used to
    demonstrate that an extension of Curry's type inference rules are
    semantically complete. An algorithm G for computing the most general
    typing associated with any given expression, and a restricted,
    optimized algorithm GA using only atomic subtyping hypotheses are
    developed. Both algorithms may be extended to insert type conversion
    functions at compile time or allow polymorphic function declarations
    as in ML.",
  paper = "Mitc91.pdf"
}
```

———-

— **axiom.bib** —

```
@InCollection{Mitc91a,
  author = "Mitchell, John C.",
  title = {{Type Systems for Programming Languages}},
  booktitle = "Handbook of Theoretical Computer Science (Vol B.)",
  pages = "365-458",
  year = "1991",
  publisher = "MIT Press",
  isbn = "0-444-88074-7",
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mitc91b,
  author = "Mitchell, John and Meidal, Sigurd and Madhav, Neel",
  title = {{An Extension of Standard ML Modules with Subtyping and
           Inheritance}},
  booktitle = "POPL'91",
  pages = "270-278",
  year = "1991",
  isbn = "0-89791-419-8",
  abstract =
    "We describe a general module language integrating abstract data
    types, specifications and object-oriented concepts.  The framework is
    based on the Standard ML module system, with three main extensions:
    subtyping, a form of object derived from ML structures, and inheritence
    primitives.  The language aims at supporting a range of
    programming styles, including mixtures of object-oriented programming
    and programs built around specified algebraic or higher-order abstract
    data types.  We separate specification from implementation,
    and provide independent inheritance mechanisms for each.  In order to
    support binary operations on objects within this framework, we
    introduce ``internal interfaces// which govern the way that function
    components of one structure may access components of another.  The
    language design has been tested by writing a number of program examples;
    an implementation is under development in the context of a
    larger project.",
  paper = "Mitc91b.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Moba09,
  author = "Mobarakeh, S. Saeidi",
  title = {{Type Inference Algorithms}},
  year = "2009",
  link = "\url{https://www.win.tue.nl/~hzantema/semssm.pdf}",
```

```
  abstract =
    "In this paper we are going to describe the Wands type inference
    algorithm and well try to extend this algorithm with the notion of
    polymorphic let.  By means of a type system, which were going to
    extend with some constraint language, we are able to extend the
    algorithms first phase (generation of equations) with
    let-polymorphism.  The second phase of the algorithm (solving of the
    generated equations) needs some modifications to be done on standard
    unification algorithms because the new generated equation constructs
    can not directly be fed in to the standard unification algorithm.  The
    correctness of the first phase of the algorithm is been proved by
    extending the Wands soundness and completeness prove. Finally a
    simple example is given to clarify the idea behind the algorithm.",
  paper = "Moba09.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Moll19,
  author = "Moller, Anders and Schwartzbach, Michael I.",
  title = {{Static Program Analysis}},
  publisher = "Aarhus University",
  year = "2019"
}
```

———-

— **axiom.bib** —

```
@article{Mona12,
  author = "Monagan, Michael and Pearce, Roman",
  title = {{POLY: A New Polynomial Data Structure for Maple 17}},
  journal = "Communications in Computer Algebra",
  publisher = "ACM",
  volume = "46",
  number = "4",
  year = "2012",
  abstract =
    "We demonstrate how a new data structure for sparse distributed
    polynomials in the Maple kernel significantly accelerates a large
    subset of Maple library routines The POLY data structure and its
    associated kernel operations (degree, coeff, subs, has, diff,
    eval,...) are programmed for high scalability, allowing
    polynomials to have hundreds of millions of terms, and very low
    overhead, increasing parallel speedup in existing routines and
    improving the performance of high live Maple library routines.",
  paper = "Mona12.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Moni03,
  author = "Monin, Jean-Francois",
  title = {{Understanding Formal Methods}},
  publisher = "Springer",
  year = "2003",
  isbn = "1-85233-247-6",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Monk76,
  author = "Monk, J. Donald",
  title = {{Mathematical Logic}},
  publisher = "Springer",
  year = "1976",
  isbn = "978-1-4684-9452-5",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Mont12,
  author = "Montanaro, Ashley",
  title = {{Computational Complexity Lecture Notes}},
  year = "2012",
  link = "\url{https://people.maths.bris.ac.uk/~csxam/teaching/cc-lecturenotes.pdf}",
  paper = "Mont12.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Moor18,
  author = "Moore, Brandon and Pena, Lucas and Rosu, Grigore",
  title = {{Program Verification by Coinduction}},
  booktitle = "Programming Languages and Systems",
  publisher = "Springer",
  year = "2018",
  pages = "589-618",
  link = "\url{https://link.springer.com/content/pdf/10.1007%2F978-3-319-89884-1.pdf}",
  abstract =
```

```
    "We present a novel program verification approach based on
    coinduction, which takes as input an operational semantics. No
    intermediates like program logics or verifcation condition
    generators are needed. Specifications can be written using any
    state predicates. We implement our approach in Coq, giving a
    certifying language-independent verification framework. Our proof
    system is implemented as a single module imported unchanged into
    language-specific proofs. Automation is reached by instantiating a
    generic heuristic with language-specific tactics. Manual
    assistance is also smoothly allowed at points the automation
    cannot handle. We demonstrate the power and versatility of our
    approach by verifying algoirthms as complicated as Schorr-Waite
    graph marking and instantiating our framework for object languages
    in several styles of semantics. Finally, we show that our
    coinductive approach subsumes reachability logic, a recent
    language-independent soudn and (relatively) complete logic for
    program verification that has been instantiated with operational
    semantics of languages as complex as C, Java and Javascript.",
  paper = "Moor18.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Mour15,
  author = "de Moura, Leonardo and Avigad, Jeremy and Kong, Soonho
            and Roux, Cody",
  title = {{Elaboration in Dependent Type Theory}},
  link = "\url{https://arxiv.org/pdf/1505.04324.pdf}",
  year = "2015",
  abstract =
    "To be usable in practice, interactive theoremprovers need to
    provide convenient and efficient means of writing expressions,
    definitions, and proofs. This involves inferring information that
    is often left implicit in an ordinary mathematical text, and
    resolving ambiguities in mathematical expressions. We refer to the
    rpocess of passing from a quasi-formal and partially-specified
    expression to a completely precise formal one as {\sl
    elaboration}. We describe an elaboration algorithm for dependent
    type theory that has been implemented in the Lean theorem
    prover. Lean's elaborator supports higher-order unification, type
    class inference, ad hoc overloading, insertion of coercions, the
    use of tactics, and the computational reduction of terms. The
    interactions between these components are subtle and complex, and
    the elaboration algorithm has been carefully designed to balance
    efficiency and usability. We describe the central design goals,
    and the means by which they are achieved.",
  paper = "Mour15.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Mour16a,
  author = "de Moura, Leonardo",
  title = {{The Lean Theorem Prover}},
  link = "\url{https://www.youtube.com/watch?v=69ytTKfSSgc}",
  conference = "PLSE '16",
  comment = "video",
  year = "2016",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Mour19,
  author = "de Moura, Leonardo and Kong, Soonho and Avigad, Jeremy
            and van Doorn, Floris and von Raumer, Jakob",
  title = {{The Lean Theorem Prover (system description)}},
  link = "\url{http://florisvandoorn.com/papers/lean_description.pdf}",
  year = "2019",
  abstract =
    "Lean is a new open source theorem prover being developed at
    Microsoft Research and Carnegie Mellon University, with a small
    trusted kernel based on dependent type theory. It aims to bridge the
    gap between interactive and automated theorem proving, by situating
    automated tools and methods in a framework that supports user
    interaction and the construction of fully specified axiomatic
    proofs. Lean is an ongoing and long-term effort, but it already
    provides many useful components, integrated development
    environments, and a rich API which can be used to embed it into
    other systems. It is currently being used to formalize category
    theory, homotopy type theory, and abstract algebra. We describe the
    project goals, system architecture, and main features, and we
    discuss applications and continuing work.",
  paper = "Mour19.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Morr73,
  author = "Morris Jr., J.H.",
  title = {{Types are not Sets}},
  booktitle = "Symp. on the Principles of Programming Languages",
  publisher = "ACM",
  pages = "120-124",
  year = "1973"
}
```

———-

— **axiom.bib** —

```
@article{Morr84,
  author = "Morris, F.L. and Jones, C.B.",
  title = {{An Early Program Proof by Alan Turing}},
  journal = "Annals of the History of Computing",
  volume = "6",
  number = "2",
  year = "1984",
  pages = "139-143",
  abstract =
    "The paper reproduces, with typographical corrections and
    comments, a 1949 paper by Alan Turing that foreshadoes much
    subsequent work in program proving.",
  paper = "Morr84.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Mosc84,
  author = "Moschovakis, Y.N.",
  title = {{Abstract Recursion as a Foundation of the Theory of Algorithms}},
  journal = "LNCS",
  volume = "1104",
  pages = "289-364",
  publisher = "Springer",
  year = "1984",
  abstract =
    "The main object of this paper is to describe an abstract,
    (axiomatic) theory of recursion and its connection with some of
    the basic, foundational questions of computer science.",
  paper = "Mosc84.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Mose72,
  author = "Moses, Joel",
  title = {{Toward a General Theory of Special Functions}},
  journal = "Communications of the ACM",
  volume = "15",
  number = "7",
```

```
  pages = "550-554",
  year = "1972",
  abstract =
    "A list of a number of natural developments for the field of
    algebraic manipulation is given. Then the prospects for a general
    theory of functions defined by ordinary differential equations are
    discussed. The claim is made that recent developments in
    mathematics indicate that it should be possible to algorithmically
    generate many properties of solutions to differential
    equations. Such a theory is preferable to a less general effort to
    make algebraic manipulation systems knowledgeable about the usual
    special functions (e.g. exponential, hypergeometric).",
  paper = "Mose72.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mose73,
  author = "Moses, Joel and Yun, David Y.Y.",
  title = {{The EZ GCD Algorithm}},
  booktitle = "ACM Annual Conference",
  year = "1973",
  publisher = "ACM",
  abstract =
    "This paper presents a preliminary report on a new algorithm for
    computing the Greatest Common Divisor (GCD) of two multivariate
    polynomials over the integers. The algorithm is strongly
    influenced by the method used for factoring multivariate
    polynomials over the integers. It uses an extension of the Hensel
    lemma approach originally suggested by Zassenhaus for factoring
    univariate polynomials over the integers. We point out that the
    cost of the Modular GCD algorithm applied to sparse multivariate
    polynomials grows at least exponentially in the number of
    variables appearing in the GCD. This growth is largely independent
    of the number of terms in the GCD. The new algorithm, called the
    EZ (Extended Zassenhaus) GCD Algorithm, appears to have a
    computing bound which in most cases is polynomial in
    them. Especially difficult cases for the EZ GCD Algorithm are
    described. Applications of the algorithm to the computation of
    contents and square-free decomposition of polynomials is
    indicated.",
  paper = "Mose73.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mose79,
```

```
  author = "Moses, Joel and Zippel, Richard",
  title = {{Algorithms for the Integration of Algebraic Functions}},
  journal = "LNCS",
  volume = "72",
  pages = "426-430",
  year = "1979",
  paper = "Mose79.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mose79a,
  author = "Moses, Joel and Zippel, Richard",
  title = {{An Extension of Liouville's Theorem}},
  journal = "LNCS",
  volume = "72",
  pages = "426-430",
  year = "1979",
  paper = "Mose79a.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Moss80,
  author = "Mosses, Peter",
  title = {{A Constructive Approach to Compiler Correctness}},
  journal = "LNCS",
  volume = "85",
  year = "1980",
  booktitle = "Automata, Languages and Programming",
  publisher = "Springer",
  abstract =
    "It is suggested that denotational semantics definitions of
    programming languages should be based on a small number of
    abstract data types, each embodying a fundamental concept of
    computation. Once these fundamental abstract data types have been
    implemented in a particular target language (e.g. stack-machine
    code), it is a simple matter to construct a correct compiler for
    any source language from its denotational semantic definition. The
    approach is illustrated by constructing a compiler similar to the
    one which was proved correct by Thatcher, Wagner \& Wright
    (1979). Some familiarity with many-sorted algebras is presumed.",
  paper = "Moss80.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Moyx10,
  author = "Moy, Yannick and Wallenburg, Angela",
  title = {{Tokeneer: Beyond Formal Program Verification}},
  year = "2010",
  link = "\url{http://www.cse.chalmers.se/~angelaw/papers/ERTS2010.pdf}",
  abstract =
    "Tokeneer is a small-sized (10 kloc) security system which was
    formally developed and verified by Praxis at the request of NSA, using
    SPARK technology.  Since its open-source release in 2008, only two
    problems were found, one by static analysis, one by code review.  In
    this paper, we report on experiments where we systematically applied
    various static analysis tools (compiler, bug-finder, proof tools)
    and focused code reviews to all of the SPARK code (formally verified)
    and supporting Ada code (not formally verified) of the Tokeneer
    Project.  We found 20 new problems overall, half of which are defects
    that could lead to a system failure should the system be used in its
    current state.  Only two defects were found in SPARK code, which
    confirms the benefits of applying formal verification to reach
    higher levels of assurance.  In order to leverage these benefits to
    code that is was not formally verified from the start, we propose to
    associate static analyses and dynamic analyses around a common
    expression of properties and constraints. This is the goal of starting
    project Hi-Lite, which involves AdaCore and Altran Praxis together
    with several industrial users and research labs.",
  paper = "Moyx10.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Muel88,
  author = "Mueller, Robert A. and Page, L. Rex",
  title = {{Symbolic Computing with Lisp and Prolog}},
  publisher = "Wiley",
  year = "1988",
  isbn = "0-471-60771-1"
}
```

———-

— **axiom.bib** —

```
@article{Mull17,
  author = "Muller, Dennis and Gauthier, Thibault and Kaliszyk, Cezary
            and Kohlhase, Michael and Rabe, Florian",
  title = {{Classification of Alignments Between Concepts of Formal
            Mathematical Systems}},
  journal = "LNCS",
```

```
  volume = "10383",
  year = "2017",
  abstract =
    "Mathematical knowledge is publicly available in dozens of
    different formats and languages, ranging from informal
    (e.g. Wikipedia) to formal corpora (e.g. Mizar). Despite an
    enormous amount of overlap between these corpora, only few
    machine-actionalbe connections exist. We speak of alignment if the
    same concept occurs in different libraries, possibly with slightly
    different names, notations, or formal definitions. Leveraging
    these alignments creates a huge potential for knowledge sharing
    and transfer, e.g. integrating theorem provers ore reusing
    services across systems. Notably, even imperfect alignments,
    i.e. concepts that are very similar rather than identical, can
    often play very important roles. Specifically, in machine learning
    techniques for theorem proving and in automation techniques that
    use these, they allow learning-reasoning base automation for
    theorem provers to take inspiration from proofs from different
    formal proof libraries or semi-formal libraries even if the latter
    is based on a different mathematical foundation. We present a
    classification of alignments and design a simple format for
    describing alignments, as well as an infrastructure for sharing
    them. We propose these as a centralized standard for the
    community. Finally, we present an initial collection of
    approximately 12000 alignments from the different kinds of
    mathematical corpora, including proof assistant libraries and
    semi-formal corpora as a public resource.",
  paper = "Mull17.pdf"
}
```

———-

— axiom.bib —

```
@article{Mull18,
  author = "Muller, Dennis and Kohlhase, Michael and Rabe, Florian",
  title = {{Automatically Finding Theory Morphisms for Knowledge
            Management}},
  journal = "LNCS",
  volume = "11006",
  year = "2018",
  abstract =
    "We present a method for finding morphisms between formal
    theories, both within as well as across libraries based on
    different logical foundations. As they induce new theorems in the
    target theory for any of the source theory, theory morphisms are
    high-value elements of a modular formaly library. Usually, theory
    morphisms are manually encoded, but this practice requires authors
    who are familiar with source and target theories at the same time,
    which limits the scalability of the manual approach.

    To remedy this problem, we have developed a morphism finder
    algorithm that automates theory morphism discovery. In this paper we
```

```
    present an implementation in the MMT system and show specific use
    cases. We fous on an application of theory discovery, where a user
    can check whether a (part of a) formal theory already exists in
    some library, potentially avoiding duplication of work or
    suggesting an opportunity for refactoring.",
  paper = "Mull18.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Mull16,
  author = "Muller, Jean-Michel",
  title = {{Elementary Functions: Algorithms and Implementation}},
  isbn = "978-1-4899-7981-0",
  publisher = "Birkhauser",
  year = "2016",
  paper = "Mull16.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Mull13,
  author = "Mulligan, Dominic P.",
  title = {{Mosquito: An Aimplementation for Higher-Order Logic}},
  school = "University of Cambridge",
  year = "2013",
  link = "\url{https://bitbucket.org/MosquitoProofAssistant/mosquito}",
  abstract =
    "We present Mosquito: an experimental stateless, pure, largely total
    LCF-style implementation of higher-order logic using Haskell as a
    metalanguage. We discuss details of the logic implemented, kernel
    design, and novel proof state and tactic representations.",
  paper = "Mull13.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Murp18,
  author = "Murphy, Robin R.",
  title = {{Robotics Through Science Fiction}},
  publisher = "MIT Press",
  year = "2018",
  link =
    "\url{https://mitpress.mit.edu/books/robotics-through-science-fiction}",
```

```
  isbn = "978-0-262-53626-4",
  comment = "verification, validation, and trust"
}
```

———-

— **axiom.bib** —

```
@misc{Murr18,
  author = "Murray, Toby and van Oorcchot, P.C.",
  title = {{BP: Formal Proofs, the Fine Print and Side Effects}},
  link = "\url{https://people.eng.unimelb.edu.au/tobym/papers/secdev2018.pdf}",
  comment = "Version: 26 June 2018 to appear in IEEE SecDev 2018",
  abstract =
    "Given recent high-profile successes in formal verification of
    security-related properties (e.g. for seL4), and the rising
    popularity of applying formal methods to cryptographic libraries
    and security protocols like TLS, we revisit the meaning of
    security-related proofs about software. We re-examine old issues,
    and identify new questions that have escaped scrutiny in the
    formal methods literature. We consider what value proofs about
    software systems deliver to end-users (e.g. in terms of net
    assurance benefits), and at what cost in terms of side effects
    (such as changes made to software to facilitate the proofs, and
    assumption-related deployment restrictions imposed on software if
    these proofs are to remain valid in operation). We consider in
    detail, for the first time to our knowledge, possible
    relationships between proofs and side effects. To make our
    discussion concrete, we draw on tangible examples, experience, and
    the literature.",
  paper = "Murr18.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@techreport{Murt91,
  author = "Murthy, Chetan R.",
  title = {{Classical Proofs as Programs: How, What and Why}},
  type = "technical report",
  institution = "Cornell University",
  number = "TR91-1215",
  year = "1991",
  abstract =
    "We recapitulate Friedman's conservative extension result of
    (suitable) classical over constructive systems for $\prod_2^0$
    sentences, viewing it in two lights: as a translation of programs
    from an almost-functional language (with $C$) back to its
    functional core, and as a translation of a constructive logic for
    a functional language to a classical logic for an
```

```
    almost-functional language. We investigate the computational
    properties of the translation and of classical proofs and
    characterize the classical proofs which give constructions in
    concrete, computational terms, rather than logical terms. We
    characterize different versions of Friedman's translation as
    translating slightly different almost-functional languages to a
    functional language, thus giving a general method for arriving at
    a sound reduction semantics for an almost-functional language with
    a mixture of eager and lazy constructors and destructors, as well
    as integers, pairs, unions, etc. Finally, we describe how to use
    classical reasoning in a disciplined manner in giving classical
    (yet constructivizable) proofs of sentences of greater complexity
    than $\prod_2^0$. This direction offers the possibility of
    applying classical reasoning to more general programming problems.",
  paper = "Murt91.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mylo95,
  author = "Mylonakis, Nikos",
  title = "Behavioural Specifications in Type Theory",
  booktitle = "Lecture Notes in Computer Science",
  publisher = "Springer",
  volume = "1130",
  year = "1995",
  abstract =
    "In this paper we give a new view of the type theory UTT (Uniform
    theory of dependent types) [5] as a system to formally develop
    programs from algebraic specifications, comparable to
    e.g. EML([9]). We will focus our attention on behavioural
    specifications since they have not been deeply studied in a type
    theoretical setting, and we describe how to develop proofs about
    behavioural satifaction.",
  paper = "Mylo95.pdf",
  keywords = "printed"
}
```

———-

1.2.14 N

— **axiom.bib** —

```
@incollection{Nada92,
  author = "Nadathur, Gopalan and Pfenning, Frank",
  title = {{The Type System of a Higher-Order Logic Programming Language}},
  booktitle = "Types in Logic Programming",
  isbn = "9780262161312",
```

```
  publisher = "MIT Press",
  year = "1992"
}
```

———-

— **axiom.bib** —

```
@book{Nage01,
  author = "Nagel, Ernest and Newman, James R.",
  title = {{Godel's Proof}},
  publisher = "New York University Press",
  year = "2001",
  isbn = "0-8147-5816-9",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Naji84,
  author = "Najid-Zejli, H.",
  title = {{Computation in Radical Extensions}},
  booktitle = "International Symbposium on Symbolic and Algebraic
               Manipulation",
  pages = "115-122",
  publisher = "Springer",
  year = "1984",
  comment = "LNCS 174",
  paper = "Naji84.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Nath08,
  author = "Nathanson, Melvyn B.",
  title = {{Desperately Seeing Mathematical Proof}},
  journal = "Notices of the American Math. Society",
  volume = "55",
  number = "7",
  pages = "773",
  year = "2008"
}
```

———-

— **axiom.bib** —

```
@misc{Naur85,
  author = "Naur, Peter",
  title = {{Programming as Theory Building}},
  link = "\url{http://pages.cs.wisc.edu/~remzi/Naur.pdf}",
  paper = "Naur85.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Nayl00,
  author = "Naylor, Bill",
  title = {{Polynomial GCD Using Straight Line Program Representation}},
  school = "University of Bath",
  year = "2000",
  link = "\url{http://www.sci.csd.uwo.ca/~bill/thesis.ps}",
  abstract = "
    This thesis is concerned with calculating polynomial greatest common
    divisors using straight line program representation.

    In the Introduction chapter, we introduce the problem and describe
    some of the traditional representations for polynomials, we then talk
    about some of the general subjects central to the thesis, terminating
    with a synopsis of the category theory which is central to the Axiom
    computer algebra system used during this research.

    The second chapter is devoted to describing category theory. We follow
    with a chapter detailing the important sections of computer code
    written in order to investigate the straight line program subject.
    The following chapter on evalution strategies and algorithms which are
    dependant on these follows, the major algorith which is dependant on
    evaluation and which is central to our theis being that of equality
    checking. This is indeed central to many mathematical problems.
    Interpolation, that is the determination of coefficients of a
    polynomial is the subject of the next chapter. This is very important
    for many straight line program algorithms, as their non-canonical
    structure implies that it is relatively difficult to determine
    coefficients, these being the basic objects that many algorithms work
    on. We talk about three separate interpolation techniques and compare
    their advantages and disadvantages. The final two chapters describe
    some of the results we have obtained from this research and finally
    conclusions we have drawn as to the viability of the straight line
    program approach and possible extensions.

    Finally we terminate with a number of appendices discussing side
    subjects encountered during the thesis.",
  paper = "Nayl00.pdf"
}
```

———-

— axiom.bib —

```
@phdthesis{Necu98,
  author = "Necula, George Ciprian",
  title = {{Compiling with Proofs}},
  school = "Carnegie Mellon University",
  year = "1998",
  link = "\url{https://www.cs.cmu.edu/~rwh/theses/necula.pdf}",
  abstract =
    "One of the major challenges of building software systems is to
    ensure that the various components fit together in a well-defined
    manner. This problem is exacerbated by the recent advent of
    software components whose origin is unknown or inherently
    untrusted, such as mobile code or user extensions for operating
    system kernels or database servers. Such extensions are useful for
    implementing an efficient interaction model between a client and a
    server because several data exchanges between them can be saved at
    the cost of a single code exchange.

    In this dissertation, I propose to tackle such system integrity
    and security problems with techniques from mathematical logic and
    programming language semantics. I propose a framework, called
    {\sl proof-carrying code}, in which the extension provider sends
    along with the extension code a representation of a formal proof
    that the code meets certain safety and correctness
    requirements. Then, the code receiver can ensure the safety of
    executing the extension by validating the attached proof. The
    major advantages of proof-carrying code are that it requires a
    simple trusted infrastructure and that it does not impose run-time
    penalties for the purpose of ensuring safety.

    In addition to the concept of proof-carrying code, this
    dissertation contributes the idea of certifying compilation. A
    {\sl certifying compiler} emits, in addition to optimized target
    code, function specifications and loop invariants that enable a
    theorem-proving agent to prove non-trivial properties of the
    target code, such as type safety. Such a certifying compiler,
    along with a proof-generating theorem prover, is not only a
    convenient producer of proof-carrying code but also a powerful
    software-engineering tool. The certifier also acts as an effective
    referee for the correctness of each compilation, thus simplifying
    considerably compiler testing and maintenance.

    A complete system for proof-carrying code must also contain a
    {\sl proof-generating theorem prover} for the purpose of producing
    the attached proofs of safety. This dissertation shows how
    standard decision procedures can be adapted so that they can
    produce detailed proofs of the proved predicates and also how
    these proofs can be encoded compactly and checked
    efficiently. Just like for the certifying compiler, a
    proof-generating theorem prover has significant software
    engineering advantages over a traditional prover. In this case, a
    simple proof checker can ensure the soundness of each successful
```

```
    proving task and indirectly assist in testing and maintenance of
    the theorem prover.",
  paper = "Necu98.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Nede01,
  author = "Nederpelt, R. and Kamareddine, F.",
  title = {{An Abstract Syntax for a Formal Language of Mathematics}},
  booktitle = "4th int. Tbilisi Symp. on Language, Logic, and Computation",
  publisher = "unknown",
  year = "2001",
  abstract =
    "This paper provides an abstract syntax for a formal language of
    mathematics. We call our language Weak Type Theory (abbreviated
    WTT). WTT will be as faithful as possible to the mathematician's
    language yet will be formal and will not allow ambiguities. WTT
    can be used as an intermediatry between the natural language of
    the mathematician and the formal language of the logician. As far
    as we know, this is the first extensive formalization of an
    abstract syntax of a formal language of mathematics.",
  paper = "Nede01.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Negr01,
  author = "Negri, Sara and von Plato, Jan",
  title = {{Structural Proof Theory}},
  publisher = "Cambridge University Press",
  year = "2001",
  isbn = "978-0-521-06842-0",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inbook{Neub82,
  author = "Neubuser, J.",
  title = {{Computing with Groups and Their Character Tables}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
```

```
  pages = "45-56",
  abstract =
    "In this survey an attempt is made to give some impression of the
    capabilities of currently available programs for computations with
    finitely generated groups and their representations.",
  paper = "Buch82.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Newt16,
  author = "Newton, Jim E. and Verna, Didier and Colange, Maximilien",
  title = {{Programmatic Manipulation of Common Lisp Type Specifiers}},
  year = "2016",
  abstract =
    "In this article we constrast the use of the s-expression with the
    BDD (Binary Decision Diagram) as a data structure for
    programmatically manipulating Common Lisp type specifiers. The
    s-expression is the {\sl de facto} standard surface syntax and also
    programmatic representation of the type specifier, but the BDD
    data structure offers advantages: most notably, type equivalence
    checks using s-expressions can be computationally intensive,
    whereas the type equivalence check using BDDs is a check for
    object identity. As an implementation and performance experiment,
    we define the notion of maximal disjoint type decomposition, and
    discuss implementations of algorithms to compute it: a brute force
    iteration, and as a tree reduction. The experimental
    implementations represent type specifiers by both aforementioned
    data structures, and we compare the performance observed in each
    approach.",
  paper = "Newt16.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Niel19,
  author = "Nielson, Flemming and Nielson, Hanne Riis",
  title = {{Formal Methods}},
  year = "2019",
  publisher = "Springer",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Nimm02,
  author = "Nimmer, Jeremy W. and Ernst, Michael D.",
  title = {{Automatic Generation of Program Specifications}},
  booktitle = "Symp. on Software Testing and Analysis",
  publisher = "ACM Press",
  year = "2002",
  abstract =
    "Producing specifications by dynamic (runtime) analysis of program
    executions is potentially unsound, because the analyzed executions may
    not fully characterize all possible executions of the program.  In
    practice, how accurate are the results of a dynamic analysis?  This
    paper describes the results of an investigation into this question,
    determining how much specifications generalized from program runs
    must be changed in order to be verified by a static checker.
    Surprisingly, small test suites captured nearly all program behavior
    required by a specific type of static checking; the static checker
    guaranteed that the implementations satisfy the generated specifications
  , and ensured the absence of runtime exceptions. Measured
    against this verification task, the generated specifications scored
    over 90 percent on precision, a measure of soundness, and on recall, a
    measure of completeness.

    This is a positive result for testing, because it suggests that
    dynamic analyses can capture all semantic information of interest for
    certain applications. The experimental results demonstrate that a
    specific technique, dynamic invariant detection, is effective at
    generating consistent, sufficient specifications for use by a static
    checker.  Finally, the research shows that combining static and
    dynamic analyses over program specifications has benefits for users of
    each technique, guaranteeing soundness of the dynamic analysis and
    lessening the annotation burden for users of the static analysis.",
  paper = "Nimm02.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Nipk91,
  author = "Nipkow, Tobias and Snelting, Gregor",
  title = {{Type Classes and Overloading Resolution via Order-Sorted
           Unification}},
  booktitle = "Proc 5th ACM Conf. Functional Prog. Lang. and Comp. Arch.",
  year = "1991",
  publisher = "Springer",
  journal = "LNCS",
  volume = "523",
  pages = "1-14",
  abstract =
    "We present a type inference algorithm for a Haskell-like language
    based on order-sorted unification. The language features polymorphism,
    overloading, type classes and multiple inheritance. Class and instance
```

```
    declarations give rise to an order-sorted algebra of types. Type
    inference essentially reduces to the Hindley/Milner algorithm where
    unification takes place in this order-sorted algebra of types. The
    theory of order-sorted unification provides simple sufficient
    conditions which ensure the existence of principal types. The
    semantics of the language is given by a translation into ordinary
    lambda-calculus. We prove the correctness of our type inference
    algorithm with respect to this semantics.",
  paper = "Nipk91.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Nipk93,
  author = "Nipkow, Tobias and Prehofer, Christian",
  title = {{Type Checking Type Classes}},
  booktitle = "Principles of Programming Languages POPL'93",
  publisher = "ACM Press",
  year = "1993",
  pages = "409-418",
  abstract =
    "We study the type inference problem for a system with type classes as
    in the functional programming language Haskell. Type classes are an
    extension of ML-style polymorphism with overloading. We generalize
    Milner's work on polymorphism by introducing a separate context
    constraining the type variables in a typing judgement. This leads to
    simple type inference systems and algorithms which closely resemble
    those for ML. In particular we present a new unification algorithm
    which is an extension of syntactic unification with constraint
    solving. The existence of principal types follows from an analysis of
    this unification algorithm."
}
```

———-

— **axiom.bib** —

```
@article{Nipk95,
  author = "Nipkow, Tobias and Prehofer, Christian",
  title = {{Type Reconstruction for Type Classes}},
  journal = "J. of Functional Programming",
  year = "1995",
  pages = "201-224",
  abstract =
    "We study the type inference problem for a system with type classes as
    in the functional programming language Haskell. Type classes are an
    extension of ML-style polymorphism with overloading. We generalize
    Milner's work on polymorphism by introducing a separate context
    constraining the type variables in a typing judgement. This leads to
    simple type inference systems and algorithms which closely resemble
```

```
    those for ML. In particular we present a new unification algorithm
    which is an extension of syntactic unification with constraint
    solving. The existence of principal types follows from an analysis of
    this unification algorithm.",
  paper = "Nipk95.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Nipk02a,
  author = "Nipkow, Tobias and Paulson, Lawrence C. and Wenzel, Markus",
  title = {{Isabelle / HOL. A Proof Assistant for Higher-Order Logic}},
  publisher = "Springer",
  year = "2002"
}
```

———-

— **axiom.bib** —

```
@misc{Nipk18,
  author = "Nipkow, Tobias and Tabacznyj, Christophe and
            Paulson, Lawrence C. and Chaieb, Amine and Rasmussen, Thomas M.
            and Avigad, Jeremy",
  title = {{GCD in Isabelle}},
  link = "\url{http://isabelle.in.tum.ed/dist/library/HOL/HOL/GCD.html}",
  year = "2018"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Noon18,
  author = "Noonan, Matt",
  title = {{Ghosts of Departed Proofs (Functional Pearl)}},
  booktitle = "Haskell '18",
  publisher = "ACM",
  isbn = "978-1-4503-5835-4",
  year = "2018",
  abstract =
    "Library authors often are faced with a design chice: should a
    function with preconditions be implemented as a partial function,
    or by returning a failure condition on incorrect use? Neither
    option is ideal. Partial functions lead to frustrating run-time
    errors. Failure conditions must be checked at the use-site,
    placing an unfair tax on the users who have ensured that the
    function's preconditions were correctly met.
```

```
    In this paper, we introduce an API design concept called ‘ghosts
    of departed proofs’ based on the following observation:
    sophisticated preconditions can be encoded in Haskell’s type
    system with no run-time overhead, by using proofs that inhabit
    phantom type parameters attached to new type wrappers. The user
    expresses correctness arguments by constructing proofs to inhabit
    these phantom types. Critically, this technique allows the library
    {\sl user} to decide when and how to validate that the API’s
    preconditions are met.

    The ’ghost of departed proofs’ approach to API design can achieve
    many of the benefits of dependent types and refinement types, yet
    only requires some minor and well-understood extensions to Haskell
    2010. We demonstrate the utility of this approach through a series
    of case studies, showing how to enforce novel invariants for
    lists, maps, graphs, shared memory regions, and more.",
  paper = "Noon18.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Nore07,
  author = "Norell, Ulf",
  title = {{Towards a Practical Programming Language Based on
            Dependent Type Theory}},
  school = "Chalmers University",
  year = "2007",
  link = "\url{http://www.cse.chalmers.se/~ulfn/papers/thesis.pdf}",
  abstract =
    "Dependent type theories have a long history of being used for
    theorem proving. One aspect of type theory which makes it very
    powerful as a proof language is that it mixes deduction with
    computation. This also makes type theory a good candidate for
    programming -- the strength of the type system allows properties
    of programs to be stated and established, and the computational
    properties provide semantics for the programs.

    This thesis is concerned with bridging the gap between the
    theoretical presentations of type theory and the requirements of
    practical programming languages. Although there are many
    challenging research problems left to solve before we have an
    industrial scale programming language based on type theory, this
    thesis takes us a good step along the way.

    In functional programming languages pattern matching provides a
    concise notation for defining functions. In dependent type theory,
    pattern matching becomes even more powerful, in that inspecting
    the value of a particular term can reveal information about the
    types and values of other terms. In this thesis we give a type
    checking algorithm for definitions by pattern matching in type
```

```
    theory, supporting overlapping patterns, and pattern matching on
    intermediate results using the {\sl with} rule.

    Traditional presentations of type theory suffer from rather
    verbose notation, cluttering programs and proofs with, for
    instance, explicit type information. One solution to this problem
    is to allow terms that can be inferred automatically to be
    omitted. This is usually implemented by inserting metavariables in
    place of the omitted terms and using unification to solve these
    metavariables during type checking. We present a type checking
    algorithm for a theory with metavariables and prove its soundness
    independent of whether the metavariables are solved or not.

    In any programming language it is important to be able to
    structure large programs into separate units or modules and limit
    the interaction between these modules. In this thesis we present a
    simple, but powerful module system for a dependently typed
    language. The main focus of the module system is to manage the
    name space of a program, and an important characteristic is a
    clear separation between the module system and the type checker,
    making it largely independent of the underlying language.

    As a side track, not directly related to the use of type theory
    for programming, we present a connnection between type theory and
    a first-order logic theorem prover. This connection saves the user
    the burden of proving simple, but tedious first-order theorems by
    leaving them for the prover. We use a transparent translation to
    first-order logic which makes the proofs constructed by the
    theorem prover human readable. The soundness of the connection is
    established by a general metatheorem.

    Finally we put our work into practice in the implementation of a
    programming language, Agda, based on type theory. As an
    illustrating example, we show how to program a simple certified
    prover for equations in a commutative monoid, which can be used
    internally in Agda. Much more impressive examples have been done
    by others, showing that the ideas developed in this thesis are
    viable in practice.",
  paper = "Nore07.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Norr02,
  author = "Norrish, Michael and Slind, Konrad",
  title = {{A Thread of HOL Development}},
  journal = "Computer Journal",
  volume = "45",
  number = "1",
  pages = "37-45",
  year = "2002",
```

```
  abstract =
    "The HOL system is a mechanized proof assistant for higher order
    logic that has been under continuous development since the
    mid-1980s, by an ever-changing group of developers and external
    contributors. we give a brief overview of various implementations
    of the HOL logic before focusing on the evolution of certain
    important features available in a recent implementation. We also
    illustrate how the module system of Standard ML provided security
    and modularity in the construction of the HOL kernel, as well as
    serving in a separate capacity as a useful representation medium
    for persistent, hierarchical logical theories.",
  paper = "Norr02.pdf",
  keywords = "printed, DONE"
}
```

———-

1.2.15 O

— **axiom.bib** —

```
@misc{Ober18,
  author = "Oberhoff, Sebastian",
  title = {{Incompleteness Ex Machina}},
  year = "2018",
  abstract =
    "In this essay we'll prove Godel's incompleteness theorems
    twice. First, we'll prove them the good old-fashioed way. Then
    we'll repeat the feat in the setting of computation. In the
    process we'll discover that Godel's work, rightly viewed, needs to
    be split into two parts: the transport of computation into the
    arena of arithmetic on the one hand and the actual incompleteness
    theorems on the other. After we're done there will be cake.",
  paper = "Ober18.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Ocon05,
  author = "O'Connor, Russell",
  title = {{Essential Incompleteness of Arithmetic Verified by Coq}},
  booktitle = "Theorem Proving in Higher Order Logics",
  publisher = "Springer",
  pages = "245-260",
  year = "2005"
}
```

———-

— **axiom.bib** —

```
@misc{Oder01,
  author = "Odersky, Martin and Zenger, Christoph and
            Zenger, Matthias",
  title = {{Colored Local Type Inference}},
  year = "2001",
  abstract =
    "We present a type system for a language based on $F_{\le}$, which
    allows certain type annotations to be elided in actual
    programs. Local type inference determines types by a combination
    of type propagation and local constraint solving, rather than by
    global constraint solving. We refine the previously existing local
    type inference system of Pierce and Turner by allowing partial
    type information to be propagated. This is expressed by coloring
    types to indicate propagation directions. Propagating partial type
    information allows us to omit type annotations for the visitor
    pattern, the analogue of pattern matching in languages without sum
    types.",
  paper = "Oder01.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Odif92,
  author = "Odifreddi, Piergiorgio",
  title = {{Classical Recursion Theory: The Theory of Functions and Sets of
           Natural Numbers}},
  publisher = "Elsevier",
  year = "1992"
}
```

———-

— **axiom.bib** —

```
@book{Oest53,
  author = "Oesterle, John A.",
  title = {{Logic: The Art of Defining and Reasoning}},
  year = "1953",
  publisher = "Prentice-Hall",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Oisd18,
  author = "Anonymous",
  title = {{Solving Rings in Agda}},
  booktitle = "Proc. ACM Program. Lang.",
  publisher = "ACM",
  year = "2018",
  abstract =
    "We present a new library which automates the construction of
    equivalence proofs between polynomials over commutative rings and
    semirings in the programming language Agda [Norell and Chapman
    2008]. It is significantly faster than Agda's existing solver. We
    use reflection to provide a simple interface to the solver, and
    demonstrate how to use the constructed proofs to provide
    step-by-step solutions.",
  paper = "Oisd18.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@misc{Olsa19,
  author = "Olsak, Miroslav and Kaliszyk, Cezary and Urban, Josef",
  title = {{Property Invariant Embedding for Automated Reasoning}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1911.12073.pdf}",
  abstract =
    "Automated reasoning and theorem proving have recently become
    major challenges for machine learning. In other domains,
    representations that are able to abstract over unimportant
    transformations, such as abstraction over translations and
    rotations in vision, are becoming more common. Standard methods of
    embedding mathematical formulas for learning theorem proving are
    however yet unable to handle many important transformations. In
    particular, embedding previously unseen labels, that often arise
    in definitional encodings and in Skolemizatin, has been very weak
    so far. Similar problems appear when tranferring knowledge between
    known symbols.

    We propose a novel encoding of formulas that extends existing
    graph neural network models. This encoding represents symbols only
    by nodes in the graph, without giving the network any knowledge of
    the original labels. We provide additional links between such
    nodes that allow the network to recover the meaning and therefore
    correctly embed such nodes irrespective of the given labels. We
    test the proposed encoding in an automated theorem prover based on
    the tableaux connection calculus, and show that it improves on the
    best characterizations used so far. The encoding is further
    evaluated on the premise selection task and a newly introduced
    symbol guessing task, and shown to correctly predict 65\% of the
    symbol names.",
  paper = "Olsa19.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Olss04,
  author = "Olsson, Ola and Wallenburg, Angela",
  title = {{Automatic Generation of Customised Induction Rules for
            Proving Correctness of Imperative Programs}},
  year = "2004",
  comment = "paper follows thesis in file",
  link = "\url{http://www.cse.chalmers.se/~angelaw/papers/lic.pdf}",
  abstract =
    "In this paper we develop a method for automatic construction of
    customised induction rules for use in a semi-interactiv e theorem
    prover.  The induction rules are developed to prove the total
    correctness of loops in an object-oriented language.  We concentrate
    on integers.  First we compute a partition of the domain of the
    induction variable.  Our method makes use of failed proof attempts in
    the theorem prover to gain information about the problem structure and
    create the partition.  Then, based on this partition we create an
    induction rule, in destructor style, that is customised to make the
    proving of the loop simpler.  Our concern is in user interaction,
    rather than in proof strength.

    Using the customised induction rules, some separation of proof of
    control flow and data correctness is achieved, and we find that in
    comparison to standard (Peano) induction or Noetherian induction,
    simpler user interaction can be expected.  Furthermore, by using
    destructor style induction we circumvent the problem of creating
    inverses of functions.  We show the soundness of the customised
    induction rules created by the method.  Furthermore, we use the
    machinery of the theorem prover (KeY) to make the method automatic.
    Several interesting areas are also identified that could open up for
    a larger range of loops the method can handle, as well as pointing
    towards full automation of these cases.",
  paper = "Olss04.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Olss05,
  author = "Olsson, Ola and Wallenburg, Angela",
  title = {{Customised Induction Rules for Proving Correctness of
            Imperative Programs}},
  year = "2005",
  link = "\url{http://www.cse.chalmers.se/~angelaw/papers/olswbg05.pdf}",
  abstract =
```

```
    "In this paper we develop a method for automatic construction of
    customised induction rules for use in a semi-interactiv e theorem
    prover.  The induction rules are developed to prove the total
    correctness of loops in an object-oriented language.  We concentrate
    on integers.  First we compute a partition of the domain of the
    induction variable.  Our method makes use of failed proof attempts in
    the theorem prover to gain information about the problem structure and
    create the partition.  Then, based on this partition we create an
    induction rule, in destructor style, that is customised to make the
    proving of the loop simpler.  Our concern is in user interaction,
    rather than in proof strength.

    Using the customised induction rules, some separation of proof of
    control flow and data correctness is achieved, and we find that in
    comparison to standard (Peano) induction or Noetherian induction,
    simpler user interaction can be expected.  Furthermore, by using
    destructor style induction we circumvent the problem of creating
    inverses of functions.  We show the soundness of the customised
    induction rules created by the method.  Furthermore, we use the
    machinery of the theorem prover (KeY) to make the method automatic.
    Several interesting areas are also identified that could open up for
    a larger range of loops the method can handle, as well as pointing
    towards full automation of these cases.",
  paper = "Olss05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Oste93,
  author = "Ostebee, Arnold and Zorn, Paul",
  title = {{Telegraphic Reviews}},
  journal = "The American Mathematical Monthly",
  volume = "100",
  number = "8",
  pages = "812-817",
  year = "1993",
  paper = "Oste93.pdf",
  keywords = "axiomref"
}
```

———-

1.2.16 P

— **axiom.bib** —

```
@inproceedings{Padg85,
  author = "Padget, J.A.",
```

```
  title = {{Current Development in LISP}},
  booktitle = "EUROCAL 85",
  publisher = "Springer",
  year = "1985",
  abstract =
    "This paper is divided into three sections. The first gives an
    overview of the presently available and emerging dialects of LISP
    and how design decisions within them affect symbolic algebra. The
    second discusses recent developments, in particular, Common LISP
    subsetting, portability, pure language research and mixed paradigm
    systems. The third part is devoted to what is happening in
    specialised LISP hardware in Japan, in the United States and in
    Europe. The subject matter of each of these three sections is so
    tightly interwoven however that the detailed discussion of some
    material may be postponed until a later section although some
    readers it might seem appropriate for inclusion earlier. It shoud
    also be mentioned that this is a survey, therefore the items have
    been selected for mention on the grounds of interest rather than
    completeness.",
  paper = "Padg85.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pado06,
  author = "Padovani, Luca and Zacchiroli, Stefano",
  title = {{From Notation to Semantics: There and Back Again}},
  booktitle = "5th Conf. on Mathematical Knowledge Management",
  year = "2006",
  abstract =
    "Mathematical notation is a structured, open, and ambiguous
    language. In order to support mathematical notation in MKM
    applications one must necessarily take into account presentational as
    well as semantic aspects. The former are required to create a
    familiar, comfortable, and usable interface to interact with. The
    latter are necessary in order to process the information
    meaningfully. In this paper we investigate a framework for dealing
    with mathematical notation in a meaningful, extensible way, and we
    show an effective instantiation of its architecture to the field of
    interactive theorem proving. The framework builds upon well-known
    concepts and widely-used technologies and it can be easily adopted by
    other MKM applications.",
  paper = "Pado06.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Page18,
  author = "Page, William",
  title = {{MathAction Front Page}},
  year = "2018",
  link = "\url{http://axiom-wiki.newsynthesis.org/FrontPage}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Page18a,
  author = "Page, William",
  title = {{MathAction Facebook Page}},
  year = "2018",
  link =
  "\url{https://www.facebook.com/httpaxiom-wikinewsynthesisorg-229826785723}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Para19,
  author = "Paraskevopoulou, Zoe and Appel, Andrew W.",
  title = {{Closure Conversion is Safe for Space}},
  booktitle = "Inter. Conf. on Functional Programming",
  publisher = "ACM",
  year = "2019",
  abstract =
    "We formally prove that closure conversion with flat environments
    for CPS lambda calculus is correct (preserves semantics) and safe
    for time and space, meaning that produced code preserves the time
    and space required for execution of the source program.

    We give a cost model to pre- and post-closure-conversion code by
    formalizing profiling semantics that keep track of the time and
    space resources needed for the execution of a program, taking
    garbage collection into account. To show preservation of time and
    space we set up a genera 'garbage-collection compatible' binary
    logical relation that establishes invariants on resource
    consumption of the related programs, along with functional
    correctness. Using this framework, we show semantics preservation
    and space and time safety for terminating source programs, and
    divergence preservation and space safety for diverging source
    programs.

    This is the first formal proof of space-safety of a
    closure-conversion transformation. The transformation and the
    proof are parts of the CertiCoq compiler pipeline from Coq
```

```
    (Gallina) through CompCert Clight to assembly language. Our
    results are mechanized in the Coq proof assistant.",
  paper = "Para19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Pari92,
  author = "Parigot, Michel",
  title = {{$\lambda\mu$-Calculus: An Algorithmic Interpretation of
            Classical Natural Deduction}},
  journal = "LNCS",
  volume = "624",
  pages = "190-201",
  year = "1992",
  paper = "Pari92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Pari92a,
  author = "Parigot, Michel",
  title = {{Recursive Programming with Proofs}},
  journal = "Theoretical Computer Science",
  volume = "94",
  pages = "335-356",
  year = "1992",
  paper = "Pari92a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Pari19,
  author = "Parisse, Bernard",
  title = {{Computing Huge Groebner Basis like Cyclic10 over
            $\mathbb{Q}$ with Giac}},
  link = "\url{https://hal.archives-ouvertes.fr/hal-02081648}",
  year = "2019",
  abstract =
    "We present a short description on how to fine-tune the
    modular algorithm implemented in the Giac computer algebra system
    to reconstruct large Groebner basis over $\mathbb{Q}$. The
    classical cyclic10 benchmark will serve as example.",
  paper = "Pari19.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Pate78,
  author = "Paterson, M. S.",
  title = {{Linear Unification}},
  journal = "J. Computer and System Sciences",
  volume = "16",
  number = "2",
  year = "1978",
  pages = "158-167",
  abstract =
    "A unification algorithm is described which tests a set of expressions
    for unifiability and which requires time and space which are only linear
    in the size of the input",
  paper = "Pate78.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Patt19,
  author = "Patterson, Daniel and Ahmed, Amal",
  title = {{The Next 700 Compiler Correctness Theorems}},
  booktitle = "Inter. Conf. on Functional Programming",
  link = "\url{https://www.youtube.com/watch?v=qrwzpo6ISCQ}",
  publisher = "ACM",
  year = "2019",
  abstract =
    "Compiler correctness is an old problem, with results stretching
    back beyond the last half-century. Founding the field, John
    McCarthy and James Painter set out to build ' completely
    trustworthy compiler'. And yet, until quite recently, even despite
    truly impressive verification efforts, the theorems being proved
    were only about the compilation of whole programs, a theoretically
    quite appealing but practically unrealistic simplification. For a
    compiler correctness theorem to assure complete trust, the theorem
    must reflect the reality of how the compiler will be used.

    There has been much recent work on more realistic 'compositional'
    compiler correctness aimed at proving correct compilation of
    components while supporting linking with components compiled from
    different languages using different compilers. However, the
    variety of theorems, stated in remarkably different ways, raises
    questions about what researchers even mean by a 'compiler is
    correct'. In this pearl, we develop a new framework with which to
    understand compiler correctness theorems in the presence of
    linking, and apply it to understanding and comparing this
```

```
    diversity of results. In doing so, not only are we better able to
    assess their relative strengths and weaknesses, but gain insight
    into what we as a community should expect from compiler
    correctness theorems of the future.",
  paper = "Patt19.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Paul13,
  author = "Paule, Peter",
  title = {{Mathematics, Computer Science and Logic -- A Never Ending
            Story}},
  publisher = "Springer",
  year = "2013",
  paper = "Paul13.pdf"
}
```

———-

— **axiom.bib** —

```
@inbook{Paul90b,
  author = "Paulson, Lawrence C.",
  title = {{Designing a Theorem Prover}},
  booktitle = "Handbook of Logic in Computer Science, Volume 2",
  publisher = "Oxford University Press",
  pages = "415-475",
  year = "1992",
  paper = "Paul90b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Pear15,
  author = "Pearce, David J. and Groves, Lindsay",
  title = {{Designing a Verifying Compiler: Lessons Learned from
            developing Whiley}},
  year = "2015",
  link = "\url{https://homepages.ecs.vuw.ac.nz/~djp/files/SCP15-preprint.pdf}",
  abstract =
    "An ongoing challenge for computer science is the development of a
    tool which automatically verifies programms meet their
    specifications, and are free from runtime errors such as
    divide-by-zero, array out-of-bounds and null dereferences. Several
    impressive systems have been developed to this end, such as
```

```
    ESC/Java and Spec\#, which build on existing programming languages
    (e.g., Java, C\#). We have been developing a programming language
    from scratch to simplify verification, called Whiley, and an
    accompanying verifying compiler. In this paper, we present a
    technical overview of the verifying compiler and document the
    numerous design decisions made. Indeed, many of our designs
    reflect thos of similar tools. However, they have often been
    ignored in the literature and/or spread thinly throughout. In
    doing this, we hope to provide a useful resource for those
    building verifying compilers.",
  paper = "Pear15.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Perr19,
  author = "Perrone, Paolo",
  title = {{Notes on Category Theory}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1912.10642.pdf}",
  paper = "Perr19.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Petz08,
  author = "Petzold, Charles",
  title = {{The Annotated Turing}},
  publisher = "Wiley",
  year = "2008",
  isbn = "978-0-470-22905-7",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pfen91,
  author = "Pfenning, Frank",
  title = {{Logic Programming in the LF Logical Framework}},
  booktitle = "Proc. First Workshop on Logical Frameworks",
  year = "1991",
  paper = "Pfen91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pfen91a,
  author = "Pfenning, Frank",
  title =
    {{Unification and Anti-Unification in the Calculus of Constructions}},
  booktitle = "Logic in Computer Science 91",
  year = "1991",
  pages = "74-85",
  abstract =
    "We present algorithms for unification and anti-unification in the
    Calculus of Constructions, where occurrences of free variables (the
    variables subject to instantiation) are restricted to higher-order
    patterns, a notion investigated for the simply-typed $\lambda$-calculus
    by Miller. Most general unifiers and least common anti-instances are
    shown to exist and are unique up to a simple equivalence.  The
    unification algorithm is used for logic program execution and type and
    term reconstruction in the current implementation of Elf and has
    shown itself to be practical.  The main application of the
    anti-unification algorithm we have in mind is that of proof
    generalization.",
  paper = "Pfen91a.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Pfen92,
  author = "Pfenning, Frank",
  title = {{Types in Logic Programming}},
  isbn = "9780262161312",
  publisher = "MIT Press",
  year = "1992",
  abstract =
    "Types play an increasingly important role in logic programming, in
    language design as well as language implementation.  We present
    various views of types, their connection, and their role within the
    logic programming paradigm.

    Among the basic views of types we find
    the so-called descriptive systems, where types describe properties of
    untyped logic programs, and prescriptive systems, where types are
    essential to the meaning of programs.  A typical application of
    descriptive types is the approximation of the meaning of a logic
    program as a subset of the Herbrand universe on which a predicate
    might be true.  The value of prescriptive systems lies primarily in
    program development, for example, through early detection of errors
    in programs which manifest themselves as type inconsistencies, or as
```

```
    added documentation for the intended and legal use of predicates.

    Central topics within these views are the problems of type inference
    and type reconstruction, respectively.  Type inference is a form of
    analysis of untyped logic programs, while type reconstruction attempts
    to fill in some omitted type information in typed logic programs and
    generalizes the problem of type checking.  Even though analogous
    problems arise in functional programming, algorithms addressing these
    problems are quite different in our setting.

    Among the specific forms of types we discuss are simple types,
    recursive types, polymorphic types, and dependent types.  We also
    briefly touch upon subtypes and inheritance, and the role of types
    in module systems for logic programming languages.",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen94,
  author = "Pfenning, Frank",
  title = {{Elf: A Meta-Language for Deductive Systems}},
  year = "1994",
  paper = "Pfen94.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen99,
  author = "Pfenning, Frank and Schurmann, Carston",
  title = {{System Description: Twelf -- A Meta-Logical Framework for
            Deductive Systems}},
  year = "1999",
  link = "\url{https://www.cs.cmu.edu/~fp/papers/cade99.pdf}",
  abstract =
    "Twelf is a meta-logical framework for the specification,
    implementation, and meta-theory of deductive systems from the
    theory of programming languages and logics. It relies on the LF
    type theory and the jugements-as-types methodology for
    specifications [HHP93], a constraint logic programming interpreter
    for implementation [Pfe91], and the meta-logic $M_2$ for reasoning
    about object languages encoded in LF [SP98]. It is a significant
    extension and complete reimplementation of the Elf system [Pfe94].

    Twelf is written in Standard ML and runs under SML of New Jersey
    and MLWorks on Unix and Window platforms. The current version
    (1.2) is distributed with a complete manual, example suites, a
```

```
    tutorial in the form of on-line lecture notes [Pfe], and an Emacs
    interface. Source and binary distributions are accessible via the
    Twelf home page \url{http://www.cs.cmu.edu/~twelf}.",
  paper = "Pfen99.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen04,
  author = "Pfenning, Frank",
  title = {{Automated Theorem Proving}},
  year = "2004",
  link = "\url{https://www.cs.cmu.edu/~fp/courses/atp/handouts/atp.pdf}",
  comment = "Draft of Spring 2004",
  paper = "Pfen04.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen06,
  author = "Pfenning, Frank",
  title = {{Constraint Logic Programming}},
  year = "2006",
  link = "\url{http://www.cs.cmu.edu/~fp/courses/lp/lectures/27-clp.pdf}",
  paper = "Pfen06.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen07,
  author = "Pfenning, Frank",
  title = {{Logic Programming}},
  year = "2007",
  link = "\url{http://www.cs.cmu.edu/~fp/courses/lp/lectures/lp-all.pdf}",
  paper = "Pfen07.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen12,
```

```
  author = "Pfenning, Frank",
  title = {{Proof Theory Foundations}},
  year = "2012",
  link = "\url{https://www.cs.uoregon.edu/research/summerschool/summer12/videos/Pfenning1_0.mp4}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen16,
  author = "Pfenning, Frank",
  title = {{Lecture Notes on Focusing}},
  year = "2016",
  link = "\url{http://www.cs.cmu.edu/~fp/courses/15816-f16/lectures/18-focusing.pdf}",
  paper = "Pfen16.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen17,
  author = "Pfenning, Frank",
  title = {{Constructive Logic}},
  year = "2017",
  link = "\url{http://www.cs.cmu.edu/~fp/courses/15317-f17/schedule.html}",
  paper = "Pfen17.tgz"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pick20,
  author = "Pickard, Mitchell and Hutton, Graham",
  title = {{Dependently-Typed Compilers Don't Go Wrong}},
  booktitle = "Programming Languages",
  publisher = "ACM",
  year = "2020",
  link = "\url{http://www.cs.nott.ac.uk/~pszgmh/well-typed.pdf}",
  abstract =
    "Compilers are difficult to write, and difficult to get
    right. Bahr and Hutton recently developed a new technique for
    calculating compilers directly from specifications of their
    correctness, which ensures that the resulting compilers are
    correct-by-construction. To date, however, this technique has only
    been applicable to source languages that are untyped. In this
    article, we show that moving to a dependently-typed setting allows
```

```
    us to naturally support typed source languages, ensure that all
    compilation components are type-safe, and make the resulting
    calculations easier to mechanically check using a proof
    assistant.",
  paper = "Pick20.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Pier91,
  author = "Pierce, Benjamin C.",
  title = {{Programming with Intersection Types and Bounded Polymorphism}},
  school = "Carnegie Mellon University",
  year = "1991",
  comment = "CMU-CS-91-205",
  abstract =
    "Intersection types and bounded quantification are complementary
    mechanisms for extending the expressive power of statically typed
    programming languages. They begin with a common framework: a simple,
    typed language with higher-order functions and a notion of subtyping.
    Intersection types extend this framework by giving every pair of types
    $\sigma$ and $\tau$ a greatest lower bound, $\sigma \land \tau$,
    corresponding intuitively to the intersection of the sets of values
    described by $\sigma$ and $\tau$. Bounded quantification extends the
    basic framework along a different axis by adding polymorphic functions
    that operate uniformly on all the subtypes of a given type. This thesis
    unifies and extends prior work on intersection types and bounded
    quantification, previously studied only in isolation, by investigating
    theoretical and practical aspects of a typed $\lambda$-calculus
    incorporating both.

    The practical utility of this calculus, called $F_\land$ is
    established by examples showing, for instance, that it allows a rich
    form of ``coherent overloading'' and supports an analog of abstract
    interpretation during typechecking; for example, the addition function
    is given a type showing that it maps pairs of positive inputs to a
    positive result, pairs of zero inputs to a zero result, etc. More
    familiar programming examples are presented in terms of an extention
    of Forsythe (an Algol-like language with intersection types),
    demonstrating how parametric polymorphism can be used to simplify and
    generalize Forsythe's design. We discuss the novel programming and
    debugging styles that arise in $F_\land$.

    We prove the correctness of a simple semi-decision procedure for the
    subtype relation and the partial correctness of an algorithm for
    synthesizing minimal types of $F_\land$ terms. Our main tool in this
    analysis is a notion of ``canonical types,'' which allows proofs to be
    factored so that intersections are handled separately from the other
    type constructors.
```

```
    A pair of negative results illustrates some subtle complexities of
    $F_\land$. First, the subtype relation of $F_\land$ is shown to be
    undecidable; in fact, even the sutype relation of pure second-order
    bounded quantification is undecidable, a surprising result in its own
    right. Second, the failure of an important technical property of the
    subtype relation -- the existence of least upper bounds -- indicates
    that typed semantic models of $F_\land$ will be more difficult to
    construct and analyze than the known typed models of intersection
    types. We propose, for future study, some simpler fragments of
    $F_\land$ that share most of its essential features, while recovering
    decidability and least upper bounds.

    We study the semantics of $F_\land$ from several points of view. An
    untyped model based on partial equivalence relations demonstrates the
    consistency of the typing rules and provides a simple interpolation
    for programs, where ``$\sigma$ is a subtype of $\tau$'' is read as
    ``$\sigma$ is a subset of $\tau$.'' More refined models can be
    obtained using a translation from $F_\land$ into the pure polymorphic
    $\lambda$-calculus; in these models, ``$\sigma$ is a subtype of
    $\tau$'' is interpreted by an explicit coercion function from $\sigma$
    to $\tau$.  The nonexistence of least upper bounds shows up here in
    the failure of known techniques for proving the coherence of the
    translation semantics.  Finally, an equational theory of equivalences
    between $F_\land$ terms is presented and its soundness for both styles
    of model is verified.",
  paper = "Pier91.pdf"
}
```

———-

— axiom.bib —

```
@techreport{Pier91a,
  author = "Pierce, Benjamin C.",
  title = {{Bounded Quantification is Undecidable}},
  year = "1991",
  number = "CMU-CS-91-161",
  institution = "Carnegie Mellon University",
  link = "\url{http://repository.cmu.edu/cgi/viewcontent.cgi?article=3059}",
  abstract =
    "$F_\le$ is a typed $\lambda$-calculus with subtyping and bounded
    second-order polymorphism. First introduced by Cardelli and Wegner, it
    has been widely studied as a core calculus for type systems with
    subtyping.

    Curien and Ghelli proved the partial correctness of a recursive
    procedure for computing minimal types of $F_\le$ terms and showed
    that the termination of this procedure is equivalent to the
    termination of its major component, a procedure for checking the
    subtype relation between $F_\le$ types. Ghelli later claimed that
    this procedure is also guaranteed to terminate, but the discovery of a
    subtle bug in his proof led him recently to observe that, in fact,
    there are inputs on which the subtyping procedure diverges. This
```

```
    reopens the question of the decidability of subtyping and hence of
    typechecking.

    This question is settled here in the negative, using a reduction from
    the halting problem for two-counter Turing machines to show that the
    subtype relation of $F_\le$ is undecidable.",
  paper = "Pier91a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pier93,
  author = "Pierce, Benjamin C. and Turner, David N.",
  title = {{Object-oriented Programming without Recursive Types}},
  booktitle = "POPL'93",
  pages = "299-312",
  year = "1993",
  abstract =
    "It is widely agreed that recursive types are inherent in the static
    typing of the essential mechanisms of object-oriented programming:
    encapsulation, message passing, subtyping, and inheritance. We
    demonstrate here that modeling object encapsulation in terms of
    existential types yields a substantially more straightforward
    explanation of these features in a simpler calculus without recursive
    types.",
  paper = "Pier93.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Pier97,
  author = "Pierce, Benjamin C. and Turner, David N.",
  title = {{Local Type Inference}},
  institution = "Indiana University",
  year = "1997",
  type = "CSCI Technical Report",
  number = "493",
  abstract =
    "We study two partial type inference methods for a language combining
    subtyping and impredicative polymorphism. Both methods are {\sl local}
    in the sense that missing annotations are recovered using only
    information from adjacent nodes in the syntax tree, without long
    distance constraints such as unification variables. One method infers
    type arguments in polymorphic applications using a local constraint
    solver. The other infers annotations on bound variables in function
    abstractions by propagating type constraints downward from enclosing
    application nodes. We motivate our design choices by a statistical
    analysis of the uses of type inference in a sizable body of existing
```

```
    ML code.",
  paper = "Pier97.pdf, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Pier08,
  author = "Pierce, Benjamin C.",
  title = {{Types Considered Harmful}},
  comment = "Talk at Mathematical Foundations of Programming Languages",
  year = "2008",
  link = "\url{https://www.cis.upenn.edu/~bcpierce/papers/harmful-mfps.pdf}",
  paper = "Pier08.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Pier16,
  author = "Pierce, Benjamin and Appel, Andrew W. and Weirich, Stephanie
            and Zdancewic, Steve and Shao, Zhong and Chlipala, Adam",
  title = {{The Science of Deep Specification}},
  year = "2016",
  link = "\url{https://www.cis.upenn.edu/~bcpierce/papers/deepspec-hcss2016-slides.pdf}",
  paper = "Pier16.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Pier18,
  author = "Pierce, Benjamin",
  title = {{The Science of Deep Specification}},
  year = "2018",
  link = "\url{https://www.cis.upenn.edu/~bcpierce/papers/pierce-etaps2018.pdf}",
  paper = "Pier18.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Pier19,
  author = "Pierce, Benjamin and de Amorim, Arthur Azevedo and
            Casinghino, Chris and Gaboardi, Marco and Greenberg, Michael
```

```
            and Hritcu, Catalin and Sjoberg, Vilhelm and Yorgey, Brent",
  title = {{Logical Foundations}},
  year = "2019",
  link = "\url{https://softwarefoundations.cis.upenn.edu/lf-current/lf.tgz}",
  paper = "Pier19.tgz"
}
```

———-

— **axiom.bib** —

```
@misc{Piot19,
  author = "Piotrowski, Bartosz and Brown, Chad E. and
            Kaliszyk, Cezary",
  title = {{Can Neural Networks Learn Symbolic Rewriting?}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1911.04783.pdf}",
  abstract =
    "This work investigates if the current neural architectures are
    adequate for learning symbolic rewriting. Two kinds of data sets
    are proposed for this research -- one based on automated proofs
    and the other being a synthetic set of polynomial terms. The
    experiments with use of the current neural machine translation
    models are performed and its results are discussed. Ideas for
    extending this line of research are proposed and its relevance is
    motivated.",
  paper = "Piot19.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Piroxx,
  author = "Pirog, Maciej and Gibbons, Jeremy",
  title = {{Extended Abstract: A Functional Derivation of the Warren
            Abstract Machine}},
  link = "\url{http://www.cs.ox.ac.uk/jeremy.gibbons/publications/wam.pdf}",
  year = "unknown",
  abstract =
    "Based on Danvy et al.'s functional correspondence, we give a
    further example of gradual refinement of an interpreter into a
    known, low-level abstract machine underlying real-world compilers,
    by deriving an abstract model of the Warren Abstract Machine from
    a simple resolution-based Prolog interpreter. We show that other
    well-known functional programming techniques (namely, explicit
    laziness and semi-persistent data structures) can help to develop
    abstract machines without detailed examination of the semantics
    realised by the interpreter.",
  paper = "Piroxx.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pisk15,
  author = "Piskac, Ruzica",
  title = {{From Decision Procedures to Synthesis Procedures}},
  booktitle = "Symp. on Symbolic and Numeric Algorithms for
               Scientific Computing",
  publisher = "ACM",
  year = "2015",
  abstract =
    "Software synthesis is a technique for automatically generating
    code from a given specification. The goal of software synthesis is
    to make software development easier while increasing both the
    productivity of the programmer and the correctness of the produced
    code. In this paper we present an approach to synthesis that
    relies on the use of automated reasoning and decision
    procedures. First we describe how to generalize decision
    procedures into predictable and complete synthesis
    procedures. Here completeness means that the procedure is
    guaranteed to find code that satisfies the given specification. We
    illustrate the process of turning a decision procedure into a
    synthesis procedure using linear integer arithmetic as an example.

    However, writing a complete specification can be a tedious task,
    sometimes even harder than writing the code itself. To overcome
    this problem, ideally the user could provide a few input-output
    examples, and then the code should be automatically derived. We
    outline how to broaden usability and applications of current
    software synthesis techniques. We conclude with an outlook on
    possible future research directions and applications of synthesis
    procedures.",
  paper = "Pisk15.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Pitt18,
  author = "Pittman, Dan",
  title = {{Proof Theory Impressionism: Blurring the Curry-Howard Line}},
  link = "\url{https://www.youtube.com/watch?v=jrVPB-Ad5Gc}",
  year = "2018",
  comment = "Strange Loop 2018 Conference",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@book{Plat18,
  author = "Platzer, Andre",
  title = {{Logical Foundations of Cyber-Physical Systems}},
  publisher = "Springer",
  year = "2018",
  isbn = "978-3-319-63587-3",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Poli18,
  author = "Polikarpova, Nadia",
  title = {{Type-Driven Program Synthesis}},
  link = "\url{https://www.youtube.com/watch?v=HnOix9TFy1A}",
  year = "2018",
  abstract =
    "A promising approach to improving software quality is to enhance
    programming languages with declarative constructs, such as logical
    specifications or examples of desired behavior, and to use program
    synthesis technology to translate these constructs into efficiently
    executable code. In order to produce code that provably satisfies a
    rich specification, the synthesizer needs an advanced program
    verification mechanism that is sufficiently expressive and fully
    automatic. In this talk, I will present a program synthesis technique
    based on refinement type checking: a verification mechanism that
    supports automatic reasoning about expressive properties through a
    combination of types and SMT solving.

    The talk will present two applications of type-driven synthesis. The
    first one is a tool called Synquid, which creates recursive functional
    programs from scratch given a refinement type as input. Synquid is the
    first synthesizer powerful enough to automatically discover programs
    that manipulate complex data structures, such as balanced trees and
    propositional formulas. The second application is a language called
    Lifty, which uses type-driven synthesis to repair information flow
    leaks. In Lifty, the programmer specifies expressive information flow
    policies by annotating the sources of sensitive data with refinement
    types, and the compiler automatically inserts access checks necessary
    to enforce these policies across the code."
}
```

———-

— **axiom.bib** —

```
@misc{Poiz85,
  author = "Poizat, B.",
```

```
  title = {{Cours de Th\'eorie des Mod\'eles}},
  comment = {Nur al-Mantiq wal-Ma'rifah, Villeurbanne, France},
  year = "1985"
}
```

———-

— **axiom.bib** —

```
@misc{Poly11a,
  author = "Unknown",
  title = {{Poly/ML}},
  link = "\url{http://www.polyml.org}",
  year = "2011"
}
```

———-

— **axiom.bib** —

```
@book{Popk94,
  author = "Popkorn, Sally",
  title = {{First Steps in Modal Logic}},
  publisher = "Cambridge University Press",
  year = "1994",
  isbn = "978-0-521-46482-6",
  link =
    "\url{https://cs.famaf.unc.edu.ar/~careces/ml18/downloads/popkorn.pdf}",
  paper = "Popk94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Post44,
  author = "Post, Emil L.",
  title = {{Recursively Enumerable Sets of Postive Integers and Their
            Decision Problems}},
  journal  = "Bull. Amer. Math Soc.",
  volume = "50",
  pages = "284-316",
  year = "1944",
  paper = "Post44.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pott06,
  author = "Pottier, Francois and Regis-Gianas, Yann",
  title = {{Stratified Type Inference For Generalized Algebraic Data
            Types}},
  booktitle = "POPL'06",
  year = "2006",
  publisher = "ACM",
  abstract =
    "We offer a solution to the type inference problem for an
    extension of Hindley and Milner's type system with generalized
    algebraic data types. Our approach is in two {\sl strata}. The
    bottom stratum is a core language that marries type
    {\sl inference} in the style of Hindley and Milner with type
    {\sl checking} for generalized algebraic data types. This results
    in an extremely simple specification, where case constructs must
    carry an explicit type annotation and type conversions must be
    made explicit. The top stratum consists of (two variants of) an
    independent {\sl shape inference} algorithm. This algorithm
    accepts a source term that contains some explicit type
    information, propagates this information in a local, predictable
    way, and produces a new source term that carries more explicit
    type information. It can be viewed as a preprocessor that helps
    produce some of the type annotations required by the bottom
    stratum. It is proven {\sl sound} in the sense that it never
    inserts annotations that could contradict the type derivation that
    the programmer has in mind.",
  paper = "Pott06.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Praw65,
  author = "Prawitz, Dag",
  title = {{Natural Deduction}},
  publisher = "Dover",
  isbn = "978-0-486-44655-4",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Pres19,
  author = "Pressler, Ron",
  title = {{Correctness and Complexity}},
  year = "2019",
  link = "\url{https://pron.github.io/posts/correctness-and-complexity}"
}
```

———-

— **axiom.bib** —

```
@misc{Prop13,
  author = "Propp, James",
  title = {{Real Analysis in Reverse}},
  year = "2013",
  link = "\url{https://arxiv.org/pdf/1204.4483.pdf}",
  abstract =
    "Many of the theorems of real analysis, against the background of
    the ordered field axioms, are equivalent to Dedekind completeness,
    and hence can serve as completeness axioms for the reals. In the
    course of demonstrating this, the article offers a tour of some
    less-familiar ordered fields, provides some of the relevant
    history, and considers pedagogical implications.",
  paper = "Prop13.pdf"
}
```

———-

1.2.17 Q

— **axiom.bib** —

```
@misc{QED94,
  author = "Anonymous",
  title = {{The QED Manifesto}},
  link = "\url{}",
  paper = "QED94.txt",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Qiux13,
  author = "Qiu, Xiaokang and Garg, Pranav and Stefanescu, Andrei and
            Madhusudan, P.",
  title = {{Natural Proofs for Structure, Data, and Separation}},
  year = "2013",
  link = "\url{http://madhu.cs.illinois.edu/dryad_full_version.pdf}",
  abstract =
    "We propose {\sl natural proofs} for reasoning with programs that
    manipulate data-structures against complex specifications --
    specifications that describe the structure of the heap, the data
    stored within it, and separation and framing of
    sub-structures. Natural proofs are a subclass of proofs that are
    amenable to completely automated reasoning, that provide sound but
    incomplete procedures, and that capture common reasoning tactics
```

```
    in program verification. We develop a dialect of separation logic
    over heaps, called DRYAD, with recursive definitions that avoids
    explicit quantification. We develop ways to reason with heaplets
    using classical logic over the theory of sets, and develop natural
    proofs for reasoning using proof tactics involving disciplined
    unfoldings and formula abstractions. Natural proofs are encoded
    into decidable theories of first-order logic so as to be
    discharged using SMT solvers.

    We also implement the technique and show that a large class of
    more than 100 correct programs that manipulate data-structures are
    amenable to full functional correctness using the proposed natural
    proof method. These programs are drawn from a variety of sources
    including standard data-structures, the Schorr-Waite algorithm for
    garbage collection, a large number of low-level C routines from
    the Glib library, the OpenBSD library and the Linux kernel, and
    routines from a secure verified OS-browser project. Our work is
    the first that we know of that can handle such a wide range of
    full functional verification properties of heaps automatically,
    given pre/post and loop invariant annotations. we believe that
    this work paves the way for the deductive verification technology
    to be used by programmers who do not (and need not) understand the
    internals of the underlying logic solvers, significantly
    increasing their applicability in building reliable systems.",
  paper = "Qiux13.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Quei94,
  author = "Queinnec, Christian",
  title = {{Lisp In Small Pieces}},
  publisher = "Cambridge University Press",
  year = "1994",
  isbn = "0-521-54566-8",
  keywords = "shelf"
}
```

———-

1.2.18 R

— **axiom.bib** —

```
@article{Rabe11,
  author = "Rabe, Florian and Kohlhase, Michael and Coen, Claudio Sacerdoti",
  title = {{A Foundational View on Integration Problems}},
  journal = "LNCS",
```

```
  volume = "6824",
  year = "2011",
  abstract =
    "The integration of reasoning and computation services across
    system and language boundaries is a challenging problem of
    computer science. In this paper, we use integration for the
    scenario where we have two systems that we integrate by moving
    problems and solutions between them. While this scenario is often
    approached from an engineering perspective, we take a foundational
    view. Based on the generic declarative language MMT, we develop a
    theoretical framework for system integration using theories and
    partial theory morphisms. Because MMT permits representations of
    the meta-logical foundations themselves, this includes integration
    across logics. We discuss save and unsafe integration schemes and
    devise a general form of safe integration.",
  paper = "Rabe11.pdf"
}
```

———-

— axiom.bib —

```
@article{Rabe12,
  author = "Rabe, Florian",
  title = {{A Query Language for Formal Mathematical Libraries}},
  journal = "LNCS",
  volume = "7362",
  year = "2012",
  abstract =
    "One of the most promising applications of mathematical knowledge
    management is search: Even if we restrict attention to the tiny
    fragment of mathematics that has been formalized, the amount
    exceeds the comprehension of an individual human.

    Based on the generic representation language MMT, we introduce the
    mathematical query langauge QMT: It combines simplicity,
    expressivity, and scalability while avoiding a commitment to a
    particular logical formalism. QMT can integrate various search
    paradigms such as unification, semantic web, or XQuery style
    queries, and QMT queries can span different mathematical
    libraries.

    We have implemented QMT as a part of the MMT API. This combination
    provides a scalable indexing and query engine that can be readily
    applied to any library of mathematical knowledge. While our focus
    here is on libraries that are available in a content markup
    language, QMT naturally extends to presentation and narration
    markup languages.",
  paper = "Rabe12.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Rabe15,
  author = "Rabe, Florian",
  title = {{Generic Literals}},
  journal = "LNCS",
  volume = "9150",
  year = "2015",
  abstract =
    "MMT is a formal framework that combines the flexibility of
    knowledge representation languages like OPENMATH with the formal
    rigor of logical frameworks like LF. It systematically abstracts
    from theoretical and practical aspects of individual formal
    languages and tries to develop as many solutions as possible
    generically.

    In this work, we allow MMT theories to declare user-devined
    literals, which makes literals as user-extensible as operators,
    axioms, and notations. This is particularly important for
    framework languages, which must be able to represent any choice of
    literals. Theoretically, our literals are introduced by importing
    a model that defines the denotations of some types and function
    symbols. Practically, MMT is coupled with a programming language,
    in which these models are defined.

    Our results are implemented in the MMT system. In particular,
    literals and computation on them are integrated with the parser
    and type checker.",
  paper = "Rabe15.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Rabe20,
  author = "Rabe, Florian",
  title = {{MMT: A Foundation-Independent Logical System}},
  year = "2020",
  link = "\url{https://vimeo.com/421123419}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Rado61,
  author = "Rado, Tibor",
  title = {{On Non-Computable Functions}},
  journal = "Bell System Technical Journal",
  volume = "41",
```

```
  number = "3",
  year = "1961",
  abstract =
    "The construction of non-computable functions used in this paper
    is based on the priciple that a finite, non-empty set of
    non-negative integers has a largest element. Also, this principle
    is used only for sets which are exceptionally well-defined by
    current standards. No enumeration of computable functions is used,
    and in this sense the diagonal process is not employed. This, it
    appears that an apparently self-evident principle, of constant use
    in every area of mathematics, yields non-constructive entities.",
  paper = "Rado61.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Rant93,
  author = "Ranta, Aarne",
  title = {{Type Theory and the Informal Language of Mathematics}},
  booktitle = "Types for Proofs and Programs",
  publisher = "Springer",
  year = "1993",
  pages = "352-365",
  paper = "Rant93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Rask80,
  author = "Raskovsky, Martin and Collier, Phil",
  title = {{From Standard to Implementation Denotational Semantics}},
  journal = "LNCS",
  volume = "54",
  pages = "94-139",
  year = "1980",
  abstract =
    "We are developing a compiler compiler. It takes as input the
    formal definition of a programming language in Denotational
    Semantics and produces as output a fairly efficient compiler
    written in a system programming language which in turn will
    produce code for a real machine. This work mainly deals with the
    code generation parts.",
  paper = "Rask80.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Raja06,
  author = "Raja, Amar and Rayner, Matthew and Sexton, Alan and
            Sorge, Volker",
  title = {{Towards a Parser for Mathematical Formula Recognition}},
  journal = "LNCS",
  volume = "4108",
  year = "2006",
  abstract =
    "For the transfer of mathematical knowledge from paper to
    electronic form, the reliable automatic analysis and understanding
    of mathematical texts is crucial. A robust system for this task
    needs to combine low level character recognition with higher level
    structural analysis of mathematical formulas. We present progress
    towards this goal by extending a database-driven optical character
    recognition system for mathematics with two high level analysis
    features. One extends and enhances the traditional approach of
    projection profile cutting. The second aims at integrating the
    recognition process with graph grammar rewriting by giving support
    to the interactive construction and validation of grammar
    rules. BOth approaches can be successfully employed to enhance the
    capabilities of our system to recognise and reconstruct compound
    mathematical expressions.",
  paper = "Raja06.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Rapo17,
  author = "Rapoport, Marianna and Kabir, Ifaz and He, Paul and
            Lhotak, Ondrej",
  title = {{A Simple Soundness Proof for Dependent Object Types}},
  year = "2017",
  link = "\url{https://arxiv.org/pdf/1706.03814v1.pdf}",
  abstract =
    "Dependent Object Types (DOT) is intended to be a core calculus
    for modelling Scala. Its distinguishing feature is abstract type
    members, fields in objects that hold types rather than
    values. Proving soundness of DOT has been surprisingly
    challenging, and existing proofs are complicated, and reason about
    multiple concepts at the same time (e.g. types, values,
    evaluation). To serve as a core calculus for Scala, DOT should be
    easy to experiment with and extend, and therefore its soundness
    proof needs to be easy to modify.

    This paper presents a simple and modular proof strategy for
    reasoning in DOT. The strategy separates reasoning about types
    from other concerns. It is centered around a theorem that connects
    the full DOT type system to a restricted variant in which the
    challenges and paradoxes caused by abstract type members are
```

```
    eliminated. Almost all reasoning in the proof is done in the
    intuitive worlds of this restricted type system. Once we have the
    necessary results about types, we observe that the other aspects
    of DOT are mostly standard and can be incorporated into a
    soundness proof using familiar techniques known from other
    calculi.

    Our paprer comes with a machine-verified version of the proof in
    Coq.",
  paper = "Rapo17.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@InCollection{Rect89,
  author = "Rector, D. L.",
  title = {{Semantics in Algebraic Computation}},
  booktitle = "Computers and Mathematics",
  publisher = "Springer-Verlag",
  year = "1989",
  pages = "299-307",
  isbn = "0-387-97019-3",
  abstract =
    "I am interested in symbolic computation for theoretical research in
    algebraic topology. Most algebraic computations in topology are hand
    calculations; that is, they can be accomplished by the researcher in
    times ranging from hours to weeks, and they are aimed at discovering
    general patterns rather than producing specific formulas understood in
    advance. Furthermore, the range of algebraic constucts used in such
    calculations is very wide.",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@article{Reed97,
  author = "Reed, Mary Lynn",
  title = {{Algebraic Structure of Genetic Inheritance}},
  journal = "Bulletin of the American Mathematical Society",
  year = "1997",
  volume = "34",
  number = "2",
  month = "April",
  pages = "107--130",
  algebra = "\newline\refto{domain ALGSC AlgebraGivenByStructuralConstants}",
  link="\url{http://www.ams.org/bull/1997-34-02/S0273-0979-97-00712-X/S0273-0979-97-00712-X.pdf}",
  abstract =
```

```
   "In this paper we will explore the nonassociative algebraic structure
    that naturally ocurs as genetic informatin gets passed down through
    the generations. While modern understanding of genetic inheritance
    initiated with the theories of Charles Darwin, it was the Augustinian
    monk Gregor Mendel who began to uncover the mathematical nature of the
    subject. In fact, the symbolism Mendel used to describe his first
    results (e.g. see his 1866 paper {\sl Experiments in
    Plant-Hybridization} is quite algebraically suggestive. Seventy four
    years later, I.M.H. Etherington introduced the formal language of
    abstract algebra to the study of genetics in his series of seminal
    papers. In this paper we will discuss the concepts of genetics that
    suggest the underlying algebraic structure of inheritance, and we will
    give a brief overview of the algebras which arise in genetics and some
    of their basi properties and relationships. With the popularity of
    biologically motivated mathematics continuing to rise, we offer this
    survey article as another example of the breadth of mathematics that
    has biological significance. The most comprehensive reference for the
    mathematical research done in this area (through 1980) is
    Worz-Busekros.",
  paper = "Reed97.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Rees82,
  author = "Rees, Jonathan and Adams, Norman I",
  title = {{T: A Dialect of Lisp or, LAMBDA: The Ultimate Software Tool}},
  booktitle = "Sym. on Lisp and Functional Programming",
  pages = "114-122",
  year = "1982",
  abstract =
    "The T proejct is an experiment in language design and
    implementation. Its purpose is to test the thesis developed by
    Steele and Sussman in their series of papers about the Scheme
    language: that Scheme may be used as the basis for a practical
    programming language of exceptional expressive power, and that
    implementations of Scheme could perform better than other Lisp
    systems, and competitively with implementations of programming
    languages, such as C and Bliss, which are usually considered to be
    inherently more efficient than Lisp on conventaional machine
    architectures. We are developing a portable implementation of T,
    currently targeted for the VAX under the Unix and VMS operating
    systems and for the Apollo, a MC68000-based workstation.",
  paper = "Rees82.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Reic47,
  author = "Reichenbach, Hans",
  title = {{Elements of Symbolic Logic}},
  year = "1947",
  publisher = "The MacMillan Company",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Reid17,
  author = "Reid, Alastair",
  title = {{How can you trust formally varified software}},
  link = "\url{https://media.ccc.de/v/34c3-8915-how_can_you_trust_formally_verified_software#t=12}",
  year = "2017",
  abstract =
    "Formal verification of software has finally started to become viable:
    we have examples of formally verified microkernels, realistic
    compilers, hypervisors, etc. These are huge achievements and we can
    expect to see even more impressive results in the future but the
    correctness proofs depend on a number of assumptions about the Trusted
    Computing Base that the software depends on. Two key questions to ask
    are: Are the specifications of the Trusted Computing Base correct? And
    do the implementations match the specifications? I will explore the
    philosophical challenges and practical steps you can taek in answering
    that question for one of the major dependencies: the hardware your
    software runs on. I will describe the combination of formal
    verification and testing that ARM uses to verify the processor
    specification and I will talk about our current challenge: getting the
    specification down to zero bugs while the architecture continues to
    evolve."
}
```

———-

— **axiom.bib** —

```
@inproceedings{Reis14,
  author = "Reis, Leonardo Vieira dos Santos and Dilorio, Oliveira and
            Bigonha, Roberto S.",
  title = {{A Mixed Approach for Building Extensible Parsers}},
  booktitle = "Programming Language",
  publisher = "Springer",
  volume = "LNCS 8771",
  pages = "1-15",
  year = "2014",
  abstract =
    "For languages whose syntax is fixed, parsers are usually built
    with a static structure. The implementation of features like macor
    mechanisms or extensible languages requires the use of parsers
```

```
    that may be dynamically extended. In this work, we discuss a mixed
    approach for building efficient top-down dynamically extensible
    parsers. Our view is based on the fact that a large part of the
    parser code can be statically compiled and only the parts that are
    dynamic should be interpreted for a more efficient processing. We
    propose the generation of code for the base parser, in which hooks
    are included to allow efficient extension of the underlying
    grammar and activation of a grammar interpreter whenever it is
    necessary to use an extended syntax. As a proof of concept, we
    present a prototype implementation of a parser generator using
    Adaptable Parsing Expression Grammars (APEG) as the underlying
    method for syntax definition. We shos that APEG has features which
    allow an efficient implementation using the proposed mixed
    approach.",
  paper = "Reis14.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Rekd14,
  author = "Rekdal, Ole Bjorn",
  title = {{Academic Urban Legends}},
  journal = "Social Studies of Science",
  volume = "44",
  number = "4",
  pages = "638-654",
  year = "2014",
  abstract =
    "Many of the messages presented in respectable scientific
    publications are, in fact, based on various forms of rumors. Some
    of these rumors appear so frequently, and in such complex,
    colorful, and entertaining ways that we can think of them as
    academic urban legends. The explanation for this phenomenon is
    usually that the authors have lazily, sloppily, or fraudulently
    employed sources, and peer reviewers and editors have not
    discovered these weaknesses in the manuscripts during
    evaluation. To illustrate this phenomenon, I draw upon a
    remarkable case in which a decimal point error appears to have
    misled millions into believing that spinach is a good nutritional
    source of iron. Through this example, I demonstrate how an
    academic urban legend can be conceived and born, and can continue
    to grow and reproduce within academia and beyond.",
  paper = "Rekd14.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Remy89,
  author = "Remy, Didier",
  title = {{Typechecking Records and Variants in a Natural Extension of ML}},
  booktitle = "POPL 89",
  isbn = "978-0-89791-294-5",
  year = "1989",
  publisher = "ACM",
  link = "\url{https://www.cs.cmu.edu/~aldrich/courses/819/row.pdf}",
  abstract =
    "We describe an extension of ML with records where inheritance is
    given by ML generic polymorphism. All common operations on records but
    concatenation are supported, in particular, the free extension of
    records. Other operations such as renaming of fields are added.  The
    solution relies on an extension of ML, where the language of types is
    sorted and considered modulo equations, and on a record extension of
    types. The solution is simple and modular and the type inference
    algorithm is efficient in practice.",
  paper = "Remy89.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Remy92,
  author = "Remy, Didier",
  title = {{Extensions to ML type system with a sorted equation theory
            on types}},
  type = "research report",
  institution = "INRIA",
  number = "RR-1766",
  year = "1992",
  abstract =
    "We extend the ML language by allowing a sorted regular equational
    theory on types for which unification is decidable and unitary. We
    prove that the extension keeps principlal typings and subject
    reduction. A new set of typing rules is proposed so that type
    generalization is simpler and more efficient. We consider typing
    problems as general unification problems, which we solve with a
    formalism of unificands. Unificands naturally deal with sharing
    between types and lead to a more efficient type inference
    algorithm. The use of unificands also simplifies the proof of
    correctness of the algorithm by splitting it into more elementary
    steps.",
  paper = "Remy92.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
#article{Renn99,
  author = "Rennert, Nicolas and Valibouze, Annick",
  title = {{Computation of Resolvents using Cauchy Modules}},
  journal = "Experimental Mathematics",
  volume = "8",
  number = "4",
  pages = "351-366",
  year = "1999",
  comment = "French",
  abstract =
    "We give a new and efficient algorithm to compute some
    characteristic polynomials using Cauchy modules. This algorithm is
    used for the computation of absolute resolvents and
    multiresolvents, which are essential tools in constructive Galois
    theory.",
  paper = "Renn99.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Rest00,
  author = "Restall, Greg",
  title = {{An Introduction to Substructural Logics}},
  year = "2000",
  publisher = "Routledge",
  isbn = "0-415-21534-X"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Reyn90,
  author = "Reynaud, Jean-Claude",
  title = {{Putting Algebraic Components Together: A Dependent Type
            Approach }},
  booktitle = "DISCO 1990",
  year = "1990",
  pages = "141-150",
  publisher = "Springer",
  abstract =
    "We define a framework based on dependent types for putting
    algebraic components together. It is defined with freely generated
    categories. In order to preserve initial, loos and constrainted
    semantics of components, we introduce the notion of
    SPEC-categories which look like specific finitely co-complete
    categories. A constructive approach which includes
    parameterization techniques is sued to define new components from
    basic predefined ones. The problem of the internal coding of
```

```
    external signature symbols is introduced.",
  paper = "Reyn90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Reyn74,
  author = "Reynolds, John C.",
  title = {{Towards a Theory of Type Structure}},
  booktitle = "Colloquim on Programming",
  year = "1974",
  pages = "408-425",
  paper = "Reyn74.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Reyn80,
  author = "Reynolds, John C.",
  title = {{Using Category Theory to Design Implicit Conversions and
           Generic Operators}},
  booktitle = "Lecture Notes in Computer Science",
  year = "1980",
  abstract =
    "A generalization of many-sorted algebras, called category-sorted
    algebras, is defined and applied to the language-design problem of
    avoiding anomalies in the interaction of implicit conversions and
    generic operators. The definition of a simple imperative language
    (without any binding mechanisms) is used as an example.",
  paper = "Reyn80.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Reyn81,
  author = "Reynolds, John C.",
  title = {{The Craft of Programming}},
  publisher = "Prentice-Hall",
  year = "1981",
  isbn = "0-13-188862-5",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Reyn83,
  author = "Reynolds, John C.",
  title = {{Types, Abstraction and Parametric Polymorphism}},
  booktitle = "Information Processing 83",
  publisher = "Elsevier Science Publishers",
  year = "1983",
  abstract =
    "We explore the thesis that type structure is a syntactic
    discipline for maintaining levels of abstraction. Traditionally,
    this view has beeen formalized algebraically, but the algebraic
    approach fails to encompass higher-order functions. For this
    purpose, it is necessary to generalize homomorphic functions to
    relations; the result is an ‘‘abstraction’’ theorem that is
    applicable to the typed lambda calculus and various extensions,
    including user-defined types.

    Finally, we consider polymorphic functions, and show that the
    abstraction theorem captures Strachey's concept of parametric, as
    opposed to ad hoc, polymorphism.",
  paper = "Reyn83.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Reyn84,
  author = "Reynolds, John C.",
  title = {{Polymorphism is not Set-theoretic}},
  booktitle = "Proc Semantics of Data Types",
  pages = "145-156",
  year = "1984",
  link = "\url{https://hal.inria.fr/inria-00076261/document}",
  abstract =
    "The polymorphic, or second-order, typed lambda calculus is an
    extension of the typed lambda calculus in which polymorphic functions
    can be defined. In this paper that the standard set-theoretic model of
    the ordinary typed lambda calculus cannot be extended to model this
    language extension.",
  paper = "Reyn84.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Reyn91,
  author = "Reynolds, John C.",
  title = {{The Coherence of Languages with Intersection Types}},
  booktitle = "TACS 91",
  year = "1991",
  abstract =
    "When a programming language has a sufficiently rich type structure,
    there can be more than one proof of the same typing judgement;
    potentially this can lead to semantic ambiguity since the semantics of
    a typed language is a function of such proofs. When no such ambiguity
    arises, we say that the language is coherent. In this paper we prove
    the coherence of a class of lambda-calculus-based languages that use
    the intersection type discipline, including both a purely functional
    programming language and the Algol-like programming language Forsythe.",
  paper = "Reyn91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Reyn94,
  author = "Reynolds, John C.",
  title = {{An Introduction to the Polymorphic Lambda Calculus}},
  year = "1994",
  paper = "Reyn94.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inbook{Reyn97,
  author = "Reynolds, John C.",
  title = {{The Essence of Algol}},
  booktitle = "ALGOL-like languages, Volume 1",
  publisher = "Birkhauser Boston Inc.",
  chapter = "1",
  pages = "67-88",
  isbn = "0-8176-3880-6",
  year = "1997",
  abstract =
    "Although Algol 60 has been uniquely influential in programming
    language design, its descendants have been signicantly different than
    their prototype. In this paper, we enumerate the principles that we
    believe embody the essence of Algol, describe a model that satisfies
    these principles, and illustrate this model with a language that,
    while more uniform and general, retains the character of Algol.",
  paper = "Reyn97.pdf",
```

```
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Reyn08,
  author = "Reynolds, John C.",
  title = {{An Introduction to Separation Logic}},
  year = "2008",
  paper = "Reyn94.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Rich18,
  author = "Rich, Albert and Scheibe, Patrick and Abbasi, Nasser",
  title = {{Rule-based Integration: An Extensive System of Symbolic
            Integration Rules}},
  journal = "Journal of Open Source Software",
  volume = "3",
  number = "32",
  year = "2018",
  abstract =
    "Finding the antiderivative of expressions is often challenging
    and requires advanced mathematical skills even for simple looking
    problems. Computer algebra aystems (CAS) like Mathematica (Wolfram
    Research, Inc., Champaign, IL), Maple (Maplesoft, a division of
    Waterloo Maple Inc., Waterloo, Ontario), and Maxima
    (maxima.sourceforge.net) provide integrators to compute
    antiderivatives symbolically. However, these systems give no
    insight as to how an antiderivative is found or why it could not
    be computed. Also, they use advanced methods incomprehensible to
    humans that often result in huge antiderivatives unnecessarily
    involving special functions.",
  paper = "Rich18.pdf",
  keywords = "axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@mastersthesis{Rijk12,
  author = "Rijke, Eghert",
  title = {{Homotopy Type Theory}},
  school = "Utrecht University",
```

```
  year = "2012",
  paper = "Rijk12.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Risc76,
  author = "Risch, Robert H.",
  title = {{Implicitly Elementary Integrals}},
  journal = "Proc. Amer. Math.",
  volume = "57",
  number = "1",
  pages = "1-7",
  year = "1976",
  paper = "Risc76.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Risc79,
  author = "Risch, Robert H.",
  title = {{Algebraic Properties of the Elementary Functions of Analysis}},
  journal = "American Journal of Mathematics",
  volume = "101",
  number = "4",
  pages = "743-759",
  year = "1979",
  abstract =
    "The elementary functions of a complex variable $z$ are those
    functions built up from the rational functions of $z$ by
    exponentiation, taking logarithms, and algebraic operations. The
    purpose of this paper is first, to prove a 'structure theorem'
    which shows that if an algebraic relation holds among a set of
    elementary functions, then they must satisfy an algebraic relation
    of a special kind. Then we make four applications of this theorem,
    obtaining both new and old results which are described here
    briefly (and imprecisely).
    \begin{enumerate}
    \item An algorithm is given for telling when two elementary
    expressions define the same function.
    \item A characterization is derived of those ordinary differential
    equations having elementary solutions
    \item The four basic functions of elementary calculus -- exp, log,
    tan, tan$^{-1}$, -- are shown to be 'irredundant'
    \item A characterization is given of elementary functions
    possessing elementary inverses.
```

```
    \end{enumerate}",
  paper = "Risc79.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
\article{Ritt25,
  author = "Ritt, J.F.",
  title = {{Elementary Functions and their Inverses}},
  journal = "Transactions of the American Mathematical Society",
  volume = "27",
  pages = "68-90",
  year = "1925",
  paper = "Ritt25.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Roan19,
  author = "Roanes-Lozano, Eugenio and Galan-Garcia, Jose Luis and
            Solano-Macias, Carmen",
  title = {{Some Reflections About the Success and Impact of the
            Computer Algebra System DERIVE with a 10-Year Time
    Perspective}},
  journal = "Mathematics in Computer Science",
  volume = "13",
  number = "3",
  pages = "417-431",
  year = "2019",
  abstract =
    "The computer algebra system DERIVE had a very important impact in
    teaching mathematics, mainly in the 1990's. The authors analyze
    the possible reasons for its success and impact and give personal
    conclusions based on the facts collected. More than 10 years after
    it was discontinued it is still used for teaching and several
    scientific papers (mostly devoted to educational issues) still
    refer to it. A summary of the history, journals and conferences
    together with a brief bibliographic analysis are included.",
  paper = "Roan19.pdf",
  keywords = "axiomref, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Robb85,
  author = "Robbiano, Lorenzo",
  title = {{Term Orderings on the Polynomial Ring}},
  booktitle = "European COnference on Computer Algebra",
  publisher = "Springer",
  pages = "513-517",
  year = "1985",
  comment = "LNCS 204",
  paper = "Robb85.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Robi65,
  author = "Robinson, J.A.",
  title = {{A Machine-Oriented Logic Based on the Resolution Principle}},
  journal = "ACM",
  volume = "12",
  pages = "23-41",
  year = "1965",
  abstract =
    "Theorem-proving on the computer, using procedures based on the
    fundamental theorem of Herbrand concerning the first-order predicate
    calculus, is examined with a view towards improving the efficiency and
    widening the range of practical applicability of these procedures.  A
    elose analysis of the process of substitution (of terms for
    variables), and the process of truth-functional analysis of the
    results of such substitutions, reveals that both processes can be
    combined into a single new process (called resolution), iterating
    which is vastty more effieient than the older cyclic procedures
    consisting of substitution stages alternating with truth-functional
    analysis stages.

    The theory of the resolution process is presented in the form of a
    system of first-order logic with .just one inference principle (the
    resolution principle).  The completeness of the system is proved; the
    simplest proof-procedure based on the system is then the direct
    implementation of the proof of completeness.  Howewer, this procedure is
    quite inefficient, and the paper concludes with a discussion of
    several principles (called search principles) which are applicable to
    the design of efficient proof-procedures employing resolution as the
    basic logical process.",
  paper = "Robi65.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Robi80,
  author = "Robinson, J.A. and Sibert, E.E.",
  title = {{Loglisp: An Alternative to Prolog}},
  type = "technical report",
  institution = "University of Syracuse",
  number = "80-7",
  year = "1980",
  paper = "Robi80.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Robi96,
  author = "Robinson, J. S. Derek",
  title = {{A Course in the Theory of Groups}},
  year = "1996",
  series = "Graduate Texts in Mathematics",
  isbn = "978-1-4612-6443-9",
  publisher = "Springer"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Roes19,
  author = "Roessle, Ian and Verbeek, Freek and Ravindran, Binoy",
  title = {{Formally Verified Big Step Semantics out of x86-64 Binaries}},
  booktitle = "CPP'19",
  year = "2019",
  publisher = "ACM",
  abstract =
    "This paper presents a methodology for generating formally proven
    equivalence theorems between decompiled x86-64 machine code and
    big step semantics. These proofs are built on top of two
    additional contributions. First, a robust and tested formal x86-64
    machine model containing small step semantics for 1625
    instructions. Second, a decompilation into logic methodology
    supporting both x86-64 assembly and machine code at large
    scale. This work enables blackbox binary verification, i.e.,
    formal verification of a binary where source code is
    unavailable. As such, it can be applied to safety-critical systems
    that consist of legacy components, or components whose source code
    is unavailable due to proprietary reasons. The methodology
    minimizes the trusted code base by leveraging machine-learned
    semantics to build a formal machine model. We apply the
    methodolgoy to several case studies, including binaries that
    heavily rely on the SSE2 floating-point instruction set, and
    binaries that are obtained by compiling code that is obtained by
```

```
    inlining assembly into C code.",
  paper = "Roes19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Roit13,
  author = "Roitman, Judith",
  title = {{Introduction to Modern Set Theory}},
  link = "\url{http://www.math.ku.edu/~roitman/SetTheory.pdf}",
  year = "2013",
  paper = "Roit13.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Roll1691,
  author = "Rolle, Michel",
  title = {{Rolle's Therem}},
  year = "1691",
  link = "\url{https://en.wikipedia.org/wiki/Rolle%27s\_theorem}",
  abstract =
    "If a real-valued function $f$ is continuous on a proper closed interval
    $[a,b]$, differentiable on the open interval $(a,b)$, and $f(a)=f(b)$,
    then there exists at least one $c$ in the open interval $(a,b)$ such
    that $f^\prime(c)=0$."
}
```

———-

— **axiom.bib** —

```
@misc{Ross13,
  author = "Rossberg, Andreas",
  title = {{HaMLet: To Be Or Not To Be Standard ML}},
  year = "2013",
  link = "\url{https://people.mpi-sws.org/~rossberg/hamlet}",
  paper = "Ross13.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Rosu20,
```

```
  author = "Rosu, Grigore and Lucanu, Dorel and Guth, Dwight",
  title = {{The K Framework}},
  year = "2020",
  link = "\url{http://www.kframework.org/index.php/Main_Page}"
}
```

———-

— **axiom.bib** —

```
@book{Russ1920,
  author = "Russell, Bertrand",
  title = {{Introduction to Mathematical Philosophy}},
  publisher = "George Allen and Unwin, Ltd",
  year = "1920",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Russ92,
  author = "Russinoff, David M.",
  title = {{A Verified Prolog Compiler for the Warren Abstract Machine}},
  journal = "Journal of Logic Programming",
  volume = "13",
  number = "4",
  pages = "367-412",
  year = "1992",
  abstract =
    "We extend the theory of Prolog to provide a framework for the
    study of Prolog compilation technology. For this purpose, we first
    demonstrate the semantic equivalence of two Prolog interpreters: a
    conventional SLD-refutation procedure and one that employs
    Warren's ``last call'' optimization. Next, we formally define the
    Warren Abstract Machine (WAM) and its instruction set and present
    a Prolog compiler for the WAM. Finally, we prove that the WAM
    execution of a compiled Prolog program produces the same result as
    the interpretation of its source.",
  paper = "Russ92.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Rute20,
  author = "Rute, Jason",
  title = {{Communicating with Lean}},
```

```
  year = "2020",
  link = "\url{https://github.com/jasonrute/communicating-with-lean/blob/master/communicate_with_lean.ipynb}"
}
```

———-

— **axiom.bib** —

```
@article{Ruzi89,
  author = "Ruzicka, Peter and Privara, Igor",
  title = {{An Almost Linear Robinson Unification Algorithm}},
  journal = "Acta Informatica",
  volume = "27",
  pages = "61-71",
  year = "1989",
  abstract =
    "Further asymptotical improvement of original Robinson's unification
    idea is presented.  By postponing the so-called occur-check in Corbin
    and Bidoit's quadratic rehabilitation of the Robinson algorithm at the
    end of unification an almost linear unification algorithm is obtained.
    In the worst case, the resulting algorithm has the time complexity
    O(p.A(p)), where p is the size of input terms and A is the inverse to
    the Ackermann function.  Moreover, the practical experiments are
    summarized comparing Corbin and Bidoit's quadratic algorithm with the
    resulting almost linear unification algorithm based on Robinson's
    principle. ",
  paper = "Ruzi89.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Ryde88,
  author = "Rydeheard, D. E. and Burstall, R. M.",
  title = {{Computational Category Theory}},
  publisher = "Prentice Hall",
  year = "1988",
  isbn = "978-0131627369"
}
```

———-

— **axiom.bib** —

```
@techreport{Ryde99,
  author = "Ryder, Chris and Thompson, Simon",
  title = {{Aldor meets Haskell}},
  type = "technical report",
  institution = "University of Kent",
  number = "21762",
```

```
  year = "1999",
  abstract =
    "The aim of this project was to attempt to output a Haskell
    representation of the Aldor compiler's abstract syntax tree. The
    purpose of this is to enable the representation to be executed and
    to give an experimental platform in which to look at how to
    circumvent some of the limitations of the Aldor compilers type
    checker.",
  paper = "Ryde99.pdf",
  keywords = "axiomref"
}
```

———-

1.2.19 S

— **axiom.bib** —

```
@inproceedings{Saib97,
  author = "Saibi, Amokrane",
  title = {{Typing Algorithm in Type Theory with Inheritance}},
  booktitle = "Symp. on Principles of Programming Languages",
  publisher = "ACM",
  pages = "292-301",
  year = "1997"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Saib99,
  author = "Saibi, Amokrane",
  title = {{Outils G\'en\'eriques de mod\'elisation et de
            d\'emonstration pour la Formalisation des Math\'ematiques
    en th\'eorie des Types, Application \'a la th\'eorie des
    cat\'egories}},
  school = "University of Paris 6",
  year = "1999"
}
```

———-

— **axiom.bib** —

```
@misc{Sanc19,
  author = "Sanchez-Stern, Alex and Alhessi, Yousef and Saul, Lawrence
            and Lerner, Sorin",
  title = {{Generating Correctness Proofs with Neural Networks}},
  year = "2019",
```

```
  link = "\url{https://arxiv.org/pdf/1907.07794.pdf}",
  abstract =
    "Foundational verification allows programmers to build software
    which has been empirically shown to have high levels of assurance
    in a variety of important domains. However, the cost of producing
    foundationally verified software remains prohibitively high for
    most projects, as it requires significant manual effort by highly
    trained experts. In this paper we present Proverbot9001 a proof
    search system using machine learning techniques to produce proofs
    of software correctness in interactive theorem provers. We
    deomonstrate Proverbot9001 on the proof obligations from a large
    practical proof project, the CompCert verified C compiler, and
    show that it can effectively automate what was previously manual
    proofs, automatically solving 15.77\% of proofs in our test
    dataset. This corresponds to an over 3X improvement over the prior
    state of the art machine learning technique for generating proofs
    in Coq.",
  paper = "Sanc19.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Sann86,
  author = "Sannella, Donald",
  title = {{Extended ML: An Institution-Independent Framework for
            Formal Program Development}},
  booktitle = "Proc. Workshop on Category Theory and Computer Programming",
  journal = "LNCS",
  volume = "240",
  pages = "364-389",
  year = "1986",
  paper = "Sann86.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Sann86a,
  author = "Sannella, Donald",
  title = {{Formal Specification of ML Programs}},
  link =
  "\url{http://www.lfcs.inf.ed.ac.uk/reports/86/ECS-LFCS-86-15/ECS-LFCS-86.15.ps}",
  year = "1986",
  abstract =
    "These notes were written to accompany lectures on program
    specification which formed part of a course on functional
    programming in ML. Functions can be specified using a
    specification language obtained by extending ML with
    (non-executable) first-order axioms. Simple inductive proofs
```

```
    suffice to show that in ML function satisfies such a
    specification. This approach can also be used to specify and
    verify larger programs built from smaller pieces using ML's
    modularisation facilities. Examples are used to illustrate the
    methods discussed.",
  paper = "Sann86a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Sann87,
  author = "Sannella, Donald and Tarlecki, Andrzej",
  title = {{On Observational Equivalence and Algebraic Specification}},
  journal = "J. of Computer and System Sciences",
  volume = "34",
  pages = "150-178",
  year = "1987",
  abstract =
    "The properties of a simple and natural notion of observational
    equivalence of algebras and the corresponding specification-building
    operation are studied.  We begin with a defmition of observational
    equivalence which is adequate to handle reachable algebras only, and
    show how to extend it to cope with unreachable algebras and also how
    it may be generalised to make sense under an arbitrary institution.
    Behavioural equivalence is treated as an important special case of
    observational equivalence, and its central role in program development
    is shown by means of an example.",
  paper = "Sann87.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Sann88,
  author = "Sannella, Donald and Tarlecki, Andrzej",
  title = {{Specifications in an Arbitrary Institution}},
  journal = "Information and Computation",
  volume = "76",
  pages = "165-210",
  year = "1988",
  abstract =
    "A formalism for constructing and using axiomatic specifications in an
    arbitrary logical system is presented.  This builds on the framework
    provided by Goguen and Burstalls work on the notion of an institution
    as a formalisation of the concept of a logical system for writing
    specifications.  We show how to introduce free variables into the
    sentences of an arbitrary institution and how to add quantitiers which
    bind them.  We use this foundation to define a set of primitive
```

```
    operations for building specifications in an arbitrary institution
    based loosely on those in the ASL kernel specification language.  We
    examine the set of operations which results when the definitions are
    instantiated in institutions of total and partial tirst-order logic
    and compare these with the operations found in existing specification
    languages.  We present proof rules which allow proofs to be conducted
    in specifications built using the operations we define.  Finally, we
    introduce a simple mechanism for defining and applying parameterised
    specifications and briefly discuss the program development process.",
  paper = "Sann88.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Sann89,
  author = "Sannella, Donald and Tarlecki, Andrzej",
  title = {{Toward Formal Development of ML Programs: Foundations and
            Methodology (Extended Abstract}},
  booktitle = "Int. Conf. on Theory and Practice of Software Development",
  pages = "375-389",
  publisher = "Springer",
  year = "1989",
  abstract =
    "A methodology is presented for the formal development of modular
    Standard ML programs from specifications. Program development proceeds
    via a sequence of design (modular decomposition), coding and
    refinement steps. For each of these three kinds of step, conditions
    are given which ensure the correctness of the result. These conditions
    seem to be as weak as possible under the constraint of being
    expressible as local interface matching requirements.",
  paper = "Sann89.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Sann91,
  author = "Sannella, D. and Tarlecki, A.",
  title = {{Formal Program Development in Extended ML for the Working
            Programmer}},
  booktitle = "3rd BCS/FACS Workshop on Refinement",
  publisher = "Springer",
  pages = "99-130",
  year = "1991",
  abstract =
    "Extened ML is a framework for the formal development of programs
    in the Standard ML programming language from high-level
    specifications of their required input/output behavior. It
    strongly supports the development of modular programs consisting
```

```
    of an interconnected collection of generic and reusable units. The
    Extended ML framework includes a methodology for formal program
    development which establishes a number of ways of proceeding from
    a given specification of a programming task towards a
    program. Each such step gives rise to one or more proof
    oblisgations which must be proved in order to establish the
    correctness of that step. This paper is inteded as a user-oriented
    summary of the Extended ML language and mthodology. Theoretical
    technicalities are avoided whenever possible, with emphasis placed
    on the practical aspects of formal program development. An
    extended example of a complete program development in Extended ML
    is included.",
  paper = "Sann91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Sann91a,
  author = "Sannella, Donald and Tarlecki, Andrzej",
  title = {{Extended ML: Past, Present and Future}},
  journal = "LNCS",
  volume = "534",
  pages = "297-322",
  year = "1991",
  abstract =
    "An overview of past, present and future work on the Extended ML
    formal program development framework is given, with emphasis on
    two topics of current active research: the semantics of the
    Extened ML specification langauge, and tools to support formal
    program development.",
  paper = "Sann91a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Sann97,
  author = "Sannella, Donald and Tarlecki, Andrzej",
  title = {{Essential Concepts of Algebraic Specification and Program
            Development}},
  journal = "Formal Aspects of Computing",
  volume = "9",
  pages = "229-269",
  year = "1997",
  abstract =
    "The main ideas underlying work on the model-theoretic foundations
    of algebraic specification and formal program development are
```

```
    presented in an informal way. An attempt is made to offer an
    overall view, rather than new results, and to focus on the basic
    motivation behind the technicalities presented elsewhere.",
  paper = "Sann97.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Sann99,
  author = "Sannella, Donald and Tarlecki, Andrzej",
  title = {{Algebraic Methods for Specification and Formal Development
            of Programs}},
  journal = "ACM Computing Surveys",
  volume = "31",
  year = "1999",
  paper = "Sann99.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Sann12,
  author = "Sannella, Donald and Tarlecki, Andrzej",
  title = {{Foundations of Algebraic Specification and Formal Software
            Development}},
  publisher = "Springer",
  year = "2012",
  isbn = "978-3-642-17336-3",
  paper = "Sann12.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Sanu15,
  author = "Sanuki, Masaru and Inaba, Daiju and Sasaki, Tateaki",
  title = {{Computation of GCD of Sparse Multivariate Polynomials by
            Extended Hensel Construction}},
  booktitle = "17th Int. Symp. on Symbolic and Numeric Algorithms for
               Scientific Computing",
  publisher = "IEEE",
  year = "2015",
  abstract =
    "Let $F(x,u_1,\ldots,u_i)$ be a squarefree multivariate polynomial
    in main variable $x$ and sub-variables $u_1\ldots u_i$. We say
```

```
    that the leading coefficient (LC) of $F$ is singular if it
    vanishes at the origin of the sub-variables. A representative
    algorithm for non-sparse multivariate polynomial GCD is the EZ-GCD
    algorithm, which is based on the generalized Hensel construction
    (GHC). In order to apply the GHC easily, we requires 1) the LC of
    $F$ is non-singular, 2) $F(x,0,\ldots,0)$ is squarefree, and 3)
    the initial Hensel factor of GCD is ``lucky''. These requirements
    are usually satisfied by the ``nonzero substitution'', i.e. to
    shift the origin of sub-variables. However, the nonzero
    substitution may cause a drastic increase of the number of terms
    of $F$ if $F$ is sparse. In 1993, Sasaki and Kako proposed the
    extended Hensel construction (EHC) which does not perform the
    nonzero substitution even if the LC is singular. Using the EHC,
    Inaba implemented an algorithm of multivariate polynomial
    factorization and verified that it is very useful for sparse
    polynomials. In this paper, we apply the EHC for the computation
    of GCD of sparse multivariate polynomials. In order to find a
    lucky initial factor, we utilize the weighting of sub-variables,
    etc. Our naive implementation in Maple shows that our algorithm is
    comparable in performance to Maple's GCD routine base on the
    sparse interpolation.",
  paper = "Sanu15.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Sark04,
  author = "Sarkar, Dipanwita and Waddell, Oscar and Dybvig, R. Kent",
  title = {{A Nanopass Infrastructure for Compiler Education}},
  booktitle = "9th ACM SIGPLAN",
  publisher = "ACM",
  pages = "201-212",
  year = "2004",
  isbn = "1-58113-905-5",
  abstract =
    "A compiler structured as a small number of monolithic passes is
     difficult to understand and difficult to maintain. The steep
     learning curve is daunting, and even experienced developers find
     that modifying existing passes is difficult and often introduces
     subtle and tenacious bugs. These problems are especially
     frustrating when the developer is a student in a compiler
     class. An attractive alternative is to structure a compiler as a
     collection of many fine-grained passes, each of which performs a
     single task. This structure aligns the implementation of a
     compiler with its logical organization, simplifying development,
     testing, and debugging. This paper describes the methodology and
     tools comprising a framework for constructing such compilers.",
  paper = "Sark04.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Sche18,
  author = "Scheibe, Patrick",
  title = {{Rubi - The Rule-based Integrator for Mathematica}},
  link = "\url{https://community.wolfram.com/groups/-/m/t/1421180}",
  year = "2018",
  keywords = "axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Sche18a,
  author = "Scheibe, Patrick",
  title = {{Rubi - The Rule-based Integrator for Mathematical Expressions}},
  link = "\url{https://halirutan.de/programming/Rubi}",
  year = "2018",
  keywords = "axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Schm89,
  author = "Schmidt-Schauss, M.",
  title = {{Computational Aspects of an Order-Sorted Logic with Term
           Declarations}},
  publisher = "Springer",
  isbn = "978-3-540-51705-4",
  year = "1989"
}
```

———-

— **axiom.bib** —

```
@article{Schn02,
  author = "Schnoebelen, Ph.",
  title = {{The Complexity of Temporal Logic Model Checking}},
  journal = "Advances in Modal Logic",
  volume = "4",
  pages = "1-44",
  year = "2002",
  paper = "Schn02.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Scho24,
  author = "Schoenfinkel, M.",
  title = {{Uber die Bausteine der mathematischen Logik}},
  year = "1924",
  pages = "305-316"
}
```

———-

— **axiom.bib** —

```
@article{Scho90,
  author = "Schoett, Oliver",
  title = {{Behavioural Correctness of Data Representations}},
  journal = "Science of Computer Programming",
  volume = "14",
  year = "1990",
  pages = "43-57",
  abstract =
    "Two methods for proving the correctness of data representation
    are presented which employ a mathematical relation between the
    data values in a representation and those in its abstract
    model. One method reflects the behavioural equaivalence relation
    of abstract data type theory, and the other a new ``behavioural
    inclusion'' notion that formalizes the idea of a ``partial
    representation'' of a data type.

    These correctness concepts and proof methods are strictly more
    general than the conventional ones based on abstraction functions,
    and they are no longer affected by ``implementation bias'' in
    specifications.",
  paper = "Scho90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Scho16,
  author = "Schopenhauer, Arthur",
  title = {{Essays of Schopenhauer: On Reading and Books}},
  link =
  "\url{https://ebooks.adelaide.edu.au/s/schopenhauer/arthur/essays/chapter3.html}",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Scho64,
  author = "Schorre, D.V.",
  title = {{META II: A Syntax-Oriented Compiler Writing Language}},
  booktitle = "19th National Conference of the ACM",
  publisher = "ACM",
  year = "1964",
  abstract =
    "META II is a compiler writing language which consists of syntax
    equations resembling Backus normal form and into which
    instructions to output assembly language commands are
    inserted. Compilers have been written in this language for VALGOL
    I and VALGOL II. The former is a simple algebraic language
    designed for the purpose of illustrating META II The latter
    contains a fairly arge subset of ALGOL 60.

    The method of writing compilers which is given in detail in the
    paper may be explained briefly as follows. Each synta equation is
    translated into a recursive subroutine which tests the input
    string for a particular phrase structure, and deletes it if
    found. Backup is avoided by the extensive use of factoring in the
    syntax equations. For each source language, an interpreter is
    written and programs are compiled into that interpretive language.

    META II is not intended as a standard language which everyone will
    use to write compilers. Rather, it is an example of a simple
    working language which can give one a good start in designing a
    compiler-writing compiler suited to his own needs. Indeed, the
    META II compiler is written in its own language, thus lending
    itself to modification.",
  paper = "Scho64.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Schu72,
  author = "Schubert, Horst",
  title = {{Categories}},
  publisher = "Springer-Verlag",
  year = "1972"
}
```

———-

— **axiom.bib** —

```
@misc{Schu16,
  author = "Schultz, Daniel",
  title = {{Trager's Algorithm for Integration of Algebraic Functions
            Revisited}},
  year = "2016",
  link = "\url{https://sites.psu.edu/dpsmath/files/2016/12/IntegrationOnCurves-2hhuby8.pdf}",
  abstract =
    "Building on work of Risch in the 1980s and Liouville in the
    1840s, Trager published an algorithm for deciding if a given
    algebraic function has an elementary antiderivative. While this
    algorithm is theoretically complete, it is incomplete in the sense
    that assumptions are made about the function to be integrated in
    relation to the defining equation for the algebraic
    irrationality. These assumptions can be justified by a change of
    variables in the defining equation, but this does not lead to the
    most natural algorithm for integration. We fill in the 'gaps' in
    Trager's algorithm and partially implement this algorithm in the
    computer algebra system Mathematica. Various extensions to
    Trager's algorithm are also discussed including a remedy to
    several of the possible points of failure in the algorithm as well
    as the problem of the presence of zero divisors in the algebraic
    function to be integrated.",
  paper = "Schu16.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Scot76,
  author = "Scott, Dana",
  title = {{Data Types as Lattices}},
  journal = "SIAM J. Comput.",
  volume = "5",
  year = "1976",
  pages = "522-587",
  abstract =
    "The meaning of many kinds of expressions in programming languages can
    be taken as elements of certain spaces of 'partial' objects. In this
    report these spaces are modeled in one univeral domain $P_\omega$, the
    set of all subsets of the integers. This domain renders the connection
    of this semantic theory with the ordinary theor of number theoretic
    (especailly general recursive) functions clear and straightforward.",
  paper = "Scot76.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Scot92,
  author = "Scott, Dana and Strachey, Christopher",
```

```
  title = {{Toward a Mathematical Semantics for Computer Languages}},
  year = "1992",
  comment = "Oxford Programming Research Group Monograph PRG-6",
  abstract =
    "Compilers for high-level languages are generally constructed to
    give the complete translation of the programs into machine
    language. As machines merely juggle bit patterns, the concepts of
    the original language may be lost or at least obscured during this
    passage. The purpose of a mathematical semantics is to give a
    correct and meaningful correspondence between programs and
    mathematical entities in a way that is entirely independent of an
    implementation. This plan is illustrated in a very elementary way
    in the introduction. The first section connects the general method
    with the usual idea of state transformations. The next section
    shows why the mathematics of functions has to be modified to
    accommodate recursive commands. Section 3 explains the
    modification. Section 4 introduces the environments for handling
    variables and identifiers and shows how the semantical equations
    define equivalence of programs. Section 5 gives an exposition of
    the new type of mathematical function spaces that are required for
    the semantics of procedures when these are allowed in
    assignments. The conclusion traces some of the background of the
    project and points the way to future work.",
  paper = "Scot92.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{SCSCP10,
  author = "Unknown",
  title = {{Symbolic Computation Software Composability Project and
            its implementations}},
  journal = "ACM Comm. in Computer Algebra",
  volume = "44",
  number = "4",
  year = "2010",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Sedg88,
  author = "Sedgewick, Robert",
  title = {{Algorithms}},
  publisher = "Addison-Wesley",
  year = "1988"
}
```

———-

— **axiom.bib** —

```
@misc{Sels19,
  author = "Selsam, Daniel",
  title = {{CS240H: A Standalone Proof-checker for the Lean Theorem Prover}},
  year = "2019",
  link = "\url{http://www.scs.stanford.edu/16wi-cs240h/projects/selsam.pdf}",
  comment = "\url{https://github.com/dselsam/tc}",
  paper = "Sels19.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Sels16,
  author = "Selsam, Daniel",
  title = {{A Standalone Proof-checker for the Lean Theorem Prover}},
  year = "2016",
  link = "\url{www.scs.stanford.edu/16wi-cs240h/projects/selsam.pdf}",
  comment = "\url{https://github.com/dselsam/tc}",
  paper = "Sels16.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Sels20,
  author = "Selsam, Daniel and Ullrich, Sebastian and
            de Moura, Leonardo",
  title = {{Tabled Typeclass Resolution}},
  year = "2020",
  link = "\url{https://www.dropbox.com/s/5nxklkxvdh7xna9/typeclass.pdf}",
  comment = "preprint",
  abstract =
    "Typeclasses have proven an elegant and effective way of managing
    ad-hoc polymorphism in both programming languages and interactive
    proof assistants. However, the increasingly sophisticated uses of
    typeclasses within proof assistants has exposed two critical
    problems with existing typeclass resolution procedures: the
    {\sl diamond problem} which causes exponential running times in
    both theory and practice, and the {\sl cycle problem}, which
    cuases loops in the presense of cycles and so thwarts many desired
    uses of typeclasses. We present a new typeclass resolution
    procedure, called {\sl tabled typeclass resolution}, that solve
```

```
    these problems. We have implemented our procedure for the upcoming
    version (v4) of the Lean Theorem Prover, and confirm empirically
    that it provides exponential speedups compared to all existing
    procedures in the presense of diamonds. Our procedure is
    sufficiently lightweight that it could easily be implemented in
    other systems. We hope our new procedure facilitates even more
    sophisticated uses of typeclasses in both software development and
    interactive theorem proving.",
  paper = "Sels20.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Shal94,
  author = "Shallit, Jeffrey and Sorenson, Jonathan",
  title = {{Analysis of a Left-Shift Binary GCD Algorithm}},
  journal = "Journal of Symbolic Computation",
  volume = "17",
  number = "6",
  pages = "473-486",
  year = "1994",
  abstract =
    "We introduce a new left-shift binary algorithm, LSBGCD, for
    computing the greatest common divisor of two integers, and we
    provide an analysis of the worst-case behavior of the
    algorithm. The analysis depends on a theorem of Ramharter about
    the extremal behavior of certain continuants.",
  paper = "Shal94.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Shan00,
  author = "Shankar, Natarjan and Owre, Sam",
  title = {{Principles and Pragmatics of Subtyping in PVS}},
  booktitle = "Recent Trends in Algebraic Development Techniques",
  publisher = "Springer",
  pages = "37-52",
  year = "2000"
}
```

———-

— **axiom.bib** —

```
@techreport{Shiv90,
```

```
  author = "Shivers, Olin",
  title = {{Data-Flow Analysis and Type Recovery in Scheme}},
  year = "1990",
  type = "technical report",
  institution = "Carnegie Mellon University",
  number = "CMU-CS-90-115",
  abstract =
    "The lack of explicit type information in Scheme prevents the
    application of many compiler optimizations. Implicit type
    information can oftentimes be recovered by analyzing the flow of
    control through primitive operations and conditionals. Such flow
    analysis, however, is difficult in the presence of higher-order
    functions. This paper presents a technique for recovering type
    information fro mScheme programs using flow analysis. The
    algorithm used for flow analysis is semantically based, and is
    novel in that it correctly handles the ``environment flow''
    problem. Several examples of a working implementation of the
    analysis are presented. The techniques are applicable not only to
    Scheme type recovery, but also to related languages, such as
    Common Lisp and Standard ML, and to related data-flow analyses,
    such as range analysis, future analysis, and copy propagation.",
  paper = "Shiv90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Shou93,
  author = "Shoup, Victor",
  title = {{Factoring Polynomials over Finite Fields:
            Asymptotic Complexity vs Reality*}},
  booktitle = "Proc. IMACS Symposium, Lille, France",
  year = "1993",
  link = "\url{http://www.shoup.net/papers/lille.pdf}",
  abstract =
    "This paper compares the algorithms by Berlekamp, Cantor and
    Zassenhaus, and Gathen and Shoup to conclude that (a) if large
    polynomials are factored the FFT should be used for polynomial
    multiplication and division, (b) Gathen and Shoup should be used if
    the number of irreducible factors of $f$ is small.  (c) if nothing is
    know about the degrees of the factors then Berlekamp's algorithm
    should be used.",
  paper = "Shou93.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Shpa99,
```

```
  author = "Shparlinski, Igor E.",
  title = {{Finite Fields: Theory and Computation}},
  publisher = "Kluwer Academic",
  year = "1999",
  isbn = "0-7923-5662-4",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Sici16,
  author = "Siciliano, Bruno and Oussama, Khatib",
  title = {{Springer Handbook of Robotics}},
  publisher = "Springer",
  year = "2016",
  abstract =
    "Reaching for the human frontier, robotics is vigorously engaged in
    the growing challenges of new emerging domains. Interacting,
    exploring, and working with humans, the new generation of robots will
    increasingly touch people and their lives. The credible prospect of
    practical robots among humans is the result of the scientific
    endeavour of a half a century of robotic developments that established
    robotics as a modern scientific discipline. The Springer Handbook of
    Robotics brings a widespread and well-structured compilation of
    classic and emerging application areas of robotics. From the
    foundations to the social and ethical implications of robotics, the
    handbook provides a comprehensive collection of the accomplishments in
    the field, and constitutes a premise of further advances towards new
    challenges in robotics.  This handbook, edited by two internationally
    renowned scientists with the support of an outstanding team of seven
    part editors and onehundred sixty-four authors, is an authoritative
    reference for robotics researchers, newcomers to the field, and
    scholars from related disciplines such as biomechanics, neurosciences,
    virtual simulation, animation, surgery, and sensor networks among
    others.",
  paper = "Sici16.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Sing85a,
  author = "Singer, M.F. and Davenport, J.H.",
  title = {{Elementary and Liouvillian Solutions of Linear
           Differential Equations}},
  booktitle = "European COnference on Computer Algebra",
  publisher = "Springer",
  pages = "595-596",
  year = "1985",
```

```
  comment = "LNCS 204",
  paper = "Sing85a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Siek06,
  author = "Siek, Jeremy, G. and Taha, Walid",
  title = {{Gradual Typing for Functional Languages}},
  booktitle = "Proc. of Scheme and Functional Programming Workshop",
  year = "2006",
  publisher = "ACM",
  pages = "81-92",
  abstract =
    "Static and dynamic type systems have well-known strengths and
    weaknesses, and each is better suited for different programming
    tasks. There have been many efforts to integrate static and dynamic
    typing and thereby combine the benefits of both typing disciplines in
    the same language. The flexibility of static typing can be improved
    by adding a type Dynamic and a typecase form. The safety and
    performance of dynamic typing can be improved by adding optional type
    annotations or by performing type inference (as in soft
    typing). However, there has been little formal work on type systems
    that allow a programmer-controlled migration between dynamic and
    static typing. Thatte proposed Quasi-Static Typing, but it does not
    statically catch all type errors in completely annotated
    programs. Anderson and Drossopoulou defined a nominal type system
    for an object-oriented language with optional type annotations.
    However, developing a sound, gradual type system for functional
    languages with structural types is an open problem.

    In this paper we present a solution based on the intuition that the
    structure of a type may be partially known/unknown at compile- time
    and the job of the type system is to catch incompatibilities between
    the known parts of types. We define the static and dynamic semantics
    of a $lambda$-calculus with optional type annotations and we prove that
    its type system is sound with respect to the simply-typed
    $lambda$-calculus for fully-annotated terms. We prove that this
    calculus is type safe and that the cost of dynamism is
   'pay-as-you-go'.",
  paper = "Siek06.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Siek89,
  author = "Siekmann, Jorg H.",
```

```
  title = {{Unification Theory}},
  journal = "Journal of Symbolic Computation",
  volume = "7",
  number = "3-4",
  year = "1989",
  pages = "207-274",
  abstract =
    "Most knowledge based systems in artificial intelligence (AI), with a
    commitment to asymbolic representation, support one basic operation:
    ``matching of descriptions''. This operation, called unification in work
    on deduction, is the ``addition-and-multiplication'' of AI-systems and
    is consequently often supported by special purpose hardware or by a
    fast instruction set on most AI-machines. Unification theory provides
    the formal framework for investigations into the properties of this
    operation. This article surveys what is presently known in unification
    theory and records its early history.",
  paper = "Siek89.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Sims71,
  author = "Sims, Charles",
  title = {{Determining the Conjugacy Classes of a Permutation Group}},
  journal = "Computers in Algebra and Number Theory, SIAM-AMS Proc.",
  volume = "4",
  publisher = "American Math. Soc.",
  year = "1991",
  pages = "191--195",
  algebra = "\newline\refto{domain PERMGRP PermutationGroup}"
}
```

———-

— **axiom.bib** —

```
@book{Sips13,
  author = "Sipser, Michael",
  title = {{Introduction to the Theory of Computation}},
  publisher = "Cengage Learning",
  year = "2013",
  isbn = "978-1-133-18779-0",
  link = "\url{https://theswissbay.ch/pdf/Book/Introduction%20to%20the%20theory%20of%20computation_third%20editi
  paper = "Sips13.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Site74,
  author = "Sites, Richard L.",
  title = {{Some Thoughts on Proving Clean Termination of Programs}},
  type = "technical report",
  institution = "Stanford University",
  number = "STAN-CS-74-417",
  year = "1974",
  abstract =
    "Proof of clean termination is a useful sub-goal in the process of
    proving that a program is totally correct. Clean termination means
    that the program terminates (no infinite loops) and that it does
    so normally, without any execution-time semantic errors (integer
    overflow, use of undefined variables, subscript out of range,
    etc.). In contrast to proofs of correctness, proof of clean
    termination requires no extensive annotation of a program by a
    human user, but the proof says nothing about the results
    calculated by the program, just that whatever it does, it
    terminates cleanly. Two example proofs are given, of previously
    published programs: TREESORT3 by Robert Floyd, and SELECT by
    Ronald L. Revest and Robert Floyd.",
  paper = "Site74.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Sjob15,
  author = "Sjoberg, Vilhelm",
  title = {{A Dependently Typed Language with NonTermination}},
  school = "University of Pennsylvania",
  year = "2015",
  abstract =
    "We propose a full-spectrum dependently typed programming
    language, {\sl Zombie}, which supports general recursion
    natively. The Zombie implementation is an elaborating
    typechecker. We prove type safety for a large subset of the Zombie
    core language, including features wuch as computational
    irrelevance, CBV-reduction, and propositional equality with a
    heterogeneous, completely erased elimination form. Zombie does not
    automatically $\beta$-reduce expressions, but instead uses
    congruence closure for proof and type inference. We give a
    specification of a subset of the surface language via a
    bidirectional type system, which works ‘‘up-to-congruence’’, and
    an algorithm for elaborating expressions in this language to an
    explicitly typed core language. We prove that our elaboration
    algorithm is complete with respect to the source type
    system. Zombie also features an optional termination-checker,
    allowing nonterminating programs returning proofs as well as
    external proofs about programs.",
  paper = "Sjob15.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Smit13,
  author = "Smith, Peter",
  title = {{An Introduction to Godel's Theorems}},
  publisher = "Cambridge University Press",
  year = "2013",
  isbn = "978-1-107-02284-3",
  paper = "Smit13.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Smit17,
  author = "Smith, Peter",
  title = {{Teach Yourself Logic 2017: A Study Guide}},
  link =
"\url{https://www.logicmatters.net/resources/pdfs/TeachYourselfLogic2017.pdf}",
  year = "2017",
  paper = "Smit17.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Smol88,
  author = "Smolka, G.",
  title = {{Logic Programming with Polymorphically Order-sorted Types}},
  journal = "Lecture Notes in Computer Science",
  volume = "343",
  pages = "53-70",
  year = "1988"
}
```

———-

— **axiom.bib** —

```
@InCollection{Smol89,
  author = "Smolka, G. and Nutt, W. and Goguen, J. and Meseguer, J.",
  title = {{Order-sorted Equational Computation}},
  booktitle = "Resolution of Equations in Algebra Structures (Vol 2)",
  publisher = "Academic Press",
  pages = "297-367",
```

```
  year = "1989"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Smol89a,
  author = "Smolka, G.",
  title = {{Logic Programming over Polymorphically Order-Sorted Types}},
  school = "Fachbereich Informatik, Universitat Kaiserslautern",
  year = "1989",
  paper = "Smol89a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Smul92,
  author = "Smullyan, Raymond M.",
  title = {{Godel's Incompleteness Theorems}},
  publisher = "Oxford University Press",
  year = "1992",
  isbn = "0-19-504672-2",
  paper = "Smul92.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Smul95,
  author = "Smullyan, Raymond M.",
  title = {{First-Order Logic}},
  publisher = "Dover",
  year = "1995",
  isbn = "978-0-486-68370-6",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Soar96,
  author = "Soare, Robert I.",
  title = {{Computability and Recursion}},
  journal = "Bulletin of Symbolic Logic",
  volume = "2",
```

```
  year = "1996",
  pages = "284-321",
  link = "\url{http://www.people.cs.uchicago.edu/~soare/History/compute.pdf}",
  abstract =
    "We consider the informal concept of 'computability' or 'effective
    calculability' and two of the formalisms commonly used to define
    it, '(Turing) computability' and '(general) recursiveness'. We
    consider their origin, exact technical definition, concepts,
    history, general English meanings, how they became fixed in their
    present roles, how they were first and are now used, their impact
    on nonspecialists, how their use will affect the future content of
    the subject of computability theory, and its connection to other
    related areas.

    After a careful historical and conceptual analysis of
    computability and recursion we make several recommentations in
    section 7 about preserving the {\sl intensional} differences
    between the concepts of 'computability' and
    'recursion'. Specificually we recommend that: the term 'recursive'
    should no longer carry the additional meaning of 'computable' or
    'decidable', functions defined using Turing machines, register
    machines, or their variants should be called 'computable' rather
    than 'recursive', we should distinguish the intensional difference
    between Church's Thesis and Turing's Thesis, and use the latter
    particluarly in dealing with mechanistic questions; the name of
    the subject should be '{\sl Computability Theory}' or simply
    {\sl Computability} rather than 'Recursive Function Theory'",
  paper = "Soar96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inbook{Soar99,
  author = "Soare, Robert I.",
  title = {{The History and Concept of Computability}},
  booktitle = "Handbook of Computability Theory",
  year = "1999",
  link = "\url{http://www.people.cs.uchicago.edu/~soare/History/handbook.pdf}",
  chapter = "unknown",
  pages = "unknown",
  publisher = "Elsevier",
  paper = "Soar99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Soar07,
  author = "Soare, Robert I.",
  title = {{Computability and Incomputability}},
  booktitle = "Proc. Third Conf. on Computability in Europe",
  year = "2007",
  publisher = "Springer-Verlog",
  isbn = "13-978-3-540-73000-2",
  link = "\url{http://www.people.cs.uchicago.edu/~soare/History/siena.pdf}",
  abstract =
    "The conventional wisdom presented in most computability books and
    historical papers is that there were several researchers in the early
    1930s working on various precise definitions and demonstrations of a
    function specified by a finite procedure and that they should all
    share approximately equal credit. This is incorrect. It was Turing
    alone who achieved the characterization, in the opinion of G odel. We
    also explore Turings oracle machine and its analogous properties in
    analysis.",
  paper = "Soar07.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Soze19,
  author = "Sozeau, Matthieu and Boulier, Simon and Forster, Yannick
            and Tabareau, Nicolas and Winterhalter, Theo",
  title = {{Coq Coq Correct!}},
  year = "2019",
  link = "\url{https://www.irif.fr/~sozeau/research/publications/drafts/Coq_Coq_Correct.pdf}",
  abstract =
    "Coq is built around a well-delimited kernel that performs
    typechecking for definitions in a variant of the Calculus of
    Inductive Constructions (CIC). Although the metatheory of CIC is
    very stable and reliable, the correctness of its implementation in
    Coq is less clear. Indeed, implementing an efficient type checker
    for CIC is a rather complex task, and many parts of the code rely
    on implicit invariants which can easily be broken by further
    evolution of the code. Therefore, on average, one critical bug has
    been found every year in Coq. This paper presents the first
    implementation of a type checker for the kernel of Coq, which is
    proven correct in Coq with respect to its formal
    specification. Note that because of Godel's incompleteness
    theorem, there is no hope to prove completely the correctness of
    the specification of Coq inside Coq (in particular strong
    normalisation or canonicity), but it is possible to prove the
    correctness of the implementation assuming the correctness of the
    specification. Our work is based on the METACOQ project [Anand et
    al. 2018; Sozeau et al. 2019] which provides metaprogramming
    facilities to work with terms and declarations at the level of
    this kernel. Our type checker is based on the specification of the
    typing relation of the Polymorphic Cumulative Calculus of
```

```
    Inductive Constructions (PCUIC) at the basis of Coq and the
    verification of a relatively efficient and sound type-checker for
    it. In addition to the kernel implementation, an essential feature
    of Coq is so-called {\sl extraction} [Lebouzey 2004]; the
    production of executable code in functional languages from Coq
    definitions. We present a verified version of this subtle
    type-and-proof erasure step, therefore enabling the verified
    extraction of a safe type-checker for Coq.",
  paper = "Soze19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Sriv10,
  author = "Srivastava, Saurabh and Gulwani, Sumit and Foster, Jeffrey S.",
  title = {{From Program Verification to Program Synthesis}},
  booktitle = "37th Symp. on Principles of Programming Languages",
  publisher = "ACM",
  pages = "313-326",
  year = "2010",
  abstract =
    "This paper describes a novel technique for the synthesis of
    imperative programs. Automated program synthesis has the potential
    to make programming and the design of systems easier by allowing
    programs to be specified at a higher-level than executable
    code. In our approach, which we call proof-theoretic synthesis,
    the user provides an input-output functional specification, a
    description of the atomic operations in the progamming language,
    and a specification of the synthesized program's looping
    structure, allowed stack space, and bound on usage of certain
    operations. Our technique synthesizes a program, if there exists
    one, that meets the input-output specification and uses only the
    given resources.

    The insight behind our approach is to interpret program synthesis
    as generalized program verification, which allows us to bring
    verification tools and techniques to program synthesis. Our
    synthesis algorithm works by creating a program with unknown
    statements, guards, inductive invariants, and ranking
    functions. It then generates constraints that relate the unknowns
    and enforces three kinds of requirements: partial correctness,
    loop termination, and well-formedness conditions on program
    guards. We formalize the requirements that program verification
    tools must meet to solve these constraints and use tools from
    prior work as our synthesizers.

    We demonstrate the feasibility of the proposed approach by
    synthesizing programs in three different domains: arithmetic,
    sorting, and dynamic programming. Using verification tools that we
    previously built in the VS3 project we are able to synthesize
```

```
    programs for complicated arithmetic algorithms including
    Strassen's matrix multiplication and Bresenham's line drawing;
    several sorting algorithms; and several dynamic programming
    algorithms. For these programs, the median time for synthesis is
    14 seconds, and the ratio of synthesis to verification time ranges
    between 1x to 92x (with a median of 7x), illustrating the potential
    of the approach.",
  paper = "Sriv10.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Stac17,
  author = "StackExchange",
  title = {{How do Gap generate the elements in permutation groups}},
  year = "2017",
  link = "\url{http://math.stackexchange.com/questions/1705277/how-do-gap-generate-the-elements-in-permutation-g
}
```

———-

— **axiom.bib** —

```
@inproceedings{Stan88,
  author = "Stansifer, R.",
  title = {{Type Inference with Subtypes}},
  booktitle = "POPL 88",
  pages = "88-97",
  year = "1988",
  abstract =
    "We give an algorithm for type inference in a language with functions,
    records, and variant records. A similar language was studied by
    Cardelli who gave a type checking algorithm. This language is
    interesting because it captures aspects of object-oriented programming
    using subtype polymorphism. We give a type system for deriving types
    of expressions in the language and prove the type inference algorithm
    is sound, i.e., it returns a type derivable from the proof system. We
    also prove that the type the algorithm finds is a ``principal'' type,
    i.e., one which characterizes all others. The approach taken here is
    due to Milner for universal polymorphism. The result is a synthesis of
    subtype polymorphism and universal polymorphism.",
  paper = "Stan88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Stee10,
```

```
  author = "Steel, Allan K.",
  title = {{Computing with Agebraically Closed Fields}},
  journal = "J. of Symbolic Computation",
  volume = "45",
  number = "3",
  year = "2010",
  pages = "342-372",
  abstract =
    "A practical computational system is described for computing with
    an algebraic closure of a field. The system avoids factorization
    of polynomials over extension fields, but gives the illusion of a
    genuine field to the user. All roots of an arbitrary polynomial
    defined over such an algebraically closed field can be constructed
    and are easily distinguished within the system. The difficult case
    of inseparable extensions of function fields of positive
    characteristic is also handled properly by the system. A technique
    of modular evaluation into a finite field critically ensures that
    a unique genuine field is simulated by the system but also
    provides fast optimizations for some fundamental operations. Fast
    matrix techniques are also used for several non-trivial
    operations. The system has been successfully implemented within
    the Magma Computer Algebra System, and several examples are
    presented, using this implementation.",
  paper = "Stee10.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Stee17,
  author = "Steele, Guy",
  title = {{It's Time for a New Old Language}},
  year = "2017",
  link = "\url{https://www.youtube.com/watch?v=dCuZkaaou0Q}",
  paper = "Stee17.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Stee87,
  author = "Steenkiste, Peter and Hennessy, John",
  title = {{Tags and Type Checking in LISP: Hardware and Software Approaches}},
  journal = "ACM SIGPLAN Notices",
  volume = "22",
  number = "10",
  pages = "50-59",
  year = "1987",
  abstract =
```

```
    "One of the major factors that distinguishes LISP from many other
    languages (Pascal, C, Fortran, etc.) is the need for run-time type
    checking. Run-time type checking is implemented by adding to each
    data object a tag that encodes type information. Tags must be
    compared for type compatibility, removed when using the data, and
    inserted when new data items are created. This tag manipulation,
    together with other work related to dynamic type checking and
    generic operations, constitutes a significant component of the
    execution time of LISP programs. This has led both to the
    development of LISP machines that support tag checking in hardware
    and to the avoidance of type checking by users running on stock
    hardware. To understand the role and necessity of special-purpose
    hardware for tag handling, we first measure the cost of type
    checking operations for a group of LISP programs. We then examine
    hardware and software implementations of tag operations and
    estimate the cost of tag handling with the different tag
    implementation schemes. The data shows that minimal levels of
    support provide most of the benefits, and that tag operations can
    be relatively inexpensive, even when no special hardware support
    is present.",
  paper = "Stee87.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Step09,
  author = "Stepanov, Alexander and McJones, Paul",
  title = {{Elements of Programming}},
  publisher = "Semigroup Press",
  year = "2009",
  isbn = "978-0-578-22214-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@techreport{Stic89,
  author = "Stickel, Mark E.",
  title = {{A Prolog Technology Theorem Prover: A New Exposition and
            Implemention in Prolog}},
  type = "technical note",
  institution = "SRI International",
  number = "464",
  year = "1989",
  link = "\url{www.ai.sri.com/~stickel/pttp.html}",
  abstract =
    "A Prolog technology theorem prover (PTTP) is an extension of
```

```
    Prolog that is complete for the full first-order predicate
    calculus. It differs from Prolog in its use of unification with
    the occurs check for soundness, depth-first iterative-deepening
    search instead of unbounded depth-first search to make the search
    strategy complete, and the model elimination reduction rule that
    is added to Prolog inferences to make the inference system
    complete. This paper describes a new Prolog-based implementation
    of PTTP. It uses three compile-time transformations to translate
    formulas into Prolog clauses that directly execute, with the
    support of a few run-time predicates, the model elimination
    procedure with depth-first iterative-deepening search and
    unification with the occurs check. Its high performance exceeds
    that of Prolog-based PTTP interpreters, and it is more concise and
    readable than the earlier Lisp-based compiler, which makes it
    superior for expository purposes. Examples of inputs and outputs
    of the compile-time transformations provide an easy and quite
    precise way to explain how PTTP works, This Prolog-based version
    makes it easier to incorporate PTTP theorem-proving ideas into
    Prolog programs. Some suggestions are made on extensions to Prolog
    that could be used to improve PTTP's performance.",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Stoj86,
  author = "Stojanovski, Jordan",
  title = {{A Note on Implementing Prolog in Lisp}},
  journal = "Information Processing Letters",
  volume = "23",
  pages = "261-264",
  year = "1986",
  paper = "Stoj86.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Stra00,
  author = "Strachey, Christopher",
  title = {{Fundamental Concepts in Programming Languages}},
  journal = "Higher-Order and Symbolic Computation",
  volume = "13",
  number = "1-2",
  pages = "11-49",
  year = "2000",
  abstract =
    "This paper forms the substance of a course of lectures given at the
```

```
    International Summer School in Computer Programming at Copenhagen in
    August, 1967. The lectures were originally given from notes and the
    paper was written after the course was finished. In spite of this, and
    only partly because of the shortage of time, the paper still retains
    many of the shortcomings of a lecture course. The chief of these are
    an uncertainty of aimit is never quite clear what sort of audience
    there will be for such lecturesand an associated switching from
    formal to informal modes of presentation which may well be less
    acceptable in print than it is natural in the lecture room. For these
    (and other) faults, I apologise to the reader.

    There are numerous references throughout the course to CPL [13]. This
    is a programming language which has been under development since 1962
    at Cambridge and London and Oxford. It has served as a vehicle for
    research into both programming languages and the design of
    compilers. Partial implementations exist at Cambridge and London. The
    language is still evolving so that there is no definitive manual
    available yet. We hope to reach another resting point in its evolution
    quite soon and to produce a compiler and reference manuals for this
    version. The compiler will probably be written in such a way that it
    is relatively easyto transfer it to another machine, and in the first
    instance we hope to establish it on three or four machines more or
    less at the same time.

    The lack of a precise formulation for CPL should not cause much
    difficulty in this course, as we are primarily concerned with the
    ideas and concepts involved rather than with their precise
    representation in a programming language.",
  paper = "Stra00.pdf",
  keywords = "printed, DONE"
}
```

———-

— axiom.bib —

```
@article{Stra00a,
  author = "Strachey, Christopher and Wadsworth, Christopher",
  title = {{Continuations: A Mathematical Semantics for Handling Full Jumps}},
  journal = "Higher-Order and Symbolic Computation",
  volume = "13",
  pages = "135-152",
  year = "2000",
  abstract =
    "This paper describes a method of giving the mathematical
    semantics of programming languages which include the most general
    form of jumps.",
  paper = "Stra00a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Stra08,
  author = "Stratford, Jonathan and Davenport, James H.",
  title = {{Unit Knowledge Management}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
  abstract =
    "In 9, various observations on the handling of (physical) units in
    OpenMath were made. In this paper, we update those observations,
    and make some comments based on a working unit converter that,
    because of its OpenMath-based design, is modular, extensible, and
    reflective. We also note that some of the issues in an effective
    converter, such as the rules governing abbreviations, being more
    linguistic than mathematical, do not lend themselves to easy
    expression in OpenMath.",
  paper = "Stra08.pdf"
}
```

———-

— **axiom.bib** —

```
@inbook{Stre98,
  author = "Strecker, M. and Luther, M. and von Henke, F.",
  title = {{Interactive and Automated Proof Construction in Type Theory}},
  publisher = "Springer",
  chapter = "3",
  pages = "73-96",
  isbn = "978-94-017-0435-9",
  year = "1998",
  abstract =
    "This chapter gives a survey of TYPELAB, a specification and
    verification environment that integrates interactive proof
    development and automated proof search. TYPELAB is based on a
    constructive type theory, the Calculus of Constructions, which can
    be understood as a combination of a typed $\lambda$-calculus and
    an expressive higher-order logic. Distinctive features of the type
    system are dependent function types ($\Pi$ types) for modeling
    polymorphism and dependent record types ($\Sigma$ types) for
    encoding specifications and mathematical theories.",
  paper = "Stre98.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Stre99,
  author = "Strecker, Martin",
```

```
  title = {{Construction and Deduction in Type Theories}},
  school = "Unversitat Ulm",
  year = "1999",
  abstract =
    "This dissertation is concerned with interactive proof
    construction and automated proof search in type theories, in
    particular the Calculus of Constructions and its subsystems.

    Type theories can be conceived as expressive logics which combine
    a functiona programming language, strong typing and a higher-order
    logic. They are therefore a suitable formalism for specification
    and verification systems. However, due to their expressiveness, it
    is difficult to provide appropriate deductive support for type
    theories. This dissertation first examines general methods for
    proof construction in type theories and then explores how these
    methods can be refined to yield proof search procedures for
    specialized fragments of the language.

    Proof development in type theories usually requires the
    construction of a term having a given type in a given context. For
    the term to be constructed, a {\sl metavariable} is introduced
    which is successively instantiated in the course of the proof. A
    naive use of metavariables leads to problems, such as
    non-commutativity of reduction and instantiation and the
    generation of ill-typed terms during reduction. For solving these
    problems, a calculus with {\sl explicit substitutions} is
    introduced, and it is shown that this calculus preserves
    properties such as strong normalisation and decidability of typing.

    In order to obtain a calculus appropriate for proof search, the
    usual natural deduction presentation of type theories is replaced
    by a {\sl sequent style presentation}. It is shown that the
    calculus thus obtained is correct with respect to the original
    calculus. Completeness (proved with a cut-elimination argument) is
    shown for all predicative fragments of the lambda cube.

    This dissertation concludes with a discussion of some techniques
    that make proof search practically applicable, such as unification
    and pruning of the proof search space by exploiting
    impermutabilities of the sequent calculus.",
  paper = "Stre99.pdf"
}
```

———-

— axiom.bib —

```
@book{Stre91,
  author = "Streicher, Thomas",
  title = {{Semantics of Type Theory}},
  year = "1991",
  publisher = "Springer",
  isbn = "978-1-4612-6757-7"
```

```
}
```

———-

— **axiom.bib** —

```
@inproceedings{Stor97,
  author = "Storjohann, Arne",
  title = {{A Solution to the extended GCD problem with applications}},
  booktitle = "ISSAC '97",
  publisher = "ACM",
  year = "1997",
  paper = "Stor97.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Stro95,
  author = "Stroustrup, Bjarne",
  title = {{The C++ Programming Language (2nd Edition)}},
  publisher = "Addison-Wesley",
  year = "1995",
  isbn = "0-201-53992-6"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Stru10,
  author = "Strub, Pierre-Yves",
  title = {{Coq Modulo Theory}},
  booktitle = "19th Annual Conf. on Computer Science Logic",
  publisher = "Springer",
  pages = "549-643",
  year = "2010"
}
```

———-

— **axiom.bib** —

```
@misc{Stum18,
  author = "Stump, Aaron",
  title = {{Syntax and Semantics of Cedille}},
  year = "2018",
  comment = "arXiv:1806.04709v1",
  paper = "Stum18.pdf",
```

```
  keywords = "printed"
}
```

— **axiom.bib** —

```
@article{Stur1829,
  author = "Sturm, Jacques Charles Francois",
  title = {{M\'emoire sur la r\'solution des \'equations num\'eriques}},
  journal = {Bulletin des Sciences de F\'erussac},
  year = "1829",
  volume = "11",
  pages = "419-425",
  link = "\url{https://en.wikipedia.org/wiki/Sturm%27s\_theorem}",
  abstract =
    "Let $p_0,\ldots,p_m$ be the Sturm chain of a square free polynomial $p$,
    and let $\sigma{\eta}$ denote the number of sign changes (ignoring zeros)
    in the sequence $p_0(\eta),p_1(\eta),p_2(\eta),\ldots,p_m(\eta)$.
    Sturms' theorem states that for two real numbers $a < b$, the number of
    distinct roots of $p$ in the half-open interval $(a,b]$ is
    $\sigma(a)-\sigma(b)$.

    A Sturm chain is a finite sequence of polynomials $p_0,p_1,\ldots,p_m$
    of decreasing degree with these following properties:
    \begin{itemize}
    \item $p_0=p$ is square free (no square factors, i.e. no repeated roots)
    \item if $p(\eta)=0$, then $sign(p_1(\eta))=sign(p^\prime(\eta))$
    \item if $p_i(\eta)=0$ for $0<i<m$ then
    $sign(p_{i-1}(\eta))=-sign(p_{i+1}(\eta))$
    \item $p_m$ does not change its sign
    \end{itemize}"
}
```

— **axiom.bib** —

```
@article{Stur02,
  author = "Sturm, Thomas",
  title = {{Integration of Quantifier Elimination with Constraint
            Logic Programming}},
  journal = "LNCS",
  volume = "2385",
  year = "2002",
  abstract =
    "We examine the potential of an extension of constraint logic
    programming, where the admissible constraints are arbitrary
    first-order formulas over some domain. Constraint solving is
    realized by effective quantifier elimination. The arithmetic is
    always exact. We descrbe the conceptual advantages of our approach
    and the capabilities of the current implementation
```

```
    CLP(RL). Supported domains are $\mathbb{R}$, $\mathbb{C}$, and
    $\mathbb{Q}_p$. For our discussion here we restrict to $\mathbb{R}$.",
  paper = "Stur02.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Suss16,
  author = "Sussman, Gerald Jay",
  title = {{The Power of Generic Operations}},
  year = "2016",
  link = "\url{https://vimeo.com/151465912}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Swie19,
  author = "Swierstra, Wouter and Baanen, Tim",
  title = {{A Predicate Transformer Semantics for Effects}},
  booktitle = "Inter. Conf. on Functional Programming",
  publisher = "ACM",
  year = "2019",
  abstract =
    "Reasoning about programs that use effects can be much harder than
    reasoning about their pure counterparts. This paper presents a
    predicate transformer semantics for a variety of effects,
    including exceptions, state, non-determinism, and general
    recursion. The predicate transformer semantics gives rise to a
    refinement relation that can be used to relate a program to its
    specification, or even calculate effectful programs that are
    correct by construction.",
  paper = "Swie19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Swor11,
  author = "Swords, Sol and Davis, Jared",
  title = {{Bit-oBlasting ACL2 Theorems}},
  journal = "EPTCS",
  volume = "70",
  pages = "84-102",
  year = "2011",
  abstract =
```

```
    "Interactive theorem proving requires a lot of human
    guidance. Proving a property involes (1) figuring out why it
    holds, the (2) coaxing the theorem prover into believing it. Both
    steps can take a long time. We explain how to use GL, a framework
    for proving finite ACL2 theorems with BDD- and SAT-based
    reasoning. This approach makes it unnecessary to deeply understand
    why a property is true, and automates the process of admitting it
    as a theorem. We use GL at Centaur Technology to verify execution
    units for x86 Integer, MMX, SSE, and floating-point arithmetic.",
  paper = "Swor11.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Szab82,
  author = "Szabo, P.",
  title = {{Unifikationstheorie erster Ordnung}},
  school = {Fakult\"at f\"ur Informatik, Universit\"at Karlsruhe},
  year = "1982"
}
```

———-

— **axiom.bib** —

```
@misc{Szed14,
  author = "Szegedy, Mario",
  title = {{Three Negative Results}},
  year = "2014",
  comment = "Microsoft QuArc Workshop 2104",
  link = "\url{https://www.youtube.com/watch?v=vIB478tYPJE}"
}
```

———-

1.2.20 T

— **axiom.bib** —

```
@article{Tait86,
  author = "Tait, William W.",
  title = {{Truth and Proof: The Platonism of Mathematics}},
  link = "\url{logic.harvard.edu/EFI_Tait_PlatonisminMathematics.pdf}",
  journal = "Synthese",
  volume = "69",
  pages = "341-370",
  year = "1986",
  paper = "Tait86.pdf",
```

```
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Talc90,
  author = "Talcott, Carolyn",
  title = {{A Theory for Program and Data Type Specifications}},
  booktitle = "DISCO 1990",
  publisher = "Springer",
  year = "1990",
  pages = "91-100",
  paper = "Talc90.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Tanx18,
  author = "Tan, Yong Kiam and Myreen, Magnus O. and Kumar, Ramana and
            Fox, Anthony and Owens, Scott",
  title = {{The Verified CakeML Compiler Backend}},
  journal = "Functional Programming",
  year = "2018",
  link = "\url{https://www.cs.cmu.edu/~yongkiat/files/cakeml-jfp.pdf}",
  abstract =
    "The CakeML compiler is, to the best of our knowledge, the most
    realistic verified compiler for a functional programming language to
    date.  The architecture of the compiler, a sequence of intermediate
    languages through which high-level features are compiled away
    incrementally, enables verification of each compilation pass at an
    appropriate level of semantic detail. Parts of the compilers
    implementation resemble mainstream (unverified) compilers for strict
    functional languages, and it supports several important features and
    optimisations. These include efficient curried multi-argument
    functions, configurable data representations, efficient exceptions,
    register allocation, and more. The compiler produces machine code for
    five architectures: x86-64, ARMv6, ARMv8, MIPS-64, and RISC-V. The
    generated machine code contains the verified runtime system which
    includes a verified generational copying garbage collector and a
    verified arbitrary precision arithmetic (bignum) library.  In this
    paper we present the overall design of the compiler backend, including
    its 12 intermediate languages.  We explain how the semantics and
    proofs fit together, and provide detail on how the compiler has been
    bootstrapped inside the logic of a theorem prover. The entire
    development has been carried out within the HOL4 theorem prover.",
  paper = "Tanx18.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Tani95,
  author = "Tanimoto, Steven L.",
  title = {{The Elements of Artificial Intelligence Using Common
           Lisp}},
  publisher = "Computer Science Press",
  year = "1995",
  isbn = "0-7167-8269-3",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Tank13,
  author = "Tankink, Carst and Kaliszyk, Cezary and Urban, Josef and
            Geuvers, Herman",
  title = {{Formal Mathematics on Display: A Wiki for Flyspeck}},
  journal = "LNCS",
  volume = "7961",
  year = "2013",
  abstract =
    "The AGORA system is a prototype ``Wiki for Formal Mathematics'',
    with an aim to support developing and documenting large
    formalizations of mathematics in a proof assistant. The functions
    implemented in AGORA include in-browser editing, strong AI/ATP
    proof advice, verification, and HTML rendering. The HTML rendering
    contains hyperlinks and provides on-demand explanation of the
    proof state for each proof step. In the present paper we show the
    prototype Flyspeck Wiki as an instance of AGORA for HOL Light
    formalizations. The wiki can be used for formalizations of
    mathematics and for writing informal wiki pages about
    mathematics. Such informal pages may contain islands of formal
    text, which is used here for providing an initial cross-linking
    between Hales's informal Flyspeck book, and the formal Flyspeck
    development.

    The AGORA platform intends to address distributed wiki-style
    collaboration on large formalization projects, in particular both
    the aspect of immediate editing, verification and rendering of
    formal code, and the aspect of gradual and mutual refactoring and
    correspondence of the initial informal text and its
    formalization. Here, we highlight these features with the Flyspeck
    Wiki.",
  paper = "Tank13.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Temp92,
  author = "Temperini, M.",
  title = {{Design and Implementation Methodologies for Symbolic
           Computation Systems}},
  year = "1992",
  comment = "Preprint"
}
```

———-

— **axiom.bib** —

```
@article{Tann93,
  author = "Breazu-Tannen, Val and Coquand, Thierry and Gunter, Carl A.
           and Scedrov, Andre",
  title = {{Inheritance as Implicit Coercion}},
  journal = "Information and Computation",
  volume = "93",
  pages = "172-221",
  year = "1991",
  abstract =
    "We present a method for providing semantic interpretations for
    languages with a type system featuring inheritance polymorphism. Our
    approach is illustrated on an extension of the language Fun of
    Cardelli and Wegner, which we interpret via a translation into an
    extended polymorphic lambda calculus. Our goal is to interpret
    inheritances in Fun via coercion functions which are definable in the
    target of the translation. Existing techniques in the theory of
    semantic domains can be then used to interpret the extended
    polymorphic lambda calculus, thus providing many models for the
    original language. This technique makes it possible to model a rich
    type discipline which includes parametric polymorphism and recursive
    types as well as inheritance. A central difficulty in providing
    interpretations for explit type disciplines featuring inheritance in
    the sense discussed in this paper arises from the fact that programs
    can type-check in more than one way. Since interpretations follow the
    type-checking derivations, coherence theorems are required: that is,
    one must prove that the meaning of a program does not depend on the
    way it was type-checked. Proofs of such theorems for our proposed
    interpretation are the basic technical results of this paper.
    Interestingly, proving coherence in the presence of recursive
    types, variants, and abstract types forced us to reexamine fundamental
    equational properties that arise in proof theory (in the form of
    commutative reductions) and domain theory (in the form of strict vs
    non-strict functions).",
  paper = "Tann93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Tara16,
  author = "Tarau, Paul",
  title = {{A Hitchhiker's Guide to Reinventing a Prolog Machine}},
  year = "2016",
  abstract =
    "We take a fresh, ``clean-room'' look at implementing Prolog by
    deriving its translation to an executable representation and its
    execution algorithm from a simple Horn Clause meta-interpreter.
    The resulting design has some interesting properties:
    \begin{itemize}
    \item the heap representation of terms and the abstract machine
    instruction encodings are the same.
    \item no dedicated code area is used as the code is placed
    directly on the heap.
    \item unification and indexing operations are orthogonal
    \item filtering of matching clauses happens without building new
    structures on the heap
    \item variables in function and predicate symbol positions are
    handled with no performance penalty
    \item a simple English-like syntax is used as an intermediate
    representation for clauses and goals
    \item the same English-like syntax can be used by programmers
    directly as an alternative to classic Prolog syntax
    \item solutions of (multiple) logic engines are exposed as answer
    streams that can be combined through typical functional
    programming patterns
    \item performance of a basic interpreter implemeting our design is
    within a factor of 2 of a highly optimized WAM-based system
    \end{itemize}

    To help placing our design on the fairly rich map of Prolog
    systems, we discuss similarities to existing Prolog abstract
    machines, with emphasis on separating necessary commonalities from
    arbitrary implementation choices.",
  paper = "Tara16.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Tard03,
  author = "Tarditi, David and Morrisett, Greg and Cheng, Perry and
            Stone, Chris and Harper, Roboert and Lee, Peter",
  title = {{TIL: A Type-Directed, Optimizing Compiler for ML}},
  booktitle = "20 Years of Prog. Lang. Design and Implementation",
  year = "2003",
  publisher = "ACM",
  isbn = "1-58113-623-4",
```

```
  paper = "Tard03.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Tarvxx,
  author = "Tarver, Mark",
  title = {{A Language for Implementing Arbitrary Logics}},
  year = "unknown",
  abstract =
    "SEQUEL is a new-generation functional programming language, which
    allows the specification of types in a notation based on the
    sequent calculus. The sequent calculus notation suffices for the
    construction of types for type checking and for the specification
    of arbitrary logics. Compilation techniques derived from both
    functional and logic programming are used to derive
    high-performance ATPs from these specifications.",
  paper = "Tarvxx.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Tayl16,
  author = "Taylor, Sophie",
  title = {{A working (class) Introduction to Homotopy Type Theory}},
  year = "2016",
  link = "\url{https://www.youtube.com/watch?v=xv6SwPn1dlc}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@article{Tenn79,
  author = "Tennant, Neil",
  title = {{Entailment and Proofs}},
  journal = "Proc. of the Aristotelian Society",
  volume = "79",
  pages = "167-189",
  year = "1979",
  link = "\url{https://cpb-us-w2.wpmucdn.com/u.osu.edu/dist/a/4597/files/2014/07/tennant_pas1979-1rrsm6l.pdf}",
  paper = "Tenn79.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Tenn90,
  author = "Tennant, Neil",
  title = {{Natural Logic}},
  publisher = "Edinburgh University Press",
  isbn = "0-85224-579-3",
  year = "1990",
  paper = "Tenn90.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Tenn76,
  author = "Tennent, R.D.",
  title = {{The Denotational Semantics of Programming Languages}},
  journal = "Communications of the ACM",
  volume = "19",
  number = "8",
  pages = "437-453",
  year = "1976",
  abstract =
    "This paper is a tutorial introduction to the theory of
    programming language semantics developed by D. Scott and
    C. Strachey. The application of the theory to formal language
    specification is demonstrated and other applications are
    surveyed. The first language considered, LOOP, is very elementary
    and its definition merely introduces the notion and methodology of
    the approach. Then the semantic concepts of environments, stores,
    and continuations are introduced to model classes of programming
    language features and the underlying mathematical theory of
    computation due to Scott is motivated and outlined. Finally, the
    paper presents a formal definition of the language GEDANKEN.",
  paper = "Tenn76.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{That80,
  author = "Thatcher, James W. and Wagner, Eric G. and Wright, Jesse B.",
  title = {{More on Advice on Structuring Compilers and Proving Them
            Correct}},
  journal = "LNCS",
```

```
  volume = "54",
  pages = "165-188",
  year = "1980",
  paper = "That80.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{That91,
  author = "Thatte, Satish R.",
  title = {{Coercive Type Isomorphism}},
  journal = "LNCS",
  volume = "523",
  year = "1991",
  pages = "29-49",
  abstract =
    "There is a variety of situations in programming in which it is useful
    to think of two distinct types as representations of the same abstract
    structure.  However, language features which allow such relations to
    be effectively expressed at an abstract level are lacking.  We propose
    a generalization of ML-style type inference to deal effectively with
    this problem.  Under the generalization, the (normally free) algebra
    of type expressions is subjected to an equational theory generated by
    a finite set of user-specified equations that express
    interconvertibility relations between objects of ``equivalent'' types.
    Each type equation is accompanied by a pair of conversion functions
    that are (at least partial) inverses.  We show that so long as the
    equational theory satisfies a reasonably permissive syntactic
    constraint, the resulting type system admits a complete type
    inference algorithm that produces unique principal types.  The main
    innovation required in type inference is the replacement of ordinary
    free unification by unification in the user-specified equational
    theory.  The syntactic constraint ensures that the latter is unitary,
    i.e., yields unique most general unifiers.  The proposed constraint is
    of independent interest as the first known syntactic
    characterization for a class of unitary theories.  Some of the
    applicatloils of the system are similar to those of Wadler's views
    [Wad87]. However, our system is considerably more general, and more
    orthogonal to the underlying language.",
  paper = "That91.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Thie17,
  author = "Thiemann, Rene and Yamada, Akihisa",
  title = {{Polynomial Factorization: Proof Outline}},
```

```
  journal = "Archive of Formal Proofs",
  year = "2017",
  link = "\url{https://www.isa-afp.org/browser_info/current/AFP/Polynomial_Factorization/outline.pdf}",
  abstract =
    "Based on existing libraries for polynomial interpolation and
    matrices, we formalized several factorization algorihms for
    polynomials, including Kronecker's algorithm for intgeer polynomials,
    Yun's square-free factorization algorithm for field polynomials, and a
    factorization algorithm which delivers root-free polynomials.

    As side products, we developed division algorithms for polynomials
    over integral domains, as well as primality-testing and prime
    factorization algorithms for integers.",
  paper = "Thie17.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Thie17a,
  author = "Thiemann, Rene and Yamada, Akihisa",
  title = {{Polynomial Factorization: Proof Document}},
  journal = "Archive of Formal Proofs",
  year = "2017",
  link = "\url{https://www.isa-afp.org/browser_info/current/AFP/Polynomial_Factorization/document.pdf}",
  abstract =
    "Based on existing libraries for polynomial interpolation and
    matrices, we formalized several factorization algorihms for
    polynomials, including Kronecker's algorithm for intgeer polynomials,
    Yun's square-free factorization algorithm for field polynomials, and a
    factorization algorithm which delivers root-free polynomials.

    As side products, we developed division algorithms for polynomials
    over integral domains, as well as primality-testing and prime
    factorization algorithms for integers.",
  paper = "Thie17a.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Thom99,
  author = "Thompson, Simon",
  title = {{Type Theory and Functional Programming}},
  year = "1999",
  publisher = "Addison-Wesley",
  abstract =
    "This book explores the role of Martin-Lof's constructive type
    theory in computer programming. Th emain focus of the book is how
```

```
    the theory can be successfully applied in practice. Introductory
    sections provide the necessary background in logic, lambda
    calculus and constructive mathematics, and exercises and chapter
    summaries are included to reinforce understanding",
  paper = "Thom99.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Thomxx,
  author = "Thompson, Simon and Timochouk, Leonid",
  title = {{The Aldor\-\- language}},
  year = "unknown",
  link = "\url{fricas-wiki.math.uni.wroc.pl/public/refs/thompson-aldor.pdf}",
  abstract =
    "This paper introduces the Aldor\-\- language, which is a
    functional programming language with dependent types and a
    powerful, type-based, overloading mechanism. The language is built
    on a subset of Aldor, the 'library compiler' language for the
    Axiom computer algebra system. Aldor\-\- is designed with the
    intention of incorporating logical reasoning into computer algebra
    computations.

    The paper contains a formal account of the semantics and type
    system of Aldor\-\-; a general discussion of overloading and how
    the overloading in Aldor\-\- fits into the general scheme;
    examples of logic within Aldor\-\- and notes on the implementation
    of the system.",
  paper = "Thomxx.pdf",
  keywords = "printed, axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Thor12,
  author = "Thornton, Jacob",
  title = {{What is Open Source and Why Do I Feel So Guilty}},
  year = "2012",
  link = "\url{https://www.youtube.com/watch?v=UIDb6VBO9os}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Tima18,
  author = "Timany, Amin and Sozeau, Mattieu",
```

```
  title = {{Cumulative Inductive Types in Coq}},
  booktitle = "3rd Conf. on Formal Structures for Computation and
               Deduction",
  publisher = "HAL",
  link = "\url{https://hal.inria.fr/hal-01952037/document}",
  abstract =
    "In order to avoid well-known paradoxes associated with
    self-referential definitions, high-order dependent type theories
    stratify the theory using a countably infinite hierarchy of
    universes (also known as sorts), $Type_0 : Type_1,\ldots$ Such
    type systems are called cumulative if for any type $A$ we have
    that $A:Type_i$ implies $A:Type_{i+1}$. The Predicative Calculus
    of Inductive Constructions (pCIC) which forms the basis of the Coq
    proof assistant, is one such system. In this paper we present the
    Predicative Calculus of Cumulative Inductive Constructions
    (pCUIC) which extends the cumulativity relation to inductive
    types. We discuss cumulative inductive types as present in Coq 8.7
    and their application to formalization and definitional
    translations.",
  paper = "Tima18.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Tiur90,
  author = "Tiuryn, J.",
  title = {{Type Inference Problems -- A Survey}},
  journal = "LNCS",
  volume = "452",
  pages = "105-120",
  year = "1990",
  paper = "Tiur90.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Tiur92,
  author = "Tiuryn, J.",
  title = {{Subtype Inequalities}},
  booktitle = "Proc. Logic in Computer Science 92",
  year = "1992",
  pages = "308-315",
  abstract =
    "In this paper we study the complexity of the satisfiability problem
    for subtype inequalities in simple types. The naive algorithm which
    solves this problem runs in non-deterministic exponential time for
    every pre-defined poset of atomic subtypings. In this paper we show
    that over certain finite posets of atomic subtypings the
```

```
    satisfiability problem for subtype inequalities is PSPACE-hard.  On
    the other hand we prove that if the poset of atomic subtypings is a
    disjoint union of lattices, then the satisfiability problem for
    subtype inequalities is solvable in PTIME. This result covers the
    important special case of the unification problem which can be
    obtained when the atomic subtype relation is equality (in this case
    the poset is a union of one-element lattices).",
  paper = "Tiur92.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Tour90,
  author = "Touretzky, David S.",
  title = {{Common Lisp: A Gentle Introduction to Symbolic Computation}},
  year = "1990",
  publisher = "Benjamin/Cummings Publishing Company",
  paper = "Tour90.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Toft88,
  author = "Tofte, Mads",
  title = {{Operational Semantics and Polymorphic Type Inference}},
  school = "Univ. of Edinburgh",
  year = "1988",
  abstract =
    "Three languages with polymorphic type disciplines are discussed,
    namely the $\lambda$-calculus with Milner's polymorphic type
    discipline; a language with imperative features (polymorphic
    references); and a skeletal module language with structures,
    signatures and functors. In each of the two first cases we show
    that the type inference system is consistent with an operational
    dynamic semantics.

    On the module level, polymorphic types correspond to
    signatures. There is a notion of principal signatures. So-called
    signature checking is the module level equivalent of type
    checking. In particular, there exists an algorithm which either
    fails or produces a principal signature.",
  paper = "Toft88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Toft96,
  author = "Tofte, Mads",
  title = {{Essentials of Standard ML Modules}},
  year = "1996",
  abstract =
    "The following notes give an overview of Standard ML Module
     system.

    Part 1 gives an introduction to ML Modules aimed at the reader who
    is familiar with a functional programming language but has little
    or no experience with ML programming.

    Part 2 is a half-day practical intended to give the reader an
    opportunity to modify a small, but non-trivial piece of software
    using functors, signatures and structures.",
  paper = "Toft96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Tond19,
  author = "Tonder, Rijnard van and Le Goues, Claire",
  title = {{Lightweight Multi-Language Syntax Transformation with
            Parser Parser Combinators}},
  booktitle = "PLDI",
  publisher = "ACM",
  isbn = "978-1-4503-6712-7",
  year = "2019",
  paper = "Tond19.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Torl17,
  author = "Torlak, Emina",
  title = {{Synthesis and Verifcation for All}},
  link = "\url{https://www.youtube.com/watch?v=KpDyuMIb_EO}",
  year = "2017"
}
```

———-

— **axiom.bib** —

```
@article{Tour88,
```

```
author = "de la Tour, Thierry Boy and Caferra, Richardo",
title = {{A Formal Approach to some Usually Informal Techniques used
          in Mathematical Reasoning}},
journal = "Lecture Notes in Computer Science",
volume = "358",
pages = "402-408",
year = "1988",
abstract =
  "One of the striking characteristics of mathematical reasoning is
  the contrast between the formal aspects of mathematical truth and
  the informal character of the ways to that truth. Among the many
  important and usually informal mathematical activities we are
  interested by proof analogy (i.e. common pattern between proofs of
  different theorems) in the context of interactive theorem
  proving. In some sense we propose a partial contribution of one of
  the Polya's wishes [Polya 73]:
  \begin{quote}
  Analogy pervades all our thinking, our everyday speech and our
  trivial conclusions as well as artistic way of expression and the
  highest scientific achievements... but analogy may reach the level
  of mathematical precision...
  \end{quote}

  It is a work in philosophy of mathematics [Resnik 75], in which
  mathematics is viewed as studying {\sl patterns} or
  {\sl structures}, which encouraged us to pursue our aim of
  partially formalizing {\sl analogy}. We naturally arrived at the
  need of considering other activities strongly tied to the notion
  of analogy and very well known of the working mathematician:
  {\sl generalization}, {\sl abstraction} and {\sl analysis of
  proofs}.

  Let us assume that the user has to prove a conjecture C. Given a
  proof P for a known theorem T, he describes in the langauge L a
  scheme of P. Then he expresses the syntactic analogy he views as
  relevant between the scheme of P and what ``should be'' a proof
  scheme for C, as a {\sl transformation rule}. This process needs
  analysis of proofs, generalization and abstraction.

  A second order pattern matching algorithm constructs matchers of P
  and its scheme, and applies them to the intended proof scheme of
  the conjecture C. In the best case one obtains a potential proof
  of C, but in general one obtains a more instantiated proof-scheme
  with some ``holes'' to be filled by a theorem prover. In both
  cases a proof-checking process is necessary (analogy is in general
  a heuristic way of thinking).

  A question arises naturally: ``What to do when the assumed analogy
  fails?''. We study in this paper one of the possible answers:
  ``Inforamation may be extracted from the failure in order to
  {\sl suggest lemmas} that try to {\sl semantically} restore
  analogy''. Of coure these lemmas an also serve for detecting wrong
  analogies (for example, if required lemmas clearly cannot be
  theorems.). Two kinds of failure are possible:
```

```
    \begin{enumerate}
    \item The given proof P and its proposed scheme are not matchable
    \item A type error is detected in the instantiated proof scheme
    for C.
    \end{enumerate}

    We give in the following a method to suggest lemmas in the first
    case, and sketch some ideas of how to use such a possibility in
    the second case.",
  paper = "Tour88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Trag79,
  author = "Trager, Barry",
  title = {{Integration of Simple Radical Extensions}},
  journal = "LNCS",
  volume = "72",
  pages = "408-414",
  year = "1979",
  abstract =
    "Risch's landmark paper presented the first decision procedure for
    the integration of elementary functions. In that paper he required
    that the functions appearing in the integrand be algebraically
    independent. Shortly afterwards in [Risalg] and [Ris70] he relaxed
    that restriction and outlined a complete decision procedure for
    the integration of elementary functions in finite
    terms. Unfortunately his algorithms for dealing with algebraic
    functions required considerably more complex machinery than his
    earlier ones for purely transcendental functions. Moses'
    implementation of the earlier approach in MACSYMA demonstrated its
    practicality, whereas the same has yet to be done for Risch's more
    recent approach.

    This paper will show how Risch's earlier techniques can be
    generalized to deal with unnested radicals. While this may seem a
    severe restriction, perusing an integral table such as [Bois61] will
    show that fewer than 1\% of the problems are excluded.",
  paper = "Trag79.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Troe88,
  author = "Troelstra, A.S. and van Dalen, Dirk",
  title = {{Constructivism in Mathematics, Vol 2}},
```

```
  publisher = "North-Holland",
  year = "1988"
}
```

———-

— **axiom.bib** —

```
@book{Troe96,
  author = "Troelstra, A.S. and Schwichtenberg, H.",
  title = {{Basic Proof Theory}},
  year = "1996",
  publisher = "Cambridge",
  isbn = "0-521-77911-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Tryb92,
  author = "Trybulec, Zinaida and Swieczkowska, Halina",
  title = {{Some Remarks on The Language of Mathematical Texts}},
  journal = "Studies in Logic, Grammar and Rhetoric",
  volume = "10/11",
  pages = "103-124",
  year = "1992",
  link = "\url{http://mizar.org/trybulec65/5.pdf}",
  paper = "Tryb92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Tsuj09,
  author = "Tsuji, Kuniaki",
  journal = "Journal of Symbolic Computation",
  title = {{An Improved EZ-GCD Algorithm for Multivariate Polynomials}},
  volume = "44",
  number = "1",
  year = "2009",
  pages = "99-110",
  abstract =
    "The EZ-GCD algorithm often has a bad-zero problem, which has a
    remarkable influence on polynomials with higher-degree terms. In
    this paper, by applying special ideals, the EZ-GCD algorithm for
    sparse polynomials is improved. This improved algorithm greatly
    reduces computational complexity because of the sparseness of
    polynomials. The author expects that the use of these ideals will
```

```
    be useful as a resolution for obtaining a GCD of sparse
    multivariate polynomials with higher-degree terms.",
  paper = "Tsuj09.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Turi36,
  author = "Turing, A. M.",
  title = {{On Computable Numbers, with an Application to the
           Entscheidungsproblem}},
  year = "1936",
  link =
  "\url{https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf}",
  abstract =
    "The ‘‘computeble’’ numbers may be described briefly as the real
    numbers whose expressions as a decimal are calculable by finite
    means. Although the subjec of this paper is ostensibly the
    computable {\sl numbers}, it is almost equally easy to define and
    investigate computable functions of an integral variable or a real
    or computable variable, computable predicates, and so forth. The
    fundamental problems involed are, however, the same in each case,
    and I have chosen the computable numbers for explicit treatment as
    involving the least cumbrous technique. I hope shortly to give an
    account of the relations of the computable numbers, functions, and
    so forth to one another. This will include a development of the
    theory of functions of a real variable expressed in terms of
    computable numbers. According to my definition, a number is
    computable if its decimal can be written down by a machine.",
  paper = "Turi36.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Turi47,
  author = "Turing, A. M.",
  title = {{Lecture to the London Mathematical Society on 20 February
           1947}},
  year = "1947",
  link = "\url{http://www.vordenker.de/downloads/turing-vorlesung.pdf}",
  paper = "Turi47.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Turi48,
  author = "Turing, A. M.",
  title = {{Intelligent Machinery}},
  year = "1948",
  link = "\url{https://weightagnostic.github.io/papers/turning1948.pdf}",
  abstract =
    "The possible ways in which machinery might be made to show
    intelligent behaviour are discussed. The analogy with the human
    brain is used as a guiding principle. It is pointed out that the
    potentialities of the human intelligence can only be realized if
    suitable education is provided. The investigation mainly centres
    round an analogous teaching process applied to machines. The idea
    of an unorganized machine is defined, and it is suggested that the
    infant human cortex is of this nature. Simple examples of such
    machines are given, and their education by means of rewards and
    punishments is discussed. In one case the education process is
    carried through until the organization is similar to that of an
    ACE.",
  paper = "Turi48.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Turi49,
  author = "Turing, A. M.",
  title = {{Checking a Large Routine}},
  journal = "Annals of the History of Computing",
  volume = "6",
  number = "2",
  year = "1949",
  abstract =
    "The standard references for work on program proofs attribute the
    early statement of direction to John McCarthy (e.g. McCarthy
    1963); the first workable methods to Peter Naur (1966) and Robert
    Floyd (1967); and the provision of more formal systems to C.A.R
    Hoare (1969) and Edsger Dijkstra (1976). The early papers of some
    of the computing pioneers, however, show an awareness of the need
    for proofs of program correctness and even present workabe methods
    (.e.g Goldstine and von Neumann 1947; Turing 1949).

    The 1949 paper by Alan M. Turing is remarkable in many
    respects. The three (foolscap) pages of text contain an excellent
    motivation by analogy, a proof of a program with two nested loops,
    and an indication of a general proof method very like that of
    Floyd. Unfortunately, the paper is made extremely difficult to
    read by a large number of transcription errors. For example, all
    instances of the factorial sign (Turing used $|n$) have been
    omitted in the commentary, and ten other identifiers are written
```

```
    incorrectly. It would appear to be worth correcting these errors
    and commenting on the proof from the viewpoint of subsequent work
    on program proofs.

    Turing delivered this paper in June 1949, at the inaugural
    conference of the EDSAC, the computer at Cambridge University
    built under the direction of Maurice V. Wilkes. Turing had been
    writing programs for an electronic computer since the end of 1945
    -- at first for the proposed ACE, the computer project at the
    National Physical Laboraty, but since October 1948 for the
    Manchester prototype computer, of which he was deputy
    director. The references in his paper to $2^{40}$ are reflections
    of the 40-bit ``lines'' of the Manchester machine storage system.

    The following is the text of Turing's 1949 paper, corrected to the
    best of our ability to what we believe Turing intended. We have
    made no changes in spelling, punctuation, or grammar.",
  paper = "Turi49.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Turn85,
  author = "Turner, D. A.",
  title = {{Miranda: A non-strict functional language with polymorphic types}},
  journal = "Lecture Notes in Computer Science",
  volume = "201",
  pages = "1-16",
  year = "1985",
  link = "\url{http://miranda.org.uk/nancy.html}",
  paper = "Turn85.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Turn86,
  author = "Turner, D. A.",
  title = {{An Overview of Miranda}},
  journal = "SIGPLAN Notices",
  volume = "21",
  number = "12",
  pages = "158-166",
  year = "1986",
  link = "\url{http://miranda.org.uk/}",
  paper = "Turn86.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Turn91,
  author = "Turner, Raymond",
  title = {{Constructive Foundations for Functional Languages}},
  publisher = "McGraw-Hill",
  year = "1991",
  isbn = "9780077074111"
}
```

———-

1.2.21 U

— **axiom.bib** —

```
@article{Unga91,
  author = "Ungar, David and Smith, Randall B.",
  title = {{SELF: The Power of Simplicity}},
  journal = "Lisp and Symbolic Computation",
  volume = "4",
  number = "3",
  year = "1991",
  publisher = "Kluwer",
  abstract =
    "SELF is an object-oriented language for exploratory programming
    based on a small number of simple and concrete ideas: prototypes,
    slots, and behavior. Prototypes combine inheritance and instantiation
    to provide a framework that is simpler and more flexible than most
    object-oriented languages. Slots unite variables and procedures into a
    single construct. This permits the inheritance hierarchy to take over
    the function of lexical scoping in conventional languages.  Finally,
    because SELF does not distinguish state from behavior, it narrows the
    gaps between ordinary objects, procedures, and closures.  SELF s
    simplicity and expressiveness offer new insights into object-oriented
    computation.",
  paper = "Unga91.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Unkn20,
  author = "Unknown",
  title = {{Programming is Terrible -- Lessons Learned from a life
            wasted}},
  year = "2020",
  link = "\url{https://programmingisterrible.com/post/65781074112/devils-dictionary-of-programming}"
}
```

———-

1.2.22 V

— **axiom.bib** —

```
@article{Vajd09,
  author = "Vajda, Robert and Jebelean, Tudor and Buchberger, Bruno",
  title = {{Combining Logical and Algebraic Techniques for Matural
            Style Proving in Elementary Analysis}},
  journal = "Mathematics and Computers in SImulation",
  volume = "79",
  number = "8",
  pages = "2310-2316",
  year = "2009",
  link =
    "\url{http://www.risc.jku.at/publications/download/risc_3320/acafin3.pdf}",
  abstract =
    "PCS (Proving-Computing-Solving) [Buchberber 2001] and S-Decomposition
    [Jebelean 2001] are strategies for handling proof problems by
    combining logic inference steps (e.g. modus ponens, Skolemization,
    instantiation) with rewriting steps (application of definitions) and
    solving procedures based on algebraic techniques (e.g. Groebner Bases,
    Cylindrical Algebraic Decomposition).

    If one formalizes the main notions of elementary analysis like
    continuity, convergence, etc., usually a sequence of alternating
    quantifier blocks pops up in the quantifier prefix of the
    corresponding formula. This makes the proof problems involving these
    notions not easy. S-Decomposition strategy is expecially suitable for
    property-preserving problems like continuity of sum, becuase it is
    designed for handling problems where the goal and the main assumptions
    have a similar structure. During proof deduction, existentially
    quantified goals and universal assumptions are handled by introducing
    metavariables, if no suitable ground instance is known in
    adva=nce. For finalizing proof attempts, the metavariables should be
    instantiated in such a way that they satisfy the cumulated algebraic
    constraints collected during the proof attempt. The instantiation
    problem is considered to be difficult in the logical
    calculus. Appropriate instances can be often found using quantifier
    elimination (QE) over real closed fields. In order to obtain witness
    terms we utilize the QE method based on cylindrical algebraic
    decomposition (CAD) [Collins 1975]. However, the QE method alone is
    not sufficient. One needs to pre-process the (closed, quantified)
    conjectured formula and post-process the resulting CAD-structure after
    the call of the QE algorithm.",
  paper = "Vajd09.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Vazo18,
  author = "Vazou, Niki and Breitner, Joachim and Kunkel, Rose and
            Horn, David Van and Hutton, Graham",
  title = {{Theorem Proving for All: Equational Reasoning in Liquid
            Haskell}},
  booktitle = "Haskell'18",
  publisher = "ACM",
  year = "2018",
  isbn = "978-1-4503-5835-4",
  link = "\url{http://www.cs.nott.ac.uk/~pszgmh/tpfa.pdf}",
  abstract =
    "Equational reasoning is one of the key features of pure
    functional languages such as Haskell. To date, however, such
    reasoning always took place eternally to Haskell, either manually
    on paper, or machanised in a theorem prover. This article shows
    how equational reasoning can be performed directly and seamlessly
    within Haskell itself, and be checked using Liquid Haskell. In
    particular, language learners -- to whom external theorem provers
    are out of reach -- can benefit from having their proofs
    mechanically checked. Concretely, we show how the equational
    proofs and derivations from Hutton's textbook can be recast as
    proofs in Haskell (spoiler: they look essentially the same).",
  paper = "Vazo18.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Vell06,
  author = "Vellerman, Daniel J.",
  title = {{How To Prove It}},
  publisher = "Cambridge University Press",
  year = "2006",
  isbn = "978-0-511-16116-2",
  paper = "Vell06.pdf",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{verb20,
  author = "unknown",
  title = {{Verbal Regular Expressions}},
  year = "2020",
  link = "\url{https://github.com/VerbalExpressions/implementation/wiki/List-of-methods-to-implement}"
}
```

———-

— **axiom.bib** —

```
@misc{Vict13,
  author = "Victor, Bret",
  title = {{The Future of Programming}},
  year = "2013",
  link = "\url{https://www.youtube.com/watch?v=8pTEmbeENF4}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Viry84,
  author = "Viry, G.",
  title = {{Simplification of Polynomials in n Variables}},
  booktitle = "International Sympoium on Symbolic and Algebraic
               Manipulation",
  pages = "64-73",
  publisher = "Springer",
  year = "1984",
  comment = "LNCS 174",
  paper = "Viry84.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Voev10,
  author = "Voevodsky, Vladimir",
  title = {{The Equivalence Axiom and Univalent Models of Type
           Theory}},
  year = "2010",
  link = "\url{https://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/CMU_talk.pdf}",
  abstract =
    "I will show how to define, in any type system with dependent
    sums, products and Martin-Lof identity types, the notion of a
    homotopy equivalence between two types and how to formulate the
    Equivalence Axiom which provides a natural way to assert that
    ``two homotopy equivalent types are equal''. I will then sketch a
    construction of a model of one of the standard Martin-Lof type
    theories which satisfies the equivalence axiom and the excluded
    middle thus proving that M.L. type theory with excluded middle and
    equivalence axiom is at least as consistent as ZFC theory.

    Models which satisfy the equivalence axiom are called
    univalent. This is a totally new class of models and I will argue
    that the semantics which they provide leads to the first
    satisfactory approach to type-theoretic formalization of
```

```
    mathematics.",
  paper = "Voev10.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Voev14,
  author = "Voevodsky, Vladimir",
  title = {{Univalent Foundations}},
  link = "\url{http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/2014_IAS.pdf}",
  year = "2014",
  paper = "Voev14.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@techreport{Volp91,
  author = "Volpano, Dennis M. and Geoffrey S.",
  title = {{On the Complexity of ML Typability with Overloading}},
  institution = "Cornell University",
  year = "1991",
  number = "TR91-1210",
  abstract =
    "We examine the complexity of type checking in an ML-style type system
    that permits functions to be overloaded with different types. In
    particular, we consider the extension of the ML Type system proposed
    by Wadler and Blott in the appendix of [WB89], with global overloading
    only, that is, where the only overloading is that which exists in an
    initial type assumption set; no local overloading via over and inst
    expressions is allowed. It is shown that under a correct notion of
    well-typed terms, the problem of determining whether a term is well
    typed with respect to an assumption set in this system is
    undecidable. We then investigate limiting recursion in assumption
    sets, the source of the undecidability. Barring mutual recursion is
    considered, but this proves too weak, for the problem remains
    undecidable. Then we consider a limited form of recursion called
    parametric recursion. We show that although the problem becomes
    decidable under parametric recursion, it appears harder than
    conventional ML typability, which is complete for DEXPTIME [Mai90].",
  paper = "Volp91.pdf"
}
```

———-

1.2.23 W

— axiom.bib —

```
@article{Wald92,
  author = "Waldmann, Uwe",
  title = {{Semantics of Order-sorted Specifications}},
  journal = "Theoretical Computer Science",
  volume = "94",
  number = "1-2",
  year = "1992",
  pages = "1-35",
  abstract =
    "Order-sorted specifications (i.e. many-sorted specifications with
    subsort relations) have been proved to be a useful tool for the
    description of partially defined functions and error handling in
    abstract data types.

    Several definitions for order-sorted algebras have been proposed. In
    some papers an operator symbol, which may be multiply declared, is
    interpreted by a family of functions (overloaded algebras). In other
    papers it is always interpreted by a single function (nonoverloaded
    algebras). On the one hand, we try to demonstrate the differences
    between these two approaches with respect to equality, rewriting and
    completion; on the other hand, we prove that in fact both theories can
    be studied in parallel provided that certain notions are suitably
    defined.

    The overloaded approach differs from the many-sorted and the
    nonoverloaded one in that the overloaded term algebra is not
    necessarily initial. We give a decidable sufficient criterion for the
    initiality of the term algebra, which is less restrictive than
    GJM-regularity as proposed by Goguen, Jouannaud and Meseguer.

    Sort-decreasingness is an important property of rewrite systems since
    it ensures that confluence and Church-Rosser property are equivalent,
    that the overloaded and nonoverloaded rewrite relations agree, and
    that variable overlaps do not yield critical pairs. We prove that it
    is decidable whether or not a rewrite rule is sort-decreasing, even if
    the signature is not regular.

    Finally, we demonstrate that every overloaded completion procedure may
    also be used in the nonoverloaded world, but not conversely, and that
    specifications exist that can only be completed using the
    nonoverloaded semantics.",
  paper = "Wald92.pdf"
}
```

———-

— axiom.bib —

```
@articlen{Wand78,
```

```
  author = "Wand, Mitchell and Friedman, Dan",
  title = {{Compiling Lambda-Expressions Using Continuations and
           Factorizations}},
  journal = "Computer Languages",
  volume = "3",
  pages = "241-263",
  year = "1978",
  abstract =
    "We present a source-level transformation for recursion-removal
    with several interesting characteristics:
    \begin{itemize}
    \item[(i)] the algorithm is simple and provably correct
    \item[(ii)] the stack utilization regime is chosen by the compiler
    rather than being fixed by the run-time environment
    \item[(iii)] the stack is available at the source language level
    so that further optimizations are possible
    \item[(iv)] the algorithm arises from ideas in category theory
    \end{itemize}
    In addition to its implications for compilers, the transformation
    algorithm is useful as an implementation technique for advanced
    LISP-based systems, and one such application is described.",
  paper = "Wand78.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wand87,
  author = "Wand, Mitchell",
  title = {{Complete Type Inference for Simple Objects}},
  booktitle = "Symp. on Logic in Computer Science",
  year = "1987",
  pages = "22-25",
  abstract =
    "The problem of strong typing is considered for a model of
    object-oriented programming systems. These systems permit values which
    are records of other values, and in which fields inside these records
    are retrieved by name. A type system is proposed that permits
    classification of these kinds of values and programs by the type of
    their result, as is usual in strongly-typed programming languages. The
    type system has two important properties: it admits multiple
    inheritance, and it has a syntactically complete type inference system.",
  paper = "Wand87.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wand88,
```

```
  author = "Wand, Mitchell",
  title = {{Corrigendum: Complete Type Inference for Simple Objects}},
  booktitle = "Symp. on Logic in Computer Science",
  year = "1988",
  pages = "5-8",
  abstract =
    "An error has been pointed out in the author's paper (see Proc. 2nd
     IEEE Symp. on Logic in Computer Science, p 37-44 (1987)). It appears
     that there are programs without principal type schemes in the
     system in that paper."
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wand89,
  author = "Wand, Mitchell",
  title = {{Type Inference for Record Concatenation and Multiple
           Inheritance}},
  booktitle = "Logic in Computer Science",
  year = "1989",
  isbn = "0-8186-1954-6",
  abstract =
    "The author shows that the type inference problem for a lambda
    calculus with records, including a record concatenation operator, is
    decidable. He shows that this calculus does not have principal types
    but does have finite complete sets of type, that is, for any term M in
    the calculus, there exists an effectively generable finite set of type
    schemes such that every typing for M is an instance of one of the
    schemes in the set. The author shows how a simple model of
    object-oriented programming, including hidden instance variables and
    multiple inheritance, may be coded in this calculus. The author
    concludes that type inference is decidable for object-oriented
    programs, even with multiple inheritance and classes as first-class
    values.",
  paper = "Wand89.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wand91,
  author = "Wand, Mitchell",
  title = {{Type Inference for Record Concatenation and Multiple
           Inheritance}},
  journal = "Information and Computation",
  volume = "93",
  issue = "1",
  year = "1991",
  pages = "1-15",
```

```
  abstract =
    "We show that the type inference problem for a lambda calculus with
    records, including a record concatenation operator, is decidable. We
    show that this calculus does not have principal types, but does have
    finite complete sets of types: that is, for any term M in the
    calculus, there exists an effectively generable finite set of type
    schemes such that every typing for M is an instance of one of the
    schemes in the set. We show how a simple model of object-oriented
    programming, including hidden instance variables and multiple
    inheritance, may be coded in this calculus. We conclude that type
    inference is decidable for object-oriented programs, even with
    multiple inheritance and classes as first-class values.",
  paper = "Wand91.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wang84,
  author = "Wang, Paul S.",
  title = {{Implementation of a p-adic Package for Polynomial
            Factorization and other Related Operations}},
  booktitle = "International Symbposium on Symbolic and Algebraic
               Manipulation",
  pages = "86-99",
  publisher = "Springer",
  year = "1984",
  comment = "LNCS 174",
  abstract =
    "The design and implementation of a $p-adic$ package, called
     {\bf P}-pack, for polynomial factorization, gcd, squarefree
     decomposition and univariate partial fraction expansion are
     presented. {\bf P}-pack is written in FRANZ LISP, and can be
     loaded into VAXIMA and run without modification. The physical
     organization of the code modules and their logical relations are
     described. Sharing of code among different modules and techniques
     for improved speed are discussed.",
  paper = "Wang84.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Watt85,
  author = "Watt, Stephen M.",
  title = {{A System for Parallel Computer Algebra Programs}},
  booktitle = "European COnference on Computer Algebra",
  publisher = "Springer",
  pages = "537-538",
  year = "1985",
```

```
  comment = "LNCS 204",
  paper = "Watt85.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Watt06,
  author = "Watt, Stephen M.",
  title = {{Making Computer Algebra More Symbolic}},
  journal = "Proc. Transgressive Computing",
  pages = "43-49",
  year = "2006",
  abstract =
    "This paper is a step to bring closer together two views of
    computing with mathematical objects: the view of ‘‘symbolic
    computation’’ and the view of ‘‘computer algebra’’. Symbolic
    computation may be seen as working with expression trees
    representing mathematical formulae and applying various rules to
    transform them. Computer algebra may be seen as developing
    constructive algorithms to compute algebraic quanties in various
    arithmetic domains, possibly involving indeterminates. Symbolic
    computation allows a wider range of expression, while computer
    algebra admits greater algorithmic precision. We examine the
    problem of providing polynomials symbolic exponents. We present a
    natural algebraic structure in which such polynomials may be
    defined and a notion of factorization under which these
    polynomials form a UFD.",
  paper = "Watt06.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Watt06a,
  author = "Watt, Stephen M.",
  title = {{Post Facto Type Extension for Mathematical Programming}},
  booktitle = "Workshop on Domain Specific Aspect Language",
  publisher = "ACM",
  year = "2006",
  abstract =
    "We present the concept of {\sl post facto extensions}, which may
    be used to enrich types after they have been defined. Adding
    exported behaviours without altering data representation permits
    existing types to be augmented without renaming. This allows large
    libraries to be structured in a clean, layered fashion and allows
    independently developed software components to be used
    together. This form of type extension has been found to be
    particularly useful in mathematical software, where often new
    abstractions are applicable to existing objects. We describe an
```

```
    implementation of post facto extension, as provided by Aldor, and
    explain how it has been used to structure a large mathematical
    library.",
  paper = "Watt06a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Webe92b,
  author = "Weber, Andreas",
  title = {{Structuring the Type System of a Computer Algebra System}},
  link = "\url{http://cg.cs.uni-bonn.de/personal-pages/weber/publications/pdf/WeberA/Weber92a.pdf}",
  institution = "Wilhelm-Schickard-Institut fur Informatik",
  year = "1992",
  abstract =
    "Most existing computer algebra systems are pure symbol manipulating
    systems without language support for the occuring types. This is
    mainly due to the fact taht the occurring types are much more
    complicated than in traditional programming languages. In the last
    decade the study of type systems has become an active area of
    research. We will give a proposal for a type system showing that
    several problems for a type system of a symbolic computation system
    can be solved by using results of this research. We will also provide
    a variety of examples which will show some of the problems that remain
    and that will require further research.",
  paper = "Webe92b.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@article{Webe95a,
  author = "Weber, Kenneth",
  title = {{The Accelerated Integer GCD Algorithm}},
  journal = "Trans. on Mathematica Software",
  volume = "21",
  number = "1",
  year = "1995",
  pages = "111-122",
  abstract =
    "Since the greatest common divisor (GCD) of two integers is a
    basic arithmetic operation, used in many mathematical software
    systems, new algorithsm for its computation are of widespread
    interest. The accelerated integer GCD algorithm discussed here is
    based on a reduction step proposed by Sorenson (k-ary reduction),
    coupled with the dmod operation similar to Norton's smod. Some
    practical limitations of Sorenson's reduction have been
```

```
    eliminated. Worst-case complexity is still $O(n^2)$ for $n$-bit
    input, but actual implementations given input about 4000 bits long
    perform over 5.5 times as fast as the binary GCD on one computer
    architecture having a multiply instruction. Independent research
    by Jebelean points to the same conclusion.",
  paper = "Webe95a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wirs82,
  author = "Wirsing, Martin and Broy, Manfred",
  title = {{An Analysis of Semantic Models for Algebraic Specifications}},
  booktitle = "Theoretical Foundations of Programming Methodology",
  year = "1982",
  publisher = "Springer",
  pages = "351-413",
  isbn = "978-94-009-7893-5",
  abstract =
    "Data structures, algorithms and programming languages can be
    described in a uniform implementation-independent way by axiomatic
    abstract data types i.e. by algebraic specifications defining
    abstractly the properties of objects and functions. Different semantic
    models such as initial and terminal algebras have been proposed in
    order to specify the meaning of such specifications -often involving a
    considerable amount of category theory. A more concrete semantics
    encompassing these different approaches is presented:

    Abstract data types are specified in hierarchies, employing
    ‘‘primitive’’ types on which other types are based. The semantics is
    defined to be the class of all partial heterogeneous algebras
    satisfying the axioms and respecting the hierarchy. The interpretation
    of a specification as its initial or terminal algebra is just a
    constraint on the underlying data. These constraints can be modified
    according to the specification goals. E.g. the data can be specified
    using total functions; for algorithms partial functions with
    syntactically checkable domains seem appropriate whereas for
    programming languages the general notion of partiality is needed,
    Model-theoretic and deduction-oriented conditions are developed which
    ensure properties leading to criteria for the soundness and complexity
    of specifications. These conditions are generalized to parameterized
    types, i.e. type procedures mapping types into types. Syntax and
    different semantics of parameter are defined and discussed. Criteria
    for proper parameterized specifications are developed. It is shown
    that the properties of proper specifications viz. of snowballing and
    impeccable types are preserved under application of parameterized
    types  finally guaranteeing that the composition of proper small
    specifications always leads to a proper large specification."
}
```

———-

— **axiom.bib** —

```
@InCollection{Wirs91,
  author = "Wirsing, Martin",
  title = {{Algebraic Specification}},
  booktitle = "Handbook of Theoretical Computer Science (Vol B)",
  publisher = "MIT Press",
  year = "1991",
  pages = "675-788",
  chapter = "13",
  isbn = "0-444-88074-7"
}
```

———-

— **axiom.bib** —

```
@book{Wirt13,
  author = "Wirth, Niklaus",
  title = {{Project Oberon}},
  publisher = "ACM Press",
  isbn = "0-201-54428-8",
  year = "2013"
}
```

———-

— **axiom.bib** —

```
@misc{Wirt14,
  author = "Wirth, Niklaus",
  title = {{Reviving a Computer System of 25 Years Ago}},
  link = "\url{https://www.youtube.com/watch?v=EXY78gPMvl0}",
  year = "2014"
}
```

———-

— **axiom.bib** —

```
@book{Wolf91,
  author = "Wolfram, Stephen",
  title = {{Mathematica: A System for Doing Mathematics by Computer}},
  publisher = "Addison-Wesley",
  isbn = "978-0201515022",
  year = "1991"
}
```

———-

— **axiom.bib** —

```
@article{Worz80,
  author = {W\"orz-Busekros, A.},
  title = {{Algebra in Genetics}},
  publisher = "Springer-Verlag",
  journal = "Lecture Notes in Biomathematics",
  volume = "36",
  year = "1980",
  algebra = "\newline\refto{domain ALGSC AlgebraGivenByStructuralConstants}"
}
```

———-

— **axiom.bib** —

```
@misc{Wiki17a,
  author = "Wikipedia",
  title = {{Group Action}},
  year = "2017",
  link = "\url{https://en.wikipedia.org/wiki/Group\_action}",
  abstract =
    "In mathematics, an action of a group is a way of interpreting the
    elements of the group as ‘‘acting’’ on some space in a way that
    preserves the structure of that space. Common examples of spaces that
    groups act on are sets, vector spaces, and topological spaces. Actions
    of groups on vector stacces are called representations of the group."
}
```

———-

— **axiom.bib** —

```
@misc{Wiki17b,
  author = "Wikipedia",
  title = {{Strong generating set}},
  year = "2017",
  link = "\url{http://en.wikipedia.org/wiki/Strong\_generating\_set}",
  abstract =
    "In abstract algebra, especially in the area of group theory, a strong
    generating set of a permutation group is a generating set that clearly
    exhibits the permutation structure as described by a stabilizer
    chain. A stabilizer chain is a sequence of subgroups, each containing
    the next and each stabilizing one more point."
}
```

———-

— **axiom.bib** —

```
@misc{Wiki17c,
```

```
  author = "Wikipedia",
  title = {{Compound matrix}},
  year = "2017",
  link = "\url{https://en.wikipedia.org/wiki/Compound\_matrix}"
}
```

———-

— **axiom.bib** —

```
@techreport{Wrig92,
  author = "Wright, Andrew K. and Felleisen, Mattias",
  title = {{A Syntactic Approach to Type Soundness}},
  type = "technical report",
  institution = "Rich University",
  number = "TR91-160",
  year = "1992",
  abstract =
    "We present a new approach to proving type soundness for
    Hindley/Milner-style polymorphic type systems. The keys to our
    approach are (1) an adaptation of subject reduction theorems from
    combinatory logic to programming languages, and (2) the use of
    rewriting techniques for the specification of the language
    semantics. The approach easily extends from polymorphic functional
    languages to imperative languages that provide references,
    exceptions, continuations, and similar features. We illustrate the
    technique with a type soundness theorem for the core of Standard
    ML, which includes the first type soundess proof for polymorphic
    exceptions and continuations.",
  paper = "Wrig92.pdf",
  keywords = "printed"
}
```

———-

1.2.24 X

1.2.25 Y

1.2.26 Z

— **axiom.bib** —

```
@book{Zari75,
  author = "Zariski, Oscar and Samuel, Pierre",
  title = {{Commutative Algebra}},
  Series = "Graduate Texts in Mathematics",
  year = "1975",
  publisher = "Springer-Verlag",
  isbn = "978-0387900896"
}
```

———-

1.3 Linear Algebra

— axiom.bib —

```
@Unpublished{Kalt01,
  author = "Kaltofen, E.",
  title = {{Algorithms for sparse and black box matrices
           over finite fields (Invited talk)}},
  note = "unknown",
  year = "2001",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/01/Ka01_Fq6.pdf}",
  keywords = "survey",
  abstract = "
    Sparse and structured matrices over finite fields occur in many
    settings.  Sparse linear systems arise in sieve-based integer factoring
    and discrete logarithm algorithms. Structured matrices arise in
    polynomial factoring algorithms; one example is the famous Q-matrix
    from Berlekamp's method.  Sparse diophantine linear problems, like
    computing the Smith canonical form of an integer matrix or computing
    an integer solution to a sparse linear system, are reduced via p-adic
    lifting to sparse matrix analysis over a finite field.

    In the past 10 years there has been substantial activity on the
    improvement of a solution proposed by Wiedemann in 1986. The main new
    ingredients are faster pre-conditioners, projections by an entire
    block of random vectors, Lanczos recurrences, and a connection to
    Kalman realizations of control theory. My talk surveys these
    developments and describe some major unresolved problems.",
  paper = "Kalt01.pdf"
}
```

———-

— axiom.bib —

```
@Article{Chen02,
  author = "Chen, L. and Eberly, W. and Kaltofen, E.
            and Saunders, B. D. and Turner, W. J. and Villard, G.",
  title = {{Efficient Matrix Preconditioners for Black Box Linear Algebra}},
  journal = "Linear Algebra and Applications",
  year = "2002",
  volume = "343--344",
  pages = "119--146",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/02/CEKSTV02.pdf}",
  abstract = "
    The main idea of the ``black box'' approach in exact linear algebra is
    to reduce matrix problems to the computation of minimum polynomials.
```

```
    In most cases preconditioning is necessary to obtain the desired
    result.  Here good preconditioners will be used to ensure geometrical
    / algebraic properties on matrices, rather than numerical ones, so we
    do not address a condition number. We offer a review of problems for
    which (algebraic) preconditioning is used, provide a bestiary of
    preconditioning problems, and discuss several preconditioner types to
    solve these problems.  We present new conditioners, including
    conditioners to preserve low displacement rank for Toeplitz-like
    matrices. We also provide new analyses of preconditioner performance
    and results on the relations among preconditioning problems and with
    linear algebra problems.  Thus, improvements are offered for the
    efficiency and applicability of preconditioners. The focus is on
    linear algebra problems over finite fields, but most results are valid
    for entries from arbitrary fields.",
  paper = "Chen02.pdf"
}
```

———-

— **axiom.bib** —

```
@InCollection{Kalt11d,
  author = "Kaltofen, Erich and Storjohann, Arne",
  title = {{The Complexity of Computational Problems in Exact Linear Algebra}},
  booktitle = "Encyclopedia of Applied and Computational Mathematics",
  publisher = "Springer",
  year = "2011",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/11/KS11.pdf}",
  abstract = "
    Computational problems in exact linear algebra including computing an
    exact solution of a system of linear equations with exact scalars,
    which can be exact rational numbers, integers modulo a prime number,
    or algebraic extensions of those represented by their residues modulo
    a minimum polynomial. Classical linear algebra problems are computing
    for a matrix its rank, determinant, characteristic and minimal
    polynomial, and rational canonical form (= Frobenius normal form). For
    matrices with integer and polynomial entries one computes the Hermite
    and Smith normal forms. If a rational matrix is symmetric, one
    determines if the matrix is definite.",
  paper = "Kalt11d.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Come12,
  author = "Comer, Matthew T. and Kaltofen, Erich L.",
  title = {{On the {Berlekamp}/{Massey} Algorithm and Counting Singular
           {Hankel} Matrices over a Finite Field}},
  year = "2012",
  month = "April",
```

```
  journal = "Journal of Symbolic Computation",
  volume  = "47",
  number = "4",
  pages = "480--491",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/10/CoKa10.pdf}",
  abstract = "
    We derive an explicit count for the number of singular $n\times n$
    Hankel (Toeplitz) matrices whose entries range over a finite field
    with $q$ elements by observing the execution of the Berlekamp / Massey
    algorithm on its elements. Our method yields explicit counts also when
    some entries above or on the anti-diagonal (diagonal) are fixed. For
    example, the number of singular $n\times n$ Toeplitz matrices with 0's
    on the diagonal is $q^{2n-3}+q^{n-1}-q^{n-2}$.

    We also derive the count for all $n\times n$ Hankel matrices of rank
    $r$ with generic rank profile, I.e., whose first $r$ leading principal
    submatrices are non-singular and the rest are singular, namely
    $q^r(q-1)^r$ in the case $r < n$ and $q^{r-1}(q-1)^r$ in the case
    $r=n$.  This result generalizes to block-Hankel matrices as well.",
  paper = "Come12.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt13a,
  author = "Kaltofen, Erich and Yuhasz, George",
  title = {{A Fraction Free Matrix {Berlekamp}/{Massey} Algorithm}},
  journal = "Linear Algebra and Applications",
  year = "2013",
  volume = "439",
  number = "9",
  month = "November",
  pages = "2515--2526",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/08/KaYu08.pdf}",
  abstract = "
    We describe a fraction free version of the Matrix Berlekamp / Massey
    algorithm. The algorithm computes a minimal matrix generator of
    linearly generated square matrix sequences over an integral
    domain. The algorithm performs all operations in the integral domain,
    so all divisions performed are exact. For scalar sequences, the matrix
    algorithm specializes to a more efficient algorithm than the algorithm
    currently in the literature.  The proof of integrality of the matrix
    algorithm gives a new proof of integrality for the scalar
    specialization.",
  paper = "Kalt13a.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt13,
  author = "Kaltofen, Erich and Yuhasz, George",
  title = {{On The Matrix {Berlekamp}-{Massey} Algorithm}},
  year = "2013",
  volume = "9",
  number = "4",
  month = "September",
  journal = "ACM Trans. Algorithms",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/06/KaYu06.pdf}",
  abstract = "
    We analyze the Matrix Berlekamp / Massey algorithm, which generalizes
    the Berlekamp / Massey algorithm [Massey 1969] for computing linear
    generators of scalar sequences. The Matrix Berlekamp / Massey
    algorithm computes a minimal matrix generator of a linearly generated
    matrix sequence and has been first introduced by Rissanen [1972a],
    Dickinson et al. [1974], and Coppersmith [1994]. Our version of the
    algorithm makes no restrictions on the rank and dimensions of the
    matrix sequence. We also give new proofs of correctness and complexity
    for the algorithm, which is based on self-contained loop invariants
    and includes an explicit termination criterion for a given
    determinantal degree bound of the minimal matrix generator",
  paper = "Kalt13.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt02a,
  author = "Kaltofen, Erich",
  title = {{An output-sensitive variant of the baby steps/\allowbreak
           giant steps determinant algorithm}},
  booktitle = "Proc. 2002 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "138--144",
  year = "2002",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/02/Ka02.pdf}",
  paper = "Kalt02a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt01a,
  author = "Kaltofen, E. and Villard, G.",
  title = {{On the complexity of computing determinants}},
  booktitle = "Proc. Fifth Asian Symposium on Computer Mathematics
              (ASCM 2001)",
  pages = "13--27",
  isbn = "981-02-4763-X",
  year = "2001",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/01/KaVi01.pdf}",
```

```
  abstract = "
    The computation of the determinant of an $n\times n$ matrix $A$ of
    numbers or polynomials is a challenge for both numerical and symbolic
    methods. Numerical methods, such as Clarkson's algorithm [10,7] for
    the sign of the determinant must deal with conditionedness that
    determines the number of mantissa bits necessary for obtaining a
    correct sign. Symbolic algorithms that are based on Chinese
    remaindering [6,17,Chapter 5.5] must deal with the fact that the
    length of the determinant in the worse case grows linearly in the
    dimension of the matrix. Hence the number of modular operations is $n$
    times the number of arithmetic operations in a given algorithm.
    Hensel lifting combined with rational number recovery [14,1] has cubic
    bit complexity in $n$, but the algorithm can only determine a factor
    of the determinant, namely the largest invariant factor. If the matrix
    is similar to a multiple of the identity matrix, the running time is
    again that of Chinese remaindering.",
  paper = "Kalt01a.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt04a,
  author = "Kaltofen, Erich and Villard, Gilles",
  title = {{On the Complexity of Computing Determinants}},
  journal = "Computational Complexity",
  volume = "13",
  number = "3-4",
  year = "2004",
  pages = "91--130",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/04/KaVi04_2697263.pdf}",
  abstract = "
    We present new baby steps / giant steps algorithms of asymptotically
    fast running time for dense matrix problems. Our algorithms compute
    the determinant, characteristic polynomial, Frobenius normal form and
    Smith normal form of a dense $n\times n$ matrix $A$ with integer
    entries in $(x^{3.2} log ||A||)^{1+o(1)}$ and
    $(x^{2.697263} log ||A||)^{1+o(1)}$
    bit operations; here $||A||$ denotes the largest
    entry in absolute value and the exponent adjustment by ``$+o(1)$''
    captures additional factors $C_1 (log n)^{C_2}(loglog ||A||)^{C_3}$
    for positive real constants $C_1$, $C_2$, $C_3$. The bit complexity
    $(n^{3.2} log ||A||)^{1+o(1)}$ results from using the classical cubic
    matrix multiplication algorithm.  Our algorithms are randomized, and
    we can certify that the output is the determinant of $A$ in a Las
    Vegas fashion. The second category of problems deals with the setting
    where the matrix $A$ has elements from an abstract commutative ring,
    that is, when no divisions in the domain of entries are possible. We
    present algorithms that deterministically compute the determinant,
    characteristic polynomial and adjoint of $A$ with $n^{3.2+o(1)}$ and
    $O(n^{2.697263})$ ring additions, substractions, and multiplications.",
  paper = "Kalt04a.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt97b,
  author = "Eberly, W. and Kaltofen, E.",
  title = {{On Randomized {Lanczos} Algorithms}},
  booktitle = "Proc. 1997 Internat. Symp. Symbolic Algebraic Comput.",
  year = "1997",
  pages = "176--183",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/97/EbKa97.pdf}",
  abstract = "
    Las Vegas algorithms that are based on Lanczo's method for solving
    symmetric linear systems are presented and analyzed. These are
    compared to a similar randomized Lanczos algorithm that has been used
    for integer factorization, and to the (provably reliable) algorithm of
    Wiedemann. The analysis suggests that our Lanczos algorithms are
    preferable to several versions of Wiedemann's method for computations
    over large fields, expecially for certain symmetric matrix
    computations.",
  paper = "Kalt97b.pdf"
}
```

———-

The Sylvester matrix is used to compute the **resultant** of two polynomials. The Sylvester matrix is formed from the coefficients of the two polynomials. Given a polynomial with degree m and another of degree n form an $(m+n) \times (m+n)$ matrix by filling the matrix from the upper left corner with the coefficients of the first polynomial then shifting down one row and one column to the right and filling in the coefficients starting there until they hit the right column. Starting at the next row, do the same process for the second polynomial. The determinant of this matrix is the **resultant** of the two polynomials.

For example, given $a_3x^3 + a_2x^2 + a_1x + a_0$ and $b_2x^2 + b_1x + b_0$ the Sylvester matrix is a $(3+2) \times (3+2)$ matrix:

$$\begin{bmatrix} a_3 & a_2 & a_1 & a_0 & 0 \\ 0 & a_3 & a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 & 0 & 0 \\ 0 & b_2 & b_1 & b_0 & 0 \\ 0 & 0 & b_2 & b_1 & b_0 \end{bmatrix}$$

The resultant of these two polynomials (assuming a leading coefficient of 1), is the product of the differences $p_i - q_i$ between the roots of the polynomials. If there are roots in common then the product will contain a 0 and the whole equation reduces to 0. This can be used to determine if two polynomials have common roots.

For example, given a polynomial in x with distinct roots a_1 and a_2 it can be factored as $t1 := (x - a_1)(x - a_2)$.

Given a second polynomial in x with distinct roots b_1, b_2, and b_3 it can be factored as $t2 := (x - b_1)(x - b_2)(x - b_3)$.

The Axiom call of $resultant(t1, t2, x)$ is

$$(b_1 - a_2)(b_1 - a_1)(b_2 - a_2)(b_2 - a_1)(b_3 - a_2)(b_3 - a_1)$$

In symbolic form the resultant can show the multiplicity of roots when shown in factored form.

— **axiom.bib** —

```
@InProceedings{Kalt94c,
  author = "Kaltofen, E.",
  title = {{Asymptotically fast solution of {Toeplitz}-like singular
           linear systems}},
  booktitle = "Proc. 1994 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "297--304",
  year = "1994",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/94/Ka94_issac.pdf}",
  abstract = "
    The Toeplitz-likeness of a matrix (Kailath et al. 1979) is the
    generalization of the notion that a matrix is Toeplitz. Block matrices
    with Toeplitz blocks, such as the Sylvester matrix corresponding to
    the resultant of two univariate polynomials, are Toeplitz-like, as are
    products and inverses of Toeplitz-like matrices. The displacement rank
    of a matrix is a measure for the degree of being Toeplitz-like. For
    example, an $r\times s$ block matrix with Toeplitz blocks has
    displacement rank $r+s$ whereas a generic $N\times N$ matrix has
    displacement rank $N$.  A matrix of displacement rank $\alpha$ can be
    implicitly represented by a sum of $\alpha$ matrices, each of which is
    the product of a lower trainagular and an upper triangular Toeplitz
    matrices. Such a $\sigma LU$ representation can usually be obtained
    efficiently.",
  paper = "Kalt94c.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt99,
  author = "Kaltofen, E. and Lobo, A.",
  title = {{Distributed matrix-free solution of large sparse linear systems
           over finite fields}},
  journal = "Algorithmica",
  year = "1999",
  pages = "331--348",
  month = "July--Aug.",
  volume = "24",
  number = "3--4",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/99/KaLo99.pdf}",
  abstract = "
    We describe a coarse-grain parallel approach for the homogeneous
    solution of linear systems. Our solutions are symbolic, i.e., exact
    rather than numerical approximations. We have performed an outer loop
```

```
    parallelization that works well in conjunction with a black box
    abstraction for the coefficient matrix. Our implementation can be run
    on a network cluster of UNIX workstations as well as on an SP-2
    multiprocessor. Task distribution and management are effected through
    MPI and other packages. Fault tolerance, checkpointing, and recovery
    are incorporated. Detailed timings are presented for experiments with
    systems that arise in RSA challenge integer factoring efforts. For
    example, we can solve a 252,222$\times$252,222 system with about 11.04
    million nonzero entries over the Galois field with two elements using
    four processors of an SP-2 multiprocessor, in about 26.5 hours CPU time.",
  paper = "Kalt99.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt96a,
  author = "Kaltofen, E. and Lobo, A.",
  title = {{Distributed matrix-free solution of large sparse linear systems
           over finite fields}},
  booktitle = "Proc. High Performance Computing '96",
  year = "1996",
  editor = "A. M. Tentner",
  pages = "244--247",
  organization = "Society for Computer Simulation",
  publisher = "Simulation Councils, Inc.",
  address = "San Diego, CA",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/96/KaLo96_hpc.pdf}",
  abstract = "
    We describe a coarse-grain parallel software system for the
    homogeneous solution of linear systems. Our solutions are symbolic,
    i.e., exact rather than numerical approximations. Our implementation
    can be run on a network cluster of SPARC-20 computers and on an SP-2
    multiprocessor.  Detailed timings are presented for experiments with
    systems that arise in RSA challenge integer factoring efforts. For
    example, we can solve a 252,222$\times$252,222 system with about 11.04
    million non-zero entries over the Galois field with 2 elements using 4
    processors of an SP-2 multiprocessor, in about 26.5 hours CPU time.",
  paper = "Kalt96a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt94a,
  author = "Kaltofen, E. and Lobo, A.",
  title = {{Factoring high-degree polynomials by the black box
           Berlekamp algorithm}},
  booktitle = "Proc. 1994 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "90--98",
```

```
  year = "1994",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/94/KaLo94.ps.gz}",
  abstract = "
    Modern techniques for solving structured linear systems over finite
    fields, which use the coefficient matrix as a black box and require an
    efficient algorithm for multiplying this matrix by a vector, are
    applicable to the classical algorithm for factoring a univariate
    polynomial over a finite field by Berlekamp (1967 and 1970). We report
    aon a computer implementation of this idea that is based on the
    parallel block Wiedemann linear system solver (Coppersmith 1994 and
    Kaltofen 1993 and 1995). The program uses randomization and we also
    study the expected run time behavior of our method.",
  paper = "Kalt94a.ps"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt95,
  author = "Kaltofen, E.",
  title = {{Analysis of {Coppersmith}'s block {Wiedemann} algorithm for the
           parallel solution of sparse linear systems}},
  journal = "Math. Comput.",
  year = "1995",
  volume = "64",
  number = "210",
  pages = "777--806",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/95/Ka95_mathcomp.pdf}",
  abstract = "
    By using projections by a block of vectors in place of a single vector
    it is possible to parallelize the outer loop of iterative methods for
    solving sparse linear systems. We analyze such a scheme proposed by
    Coppersmith for Wiedemann's coordinate recurrence algorithm, which is
    based in part on the Krylov subspace approach. We prove that by use of
    certain randomizations on the input system the parallel speed up is
    roughly by the number of vectors in the blocks when using as many
    processors. Our analysis is valid for fields of entries that have
    sufficiently large cardinality. Our analysis also deals with an
    arising subproblem of solving a singular block Toeplitz system by use
    of the theory of Toeplitz-like matrices.",
  paper = "Kalt95.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt90a,
  author = "Kaltofen, E. and Krishnamoorthy, M.S. and Saunders, B.D.",
  title = {{Parallel algorithms for matrix normal forms}},
  journal = "Linear Algebra and Applications",
```

```
  year = "1990",
  volume = "136",
  pages = "189--208",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/90/KKS90.pdf}",
  abstract = "
    Here we offer a new randomized parallel algorithm that determines the
    Smith normal form of a matrix with entries being univariate
    polynomials with coefficients in an arbitrary field. The algorithm has
    two important advantages over our previous one: the multipliers
    related the Smith form to the input matrix are computed, and the
    algorithm is probabilistic of Las Vegas type, i.e., always finds the
    correct answer. The Smith form algorithm is also a good sequential
    algorithm. Our algorithm reduces the problem of Smith form
    computations to two Hermite form computations.  Thus the Smith form
    problem has complexity asymptotically that of the Hermite form
    problem. We also construct fast parallel algorithms for Jordan normal
    form and testing similarity of matrices. Both the similarity and
    non-similarity problems are in the complexity class RNC for the usual
    coefficient fields, i.e., they can be probabilistically decided in
    poly-logarithmic time using polynomially many processors.",
  paper = "Kalt90a.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt87,
  author = "Kaltofen, E. and Krishnamoorthy, M.S. and Saunders, B.D.",
  title = {{Fast parallel computation of Hermite and Smith forms of
           polynomial matrices}},
  journal = "SIAM J. Alg. Discrete Math.",
  year = "1987",
  volume = "8",
  pages = "683--690",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/87/KKS87.pdf}",
  abstract = "
    Boolean circuits of polynomial size and poly-logarithmic depth are
    given for computing the Hermite and Smith normal forms of polynomial
    matrices over finite fields and the field of rational numbers. The
    circuits for the Smith normal form computation are probabilistic ones
    and also determine very efficient sequential algorithms. Furthermore,
    we give a polynomial-time deterministic sequential algorithm for the
    Smith normal form over the rationals. The Smith normal form algorithms
    are applied to the Rational canonical form of matrices over finite
    fields and the field of rational numbers.",
  paper = "Kalt87.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt92,
  author = "Kaltofen, E. and Pan, V.",
  title = {{Processor-efficient parallel solution of linear systems {II}:
           the positive characteristic and singular cases}},
  booktitle = "Proc. 33rd Annual Symp. Foundations of Comp. Sci.",
  year = "1992",
  pages = "714--723",
  publisher = "IEEE Computer Society Press",
  address = "Los Alamitos, California",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/92/KaPa92.pdf}",
  abstract = "
    We show that over any field, the solution set to a system of $n$
    linear equations in $n$ unknowns can be computed in parallel with
    randomization simultaneously in poly-logarithmic time in $n$ and with
    only as many processors as are utilized to multiply two $n\times n$
    matrices. A time unit represents an arithmetic operation in the
    field. For singular systems our parallel timings are asymptotically as
    fast as those for non-singular systems, due to our avoidance of binary
    search in the matrix rank problem, except when the field has small
    positive characteristic; in that case, binary search is avoided to a
    somewhat higher processor count measure.",
  paper = "Kalt92.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt91c,
  author = "Kaltofen, E. and Pan, V.",
  title = {{Processor efficient parallel solution of linear systems over
           an abstract field}},
  booktitle = "Proc. SPAA '91 3rd Ann. ACM Symp. Parallel Algor. Architecture",
  pages = "180--191",
  publisher = "ACM Press",
  year = "1991",
  address = "New York, N.Y.",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/91/KaPa91.pdf}",
  abstract = "
    Parallel randomized algorithms are presented that solve
    $n$-dimensional systems of linear equations and compute inverses of
    $n\times n$ non-singular matrices over a field in $O((log n)^2)$ time,
    where each time unit represents an arithmetic operation in the field
    generated by the matrix entries. The algorithms utilize with a $O(log n)$
    factor as many processors as are needed to multiply two $n\times n$
    matrices. The algorithms avoid zero divisions with controllably
    high probability provided the $O(n)$ random elements used are selected
    uniformly from a sufficiently large set. For fields of small positive
    characteristics, the processor count measures of our solutions are
    somewhat higher.",
  paper = "Kalt91c.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt91,
  author = "Kaltofen, E. and Saunders, B.D.",
  editor = "H. F. Mattson and T. Mora and T. R. N. Rao",
  title = {{On Wiedemann's method of solving sparse linear systems}},
  booktitle = "Proc. AAECC-9",
  series = "Lect. Notes Comput. Sci.",
  volume = "539",
  pages = "29--38",
  publisher = "Springer-Verlag",
  year = "1991",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/91/KaSa91.pdf}",
  abstract = "
    Douglas Wiedemann's (1986) landmark approach to solving sparse linear
    systems over finite fields provides the symbolic counterpart to
    non-combinatorial numerical methods for solving sparse linear systems,
    such as the Lanczos or conjugate gradient method (see Golub and van
    Loan (1983)). The problem is to solve a sparse linear system, when the
    individual entries lie in a generic field, and the only operations
    possible are field arithmethic; the solution is to be exact. Such is
    the situation, for instance, if one works in a finite field. Wiedemann
    bases his approach on Krylov subspaces, but projects further to a
    sequence of individual field elements. By making a link to the
    Berlekamp / Massey problem from coding theory -- the coordinate
    recurrences -- and by using randomization an algorithm is obtained
    with the following property. On input of an $n\times n$ coefficient
    matrix $A$ given by a so-called black box, which is a program that can
    multiply the matrix by a vector (see Figure 1), and of a vector $b$,
    the algorithm finds, with high probability in case the system is
    solvable, a random solution vector $x$ with $Ax=b$. It is assumed that
    the field has sufficiently many elements, say no less than $50n^2
    log(x)$, otherwise one goes to a finite algebraic extension. The
    complexity of the method is in the general singular case $O(n log
    (n))$ calls to the black box for $A$ and an additional $O(n^2
    log(n)^2)$ field arithmetic operations.",
  paper = "Kalt91.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wied86,
  author = "Wiedemann, Douglas H.",
  title = {{Solving sparse linear equations over finite fields}},
  journal = "IEEE Transactions on Information Theory",
  year = "1986",
  volume = "32",
  number = "1",
  pages = "54-62",
```

```
  link = "\url{http://www.csd.uwo.ca/~moreno/CS424/Ressources/WIEDEMANN-IEE-1986.pdf}",
  abstract = "
    A ``coordinate recurrence'' method for solving sparse systems of
    linear equations over finite fields is described. The algorithms
    discussed all require $O(n_1(\omega+n_1)log^k n_1)$ field operations,
    where $n_1$ is the maximum dimension of the coefficient matrix,
    $\omega$ is approximately the number of field operations required to
    apply the matrix to a test vector, and the value of $k$ depends on the
    algorithm. A probabilistic algorithm is shown to exist for finding the
    determinant of a square matrix. Also, probabilistic algorithms are
    shown to exist for finding the minimum polynomial and rank with some
    arbitrarily small possibility of error.",
  paper = "Wied86.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bord82,
  author = "Bordoni, L. and Colagrossi, A. and Miola, A.",
  title = {{Linear Algebraic Approach for Computing Polynomial Resultant}},
  journal = "Lecture Notes in Computer Science",
  volume = "144",
  pages = "231-236",
  year = "1982",
  abstract =
    "This paper presents a linear algebraic method for computing the
    resultant of two polynomials. This method is based on the
    computation of a determinant of order equal to the minimum of the
    degrees of the two given polynomials. This method turns out to be
    preferable to other known linear algebraic methods both from a
    computational point of view and for a total generality respect to
    the class of the given polynomials. Some relationships of this
    method with the polynomial pseudo-remainder operation are discussed.",
  paper = "Bord82.pdf"
}
```

———-

1.4 Algebraic Algorithms

— **axiom.bib** —

```
@article{Belo83,
  author = "Belovari, G.",
  title =
    {{Complex Analysis in Symbolic Computing of some Definite Integrals}},
  journal = "ACM SIGSAM",
  volume = "17",
```

```
  number = "2",
  year = "1983",
  pages = "6-11",
  paper = "Belo83.pdf"
}
```

———-

— axiom.bib —

```
@article{Brow78,
  author = "Brown, W. S.",
  title = {{The Subresultant PRS Algorithm}},
  journal = "ACM Transactions on Mathematical Software",
  volume = "4",
  number = "3",
  year = "1978",
  pages = "237-249",
  link =
    "\url{https://people.eecs.berkeley.edu/~fateman/282/readings/brown.pdf}",
  abstract =
    "Two earlier papers described the generalizations of Euclid's
    algorithm to deal with the problem of computing the greatest common
    divisor (GCD) or the resultant of a pair of polynomials. A sequal to
    those two papers is presented here.

    In attempting such a generalization one easily arrives at the concept
    of a polynomial remainder sequence (PRS) and then quickly discovers
    the phenomenon of explosive coefficient growth.  Fortunately, this
    explosive growth is not inherent in the problem, but is only an
    artifact of various naive solutions. If one removes the content (that
    is, the GCD of the coefficients) from each polynomial in a PRS, the
    coefficient growth in the resulting primitive PRS is relatively modest.
    However, the cost of computing the content (by applying Euclid's
    algorithm in the coefficient domain) may be unacceptably or even
    prohibitively high, especially if the coefficients are themselves
    polynomials in one or more addltional variables.

    The key to controlling coefficient growth without the costly
    computation of GCD's of coefficients is the fundamental theorem of
    subresuitants, which shows that every polynomial in a PRS is
    proportional to some subresultant of the first two.  By arranging for
    the constants of proportionahty to be unity, one obtains the
    subresultant PRS algorithm, in which the coefficient growth is
    essentially linear.  A corollary of the fundamental theorem is given
    here, which leads to a simple derivation and deeper understanding of
    the subresultant PRS algorithm and converts a conjecture mentioned in
    the earlier papers into an elementary remark.

    A possible alternative method of constructing a subresultant PRS is to
    evaluate all the subresultants directly from Sylvester's determinant
    via expansion by minors. A complexity analysis is given in conclusion,
    along lines pioneered by Gentleman and Johnson, showing that the
```

```
    subresultant PRS algorithm is superior to the determinant method
    whenever the given polynomials are sufficiently large and dense, but
    is inferior in the sparse extreme.",
  paper = "Brow78.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Cann73,
  author = "Cannon, John J. and Dimino, Lucien A. and Havas, George and
            Watson, Jane M.",
  title = {{Implementation and Analysis of the Todd-Coxeter Algorithm}},
  year = "1973",
  journal = "Mathematics of Computation",
  volume = "27",
  number = "123",
  pages = "463-490",
  abstract =
    "A recent form of the Todd-Coxeter algorithm, known as the lookahead
    algorithm, is described. The time and space requirements for this
    algorithm are shown experimentally to be usually either equivalent or
    superior to the Felsch and Haselgrove-Leech-Trotter algorithms. Some
    finding from an experimental study of the behaviour of Todd-Coxeter
    programs in a variety of situations is given.",
  paper = "Cann73.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Coll87,
  author = "Collins, George E.",
  title = {{Subresultants and Reduced Polynomial Remainder Sequences}},
  journal = "J. ACM",
  volume = "14",
  number = "1",
  year = "1987",
  pages = "128-142",
  link =
   "\url{https://people.eecs.berkeley.edu/~fateman/282/readings/collins.pdf}",
  paper = "Coll87.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Cool65,
```

```
  author = "Cooley, James W. and Tukey, John W.",
  title = {{An Algorithm for the Machine Calculation of Complex Fourier
           Series}},
  journal = "Mathematics of Computation",
  volume = "19",
  number = "04",
  year = "1965",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Dave17,
  author = "Davenport, James H.",
  title = {{A Generalized Successive Resultants Algorithm}},
  journal = "Lecture Notes in Computer Science",
  volume = "10064",
  year = "2017",
  abstract =
    "The Successive Resultants Algorithm (SRA) is a root-finding algorithm for
    polynomials over $\mathbb{F}_{p^n}$ and was introdoced at ANTS in 2014.
    THe algorithm is efficient when the characteristic $p$ is small and
    $n > 1$. In this paper, we abstract the core SRA algorithm to arbitrary
    finite fields and present three instantiations of our general algorithm,
    one of which is novel and makes use of a series of isogenies derived
    from elliptic curves with sufficiently smooth order.",
  paper = "Dave17.pdf"
}
```

———-

— **axiom.bib** —

```
@InCollection{Diaz97,
  author = "Diaz, A. and Kaltofen, E. and Pan, V.",
  title = {{Algebraic Algorithms}},
  booktitle = "The Computer Science and Engineering Handbook",
  publisher = "CRC Press",
  year = "1997",
  editor = "A. B. Tucker",
  pages = "226--248",
  address = "Boca Raton, Florida",
  chapter = "10",
  keywords = "survey",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/97/DKP97.ps.gz}",
  ref = "00965",
  abstract = "
    The title's subject is the algorithmic approach to algebra: arithmetic
    with numbers, polynomials, matrices, differential polynomials, such as
    $y^{\prime\prime} + (1/2 + x^4/4)y$, truncated series,
```

```
    and algebraic sets, i.e.,
    quantified expressions such as $\exists x \in {\bf R}: x^4+p\cdot x+q=0$,
    which describes a subset of the two-dimensional space with
    coordinates $p$ and $q$ for which the given quartic equation has a
    real root.  Algorithms that manipulate such objects are the backbone
    of modern symbolic mathematics software such as the Maple and
    Mathematica systems, to name but two among many useful systems. This
    chapter restricts itself to algorithms in four areas: linear matrix
    algebra, root finding ov univariate polynomials, solution of systems
    of nonlinear algebraic equations, and polynomial factorization.",
  paper = "Diaz97.ps"
}
```

———-

— **axiom.bib** —

```
@InCollection{Diaz99,
  author = "Diaz, A. and Emiris, I. and Kaltofen, E. and Pan, V.",
  title = {{Algebraic Algorithms}},
  booktitle = "Algorithms \& Theory of Computation Handbook",
  publisher = "CRC Press",
  year = "1999",
  editor = "M. J. Atallah",
  address = "Boca Raton, Florida",
  pages = "16.1--16.27",
  isbn = "0-8493-2649-4",
  keywords = "survey",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/99/DEKP99.ps.gz}",
  abstract = "
    The title's subject is the algorithmic approach to algebra: arithmetic
    with numbers, polynomials, matrices, differential polynomials, such as
    $y^n+(1/2+x^4/4)y$, truncated series, and algebraic sets, i.e.,
    quantified expressions such as $\exists x \in {\bf R}: x^4+p\cdot+q=0$,
    which describes a subset of the two-dimensional space with
    coordinates $p$ and $q$ for which the given quartic equation has a
    real root.  Algorithms that manipulate such objects are the backbone
    of modern symbolic mathematics software such as the Maple and
    Mathematica systems, to name but two among many useful systems. This
    chapter restricts itself to algorithms in four areas: linear algebra,
    root finding for univariate polynomials, solution of systems of
    nonlinear algebraic equations, and polynomial factorization (see
    section 5 on some pointers to the vast further material on algebraic
    algorithms and section 2.2 and [Pan 1993] on further applications to
    the graph and combinatorial computations).",
  paper = "Diaz99.ps"
}
```

———-

— **axiom.bib** —

```
@misc{Doye93,
  author = "Doye, Nicolas James",
  title = {{The Implementation of Various Algorithms for Permuation Groups
           in the Computer Algebra System: AXIOM}},
  year = "1993",
  comment = "M.Sc thesis, University of Bath",
  link = "\url{https://static.worldofnic.org/cdn/ps/research/msc.ps}",
  paper = "Doye93.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Fate92a,
  author = "Fateman, Richard",
  title = {{Honest Plotting, Global Extrema, and Interval Arithmetic}},
  booktitle = "ISSAC 92",
  year = "1992",
  pages = "216-223",
  isbn = "0-89791-489-9",
  abstract =
    "A computer program to honestly plot curves y = f(x) must locate
    maxima and minima in the domain of the graph.  To do so it may have to
    solve a classic problem in computation  global optimization.
    Reducing an easy problem to a hard one is usually not an advantage,
    but in fact there is a route to solving both problems if the function
    can be evaluated using interval arithmetic.  Since some computer
    algebra systems supply a version of interval arithmetic, it seems we
    have the ingredients for a solution.

    In this paper we address a particular problem how to compute and
    display ``honest'' graphs of 2-D mathematical curves.  By
    ``honest'' we mean that no significant features (such as the
    location of poles, the values at maxima or minima, or the behavior
    of a curve at asymptotes) are misrepresented, By mathematical we
    mean curves like those generally needed in scientific disciplines
    where functions are represented by composition of common
    mathematical operations: rational operations ($+, , *, /$),
    exponential and log, trigonometric functions as well as continuous
    and differentiable functions from applied mathematics.",
  paper = "Fate92a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate00b,
  author = "Fateman, Richard",
  title = {{The (finite field) Fast Fourier Transform}},
```

```
  year = "2000",
  link =
   "\url{https://people.eecs.berkeley.edu/~fateman/282/readings/fftnotes.pdf}",
  abstract =
    "There are numerous directions from which one can approach the subject
    of the fast Fourier Transform (FFT). It can be explained via numerous
    connections to convolution, signal processing, and various other
    properties and applications of the algorithm.  We (along with
    Geddes/Czapor/Labahn) take a rather simple view from the algebraic
    manipulation standpoint. As will be apparent shortly, we relate the
    FFT to the evaluation of a polynomial. We also consider it of interest
    primarily as an algorithm in a discrete (finite) computation structure
    rather than over the complex numbers.",
  paper = "Fate00b.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate00c,
  author = "Fateman, Richard",
  title = {{Additional Notes on Polynomial GCDs, Hensel construction}},
  year = "2000",
  link =
    "\url{https://people.eecs.berkeley.edu/~fateman/282/readings/hensel.pdf}",
  paper = "Fate00c.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gent76,
  author = "Gentleman, W. M. and Johnson, S. C.",
  title = {{Analysis of Algorithms, A Case Study: Determinants of Matrices
           With Polynomial Entries}},
  journal = "ACM Transactions on Mathematical Software",
  volume = "2",
  number = "3",
  year = "1976",
  pages = "232-241",
  link =
  "\url{https://people.eecs.berkeley.edu/~fateman/282/readings/gentleman.pdf}",
  abstract =
    "The problem of computing the deternunant of a matrix of polynomials
    is considered; two algorithms (expansion by minors and expansion by
    Gaussian elimination) are compared; and each is examined under two models
    for polynomial computatmn (dense univariate and totally sparse).  The
    results, while interesting in themselves, also serve to display two
    points: (1) Asymptotic results are sometimes misleading for noninfinite
    (e.g.  practical) problems.  (2) Models of computation are by
```

```
    definition simplifications of reality: algorithmic analysis should be
    carried out under several distinct computatmnal models and should be
    supported by empirical data.",
  paper = "Gent76.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Hard64,
  author = "Hardy, G. and Littlewood, J.E. and Polya, G.",
  title = {{Inequalities}},
  publisher = "Cambridge University Press",
  year = "1964"
}
```

———-

— **axiom.bib** —

```
@InCollection{Kalt87a,
  author = "Kaltofen, E.",
  editor = "J. F. Traub",
  title = {{Computer algebra algorithms}},
  booktitle = "Annual Review in Computer Science",
  pages = "91--118",
  publisher = "Annual Reviews Inc.",
  year = "1987",
  volume = "2",
  address = "Palo Alto, California",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/87/Ka87_annrev.pdf}",
  abstract = "
    The origins of the discipline of computer algebra can be found in
    Issac Newton's {\sl Universal Arithmetic} (1728) [130], where methods
    for methods for manipulating universal mathematical expressions (i.e.
    formulas containing symbolic indeterminates) and algorithms for
    solving equations built with these expressions are systematically
    discussed. One can interpret the misson of computer algebra as the
    construction of computer systems that enable scientific or engineering
    users, for instance, to carry out mathematical manipulation
    automatically. Indeed, systems with this goal already exist, among
    them {MACSYMA}, {MAPLE}, {muMATH}, {REDUCE}, {SAC/2}, {SCRATCHPAD/II},
    and {SMP}. These systems carry out scientific computing tasks, whose
    results are distinguished from numerical computing in two principle
    aspects.",
  paper = "Kalt87a.pdf",
  keywords = "survey,axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Kemp81,
  author = "Kempelmann, Helmut",
  title = {{Recursive Algorithm for the Fast Calculation of the Limit of
           Derivatives at Points of Indeterminateness}},
  journal = "ACM SIGSAM",
  volume = "15",
  number = "4",
  year = "1981",
  pages = "10-11",
  abstract =
    "It is a common method in probability and queueing theory to gain the
    $n$-th moment $E[x^n]$ of a random variable $X$
    with density function $f_x(x)$
    by the $n$-th derivative of the corresponding Laplace transform $L(s)$ at
    the point $s = 0$
    \[E[x^n] = (-1)^n\cdot L^{n}(0)\]
    Quite often we encounter indetermined
    expressions of the form $0/0$ which normally are treated by the rule of
    L'Hospital. This is a time and memory consuming task requiring
    greatest common divisor cancellations. This paper presents an
    algorithm that calculates only those derivatives of numerator and
    denominator which do not equal zero when taking the limit /1/. The
    algorithm has been implemented in REDUCE /2/. It is simpler and more
    efficient than that one proposed by /3/.",
  paper = "Kemp81.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Laza82,
  author = "Lazard, Daniel",
  title = {{Commutative Algebra and Computer Algebra}},
  journal = "Lecture Notes in Computer Science",
  volume = "144",
  pages = "40-48",
  year = "1982",
  abstract =
    "It is well known that computer algebra deals with commutative rings
    such as the integers, the rationals, the modular integers and
    polynomials over such rings.

    On the other hand, commutative algebra is that part of mathematics
    which studies commutative rings.

    Our aim is to make this link more precise. It will appear that most of
    the constructions which appear in classical commutative algebra can be
    done explicitly in finite time. However, there are
    exceptions. Moreover, most of the known algorithms are intractable :
    The problems are generally exponential by themselves, but many
```

```
    algorithms are over-over-exponential. It needs a lot of work to find
    better methods, and to implement them, with the final hope to open a
    computation domain larger than this one which is possible by hand.

    To illustrate the limits and the possibilities of computerizing
    commutative algebra, we describe an algorithm which tests the
    primality of a polynomial ideal and we give an example of a single
    algebraic extension of fields $K\subset L$ wuch that there exists an
    algorithm of factorization for the polynomials over $k$, but not for
    the polynomials over $L$.",
  paper = "Laza82.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Laza82a,
  author = "Lazard, Daniel",
  title = {{On Polynomial Factorization}},
  journal = "Lecture Notes in Computer Science",
  volume = "144",
  pages = "144-157",
  year = "1982",
  abstract =
    "We present new algorithms for factoring univariate polynomials
    over finite fields. They are variants of the algorithms of Camion
    and Cantor-Zassenhaus and differ from them essentially by
    computing the primitive idempotents of the algebra $K[X]/f$ before
    factoring $f$.

    These algorithms are probabilistic in the following sense. The
    time of computation depends on random choices, but the validity of
    the result does not depend on them. So, worst case complexity,
    being infinite, is meaningless and we compute average
    complexity. It appears that our and Cantor-Zassenhaus algorithms
    have the same asymptotic complexity and they are the best
    algorithms actuall known; with elementary multiplication and GCD
    computation, Cantor-Zassenhaus algorithm is always bettern than
    ours; with fast multiplication and GCD, it seems that ours is
    better, but this fact is not yet proven.

    Finally, for factoring polynomials over the integers, it seems
    that the best strategy consists in choosing prime moduli as big as
    possible in the range where the time of the multiplication does
    not depend on the size of the factors (machine word size). An
    accurate computation of the involved constants would be needed for
    proving this estimation.",
  paper = "Laza82a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Loos72a,
  author = "Loos, Rudiger",
  title = {{Analytic Treatment of Three Similar Fredholm Integral Equations
           of the second kind with REDUCE 2}},
  journal = "ACM SIGSAM",
  volume = "21",
  year = "1972",
  pages = "32-40"
}
```

————-

— **axiom.bib** —

```
@article{Norf82,
  author = "Norfolk, Timothy S.",
  title = {{Symbolic Computation of Residues at Poles and Essential
           Singularities}},
  journal = "ACM SIGSAM",
  volume = "16",
  number = "1",
  year = "1982",
  pages = "17-23",
  abstract =
    "Although most books on the theory of complex variables include a
    classification of the types of isolated singularities, and the
    applications of residue theory, very few concern themselves with
    methods of computing residues. In this paper we derive some results on
    the calculation of residues at poles, and some special classes of
    essential singularities, with a view to implementing an algorithm in
    the VAXIMA computer algebra system.",
  paper = "Norf82.pdf"
}
```

————-

— **axiom.bib** —

```
@article{Padg82,
  author = "Padget, J.A.",
  title = {{Escaping from Intermediate Expression Swell: A Continuing Saga}},
  journal = "Lecture Notes in Computer Science",
  volume = "144",
  pages = "256-262",
  year = "1982",
  abstract =
    "The notion of a closed continuation is introduced, is presented,
    coroutines using function call and return based on this concept, are
    applications and a functional dialect of LISP shown to be merely a
    more general form of for coroutines in algebraic simplification and
```

```
    are suggested, by extension function.  expression Potential evaluation
    and a specific example of their use is given in a novel attack on the
    phenomenon of intermediate expression swell in polynomial
    multiplication.",
  paper = "Padg82.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Plat09,
  author = "Platzer, Andre and Quesel, Jan-David and Rummer, Philipp",
  title = {{Real World Verification}},
  journal = "LNCS",
  volume = "5663",
  year = "2009",
  pages = "495-501",
  link = "\url{http://www.cs.cmu.edu/~aplatzer/pub/rwv.pdf}",
  abstract =
    "Scalable handling of real arithmetic is a crucial part of the
    verification of hybrid systems, mathematical algorithms, and mixed
    analog/digital circuits.  Despite substantial advances in verification
    technology, complexity issues with classical decision procedures are
    still a major obstacle for formal verification of real-world
    applications, e.g., in automotive and avionic industries.  To identify
    strengths and weaknesses, we examine state of the art symbolic
    techniques and implementations for the universal fragment of
    real-closed fields: approaches based on quantifier elimination,
    Groebner Bases, and semidefinite programming for the
    Positivstellensatz. Within a uniform context of the verification tool
    KeYmaera, we compare these approaches qualitatively and quantitatively
    on verification benchmarks from hybrid systems, textbook algorithms,
    and on geometric problems. Finally, we introduce a new decision
    procedure combining Groebner Bases and semidefinite programming for
    the real Nullstellensatz that outperforms the individual approaches on
    an interesting set of problems.",
  paper = "Plat09,pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Schr17,
  author = "Wikipedia",
  title = {{Schreier-Sims algorithm}},
  year = "2017",
  link =
    "\url{https://en.wikipedia.org/wiki/Schreier\%E2%80%93Sims\_algorithm}",
  abstract =
    "The Schreier-Sims algorithm is an algorithm in computational group
```

```
    theory named after mathematicians Otto Schreier and Charles Sims. Once
    performed, it allows a linear time computation of the order of a
    finite permuation group, group membership test (is a given permutation
    contained in a group?), and many other tasks. The algorith was
    introduced by Sims in 1970, based on Schreier's subgroup lemma. The
    timing was subsequently improved by Donald Knuth in 1991. Later, an
    even faster randomized version of the algorithm was developed.

    The algorithm is an efficient method of computing a base and strong
    generating set (BSGS) of a permutation group. In particular, an SGS
    determines the order of a group and makes it easy to test membership
    in the group. Since the SGS is critical for many algorithms in
    computational group theory, computer algebra systems typically rely on
    the Schreier-Sims algorithm for efficient calculations in groups"
}
```

———-

— **axiom.bib** —

```
@book{Somm01,
  author = "Sommer, Gerald",
  title = {{Geometric Computing with Clifford Algebras}},
  year = "2001",
  publisher = "Springer",
  isbn = "3-540-41198-4"
}
```

———-

— **axiom.bib** —

```
@misc{Wang90a,
  author = "Wang, Dongming",
  title = {{Some NOtes on Algebraic Methods for Geometry Theorem Proving}},
  link = "\url{http://www-polsys.lip6.fr/~wang/papers/GTPnote.ps.gz}",
  year = "1990",
  abstract =
    "A new geometry theorem prover which provides the first complete
    implementation of Wu's method and includes several Groebner bases
    based methods is reported. This prover has been used to prove a number
    of non-trivial geometry theorems including several {\sl large} ones
    with less space and time cost than using the existing provers. The
    author presents a new technique by introducing the notion of {\sl
    normal ascending set}. This technique yields in some sense {\sl
    simpler} non-degenerate conditions for Wu's method and allows one to
    prove geometry theorems using characteristic sets but Groeber bases
    type reduction. Parallel variants of Wu's method are discussed; an
    implementation of the parallelized version of his algorithm utilizing
    workstation networks has also been included in our prover. Timing
    statistics for a set of typical examples is given.",
  paper = "Wang90a.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wang92,
  author = "Wang, Dongming",
  title = {{A Method for Factorizing Multivariate Polynomials over Successive
           Algebraic Extension Fields}},
  booktitle = "Mathematics and Mathematics-Mechanization (2001)",
  pages = "138-172",
  institution = "Johannes Kepler University",
  link = "\url{http://www-polsys.lip6.fr/~wang/papers/Factor.ps.gz}",
  year = "1992",
  abstract =
    "We present a method for factorizing multivariate polynomials over
    algebraic fields obtained from successive extensions of the rational
    number field. The basic idea underlying this method is the reduction
    of polynomial factorization over algebraic extension fields to the
    factorization over the rational number vield via linear transformation
    and the computation of characteristic sets with respect to a proper
    variable ordering. The factors over the algebraic extension fields are
    finally determined via GCD (greatest common divisor) computations. We
    have implemented this method in the Maple system. Preliminary
    experiments show that it is rather efficient. We give timing
    statistics in Maple 4.3 on 40 test examples which were partly taken
    from the literature and partly randomly generated. For all those
    examples to which Maple built-in algorithm is applicable, our
    algorithm is always faster.",
  paper = "Wang92.pdf"
}
```

———-

1.5 Sparse Linear Systems

— **axiom.bib** —

```
@InProceedings{Kalt96b,
  author = "Kaltofen, E.",
  title = {{Blocked iterative sparse linear system solvers for finite fields}},
  booktitle = "Proc. Symp. Parallel Comput. Solving Large Scale Irregular
               Applic. (Stratagem '96)",
  editor = "C. Roucairol",
  publisher = "INRIA",
  address = "Sophia Antipolis, France",
  pages = "91--95",
  year = "1996",
  keywords = "survey",
```

```
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/96/Ka96_stratagem.ps.gz}",
  abstract = "
    The problem of solving a large sparse or structured system of linear
    equations in the symbolic context, for instance when the coefficients
    lie in a finite field, has arisen in several applications. A famous
    example are the linear systems of ${\bf F}_2$, the field with 2
    elements, that arise in sieve based integer factoring algorithms. For
    example, for the factorization of the RSA-130 challenge number several
    column dependencies of a $3504823\times 3516502$ matrix with an
    average of $39.4.$ non-zero entries per column needed to be computed
    [10]. A second example is the Berlekamp polynomial factorization
    algorithm [6]. In that example, the matrix is not explicitly
    constructed, but instead a fast algorithm for performing the matrix
    times vector product is used. Further examples for such ``black box
    matrices'' arise in the power series solution of algebraic or
    differential equations by undetermined coefficients. The arising
    linear systems for the coefficients usually have a distinct structure
    that allows a fast coefficient matrix times vector product.",
  paper = "Kalt96b.ps"
}
```

———-

1.6 Matrix Determinants

— **axiom.bib** —

```
@Article{Kalt04,
  author = "Kaltofen, E. and Villard, G.",
  title = {{Computing the sign or the value of the determinant of an integer
           matrix, a complexity survey}},
  journal = "J. Computational Applied Math.",
  volume = "162",
  number = "1",
  month = "January",
  pages = "133--146",
  year = "2004",
  keywords = "survey",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/02/KaVi02.pdf}",
  abstract = "
    Computation of the sign of the determinant of a matrix and the
    determiant itself is a challenge for both numerical and exact
    methods. We survey the complexity of existing methods to solve these
    problems when the input is an $n\times n$ matrix $A$ with integer
    entries. We study the bit-complexities of the algorithms
    asymptotically in $n$ and the norm $A$.  Existing approaches rely on
    numerical approximate computations, on exact computations, or on both
    types of arithmetic in combination.",
  paper = "Kalt04.pdf"
}
```

1.7 Open Problems

— **axiom.bib** —

```
@Article{Kalt00,
  author = "Kaltofen, E.",
  title = {{Challenges of Symbolic Computation My Favorite Open Problems}},
  journal = "Journal of Symbolic Computation",
  volume = "29",
  number = "6",
  pages = "891--919",
  year = "2000",
  keywords = "survey",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/2K/Ka2K.pdf}",
  abstract = "
    The success of the symbolic mathematical computation discipline is
    striking. The theoretical advances have been continuous and significant:
    Gr{\"o}bner bases, the Risch integration algorithm, integer lattice
    basis reduction, hypergeometric summation algorithms, etc. From the
    beginning in the early 60s, it has been the tradition of our discipline
    to create software that makes our ideas readily available to a
    scientists, engineers, and education: {SAC-1}, {Reduce}, {Macsyma}, etc.
    The commercial viability of our system products is proven by Maple and
    Mathematica.

    Today's user communities of symbolic computation systems are diverse:
    educators, engineers, stock market anaysts, etc. The mathematics and
    computer science in the design and implementation of our algorithms are
    sophisticated. The research challenges in symbolic computation at the
    close of the 20th century are formidable.

    I state my favorite eight open problems in symbolic computations. They
    range from problems in symbolic /numeric computing, symbolic algorithm
    synthesis, to system component construction. I have worked on seven of
    my problems and borrowed one from George Collins. I present background
    to each of my problems and a clear-cut test that evaluates whether a
    proposed attack has solved one of my problems. An additional ninth
    open problem by Rob Corless and David Jeffrey on complex function
    semantics is given in an appendix.",
  paper = "Kalt00.pdf"
}
```

1.8 Parallel Evaluation

— **axiom.bib** —

```
@InCollection{Kalt93a,
  author = "Kaltofen, E.",
  editor = "J. Reif",
  title = {{Dynamic parallel evaluation of computation {DAG}s}},
  booktitle = "Synthesis of Parallel Algorithms",
  pages = "723--758",
  publisher = "Morgan Kaufmann Publ.",
  year = "1993",
  address = "San Mateo, California",
  keywords = "survey",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/93/Ka93_synthesis.ps.gz}",
  abstract = "
    One generic parallel evaluation scheme for algebraic objects, that of
    evaluating algebraic computation trees or formulas, is presented by
    Miller in a preceding chapter of this book. However, there are basic
    algebraic functions for which the tree model of computation seems not
    sufficient to allow an eficient -- even sequential -- decision-free
    algebraic computation. The formula model essentially restricts the use
    of an intermediate result to a single place, because the parse tree
    nodes have fan-out one. If an intermediate result participates in the
    computations of several further nodes, in the tree model it must be
    recomputed anew for each of these nodes. It is a small formal change
    to allow node values to propagate to more than one node deeper level
    of the computation. Thus we obtain the {\sl algebraic circuit model},
    which is equivalent to the {\sl straight-line program model}.",
  paper = "Kalt93a.ps"
}
```

———-

1.9 Hybrid Symbolic/Numeric

— **axiom.bib** —

```
@misc{Corl93,
  author = "Corless, R. M. and Gonnet, G. H. and Hare, D. E. G. and
            Jeffrey, D. J. and Knuth, D. E.",
  title = {{On the Lambert W Function}},
  year = "1993",
  link = "\url{http://cs.uwaterloo.ca/research/tr/1993/03/W.pdf}",
  abstract =
   "The Lambert W function is defined to be the multivalued inverse of
    the function $w \rightarrow we^w$. It has many applications in pure
    and applied mathematics, some of which are briefly described here.  We
    present a new discussion of the complex branches of $W$, an asymptotic
    expansion valid for all branches, an efficient numerical procedure for
    evaluating the function to arbitrary precision, and a method for the
    symbolic integration of expressions containing $W$.",
  paper = "Corl93.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Hutt10,
  author = "Hutton, Sharon E. and Kaltofen, Erich L. and Zhi, Lihong",
  title = {{Computing the radius of positive semidefiniteness of a
           multivariate real polynomial via a dual of {Seidenberg}'s method}},
  year = "2010",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'10",
  pages = "227--234",
  month = "July",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/10/HKZ10.pdf}",
  abstract = "
    We give a stability criterion for real polynomial inequalities with
    floating point or inexact scalars by estimating from below or
    computing the radius of semdefiniteness. That radious is the maximum
    deformation of the polynomial coefficent vector measured in a weighted
    Euclidean vector norm within which the inequality remains true. A
    large radius means that the inequalities may be considered numerically
    valid.

    The radius of positive (or negative) semidefiniteness is the distance
    to the nearest polynomial with a real root, which has been thoroughly
    studied before. We solve this problem by parameterized Lagrangian
    multipliers and Karush-Kuhn-Tucker conditions. Our algorithms can
    compute the radius for several simultaneous inequalities including
    possibly additional linear coefficient constraints. Our distance
    measure is the weighted Euclidena coefficient norm, but we also
    discuss several formulas for the weighted infinity and 1-norms.

    The computation of the nearest polynomial with a real root can be
    interpreted as a dual of Seidenberg's method that decides if a real
    hypersurface contains a real point. Sums-of-squares rational lower
    bound certificates for the radius of semidefinitesness provide a new
    approach to solving Seidenberg's problem, especially when the
    coeffcients are numeric. They also offer a surprising alternative
    sum-of-squares proof for those polynomials that themselves cannot be
    represented by a polynomial sum-of-squares but that have a positive
    distance to the nearest indefinte polynomial.",
  paper = "Hutt10.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt06,
  author = "Kaltofen, Erich and Zhi, Lihong",
  title = {{Hybrid Symbolic-Numeric Computation}},
  year = "2006",
```

```
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'06",
  pages = "7",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/06/KaZhi06.pdf}",
  abstract = "
    Several standard problems in symbolic computation, such as greatest
    common divisors and factorization of polynomials, sparse
    interpolation, or computing solutions to overdetermined systems of
    polynomial equations have non-trivial solutions only if the input
    coefficients satisfy certain algebraic constraints. Errors in the
    coefficients due to floating point round-off or through physical
    measurement thus render the exact symbolic algorithms unusable. By
    symbolic-numeric methods one computes minimal deformations of the
    coefficients that yield non-trivial results. We will present hybrid
    algorithms and benchmark computations based on Gauss-Newton
    optimazation, singular value decomposition (SVD) and
    structure-preserving total least squares (STLS) fitting for several of
    the above problems.

    A significant body of results to solve those ``approximate computer
    algebra'' problems has been discovered in the past 10 years. In the
    Computer Algebra Handbook the section on ``Hybrid Methods'' concludes
    as follows [2]: ``The challenge of hybrid symbolic-numeric algorithms
    is to explore the effects of imprecision, discontinuity, and
    algorithmic complexity by applying mathematical optimization,
    perturbation theory, and inexact arithmetic and other tools in order
    to solve mathematical problems that today are not solvable by
    numeriiical or symbolic methods alone.'' The focus of our tutorial is
    on how to formulate several approximate symbolic computation problems
    as numerical problems in linear algebra and optimization and on
    software that realizes their solutions.",
  paper = "Kalt06.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Hjor10,
  author = "Hjorth-Jensen, Morten",
  title = {{Computational Physics}},
  year = "2010",
  link = "\url{http://depts.washington.edu/ph506/Hjorth-JensenLectures2010.pdf}",
  paper = "Hjor10.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt09,
  author = "Kaltofen, Erich and Yang, Zhengfeng and Zhi, Lihong",
  title = {{A Proof of the {Monotone Column Permanent (MCP) Conjecture} for
```

```
          Dimension 4 via Sums-Of-Squares of Rational Function}},
  year = "2009",
  booktitle = "Proc. 2009 Internat. Workshop on Symbolic-Numeric Comput.",
  pages = "65--69",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/09/KYZ09.pdf}",
  abstract = "
    For a proof of the monotone column permanent (MCP) conjecture for
    dimension 4 it is sufficient to show that 4 polynomials, which come
    rom the permanents of real marices, are nonnegative for all real
    values of the variables, where the degrees and the number of the
    variables of these polynomials are all 8. Here we apply a hybrid
    symbolic-numerical algorithm for certifying that these polynomials can
    be written as an exact fraction of two polynomial sums-of-squares
    (SOS) with rational coefficients.",
  paper = "Kalt09.pdf"
}
```

———-

— axiom.bib —

```
@Article{Kalt12,
  author = "Kaltofen, Erich L. and Li, Bin and Yang, Zhengfeng and
            Zhi, Lihong",
  title = {{Exact Certification in Global Polynomial Optimization
           Via Sums-Of-Squares of Rational Functions
           with Rational Coefficients}},
  year = "2012",
  month = "January",
  journal = "Journal of Symbolic Computation",
  volume  = "47",
  number = "1",
  pages = "1--15",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/09/KLYZ09.pdf}",
  abstract = "
    We present a hybrid symbolic-numeric algorithm for certifying a
    polynomial or rational function with rational coefficients to be
    non-negative for all real values of the variables by computing a
    representation for it as a fraction of two polynomial sum-of-squares
    (SOS) with rational coeficients. Our new approach turns the earlier
    methods by Peyrl and Parrilo and SCN'07 and ours at ISSAC'08 both
    based on polynomial SOS, which do not always exist, into a universal
    algorithm for all inputs via Artin's theorem.

    Furthermore, we scrutinize the all-important process of converting the
    numerical SOS numerators and denomiators produced by block
    semidefinite programming into an exact rational identity. We improve
    on our own Newton iteration-based high precision refinement algorithm
    by compressing the initial Gram matrices and by deploying rational
    vector recovery aside from orthogonal projection. We successfully
    demenstrate our algorithm on 1. various exceptional SOS problems with
    necessary polynomial denominators from the literature and on 2. very
    large (thousands of variables) SOS lower bound certificates for Rump's
```

```
    model problem (up to $n=18$, factor degree $=17$).",
  paper = "Kalt12.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt08b,
  author = "Kaltofen, Erich and Li, Bin and Yang, Zhengfeng and Zhi, Lihong",
  title = {{Exact Certification of Global Optimality of Approximate
           Factorizations Via Rationalizing Sums-Of-Squares
           with Floating Point Scalars}},
  year = "2008",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'08",
  pages = "155--163",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/08/KLYZ08.pdf}",
  abstract = "
    We generalize the technique by Peyrl and Parillo [Proc. SNC 2007] to
    computing lower bound certificates for several well-known
    factorization problems in hybrid symbolic-numeric computation. The
    idea is to transform a numerical sum-of-squares (SOS) representation
    of a positive polynomial into an exact rational identity. Our
    algorithms successfully certify accurate rational lower bounds near
    the irrational global optima for benchmark approximate polynomial
    greatest common divisors and multivariate polynomial irreducibility
    radii from the literature, and factor coefficient bounds in the
    setting of a model problem by Rump (up to $n=14$, factor degree $=13$).

    The numeric SOSes produced by the current fixed precision
    semi-definite programming (SDP) packages (SeDuMi, SOSTOOLS, YALMIP)
    are usually too coarse to allow successful projection to exact SOSes
    via Maple 11's exact linear algebra. Therefore, before projection we
    refine the SOSes by rank-preserving Newton iteration. For smaller
    problems the starting SOSes for Newton can be guessed without SDP
    (``SDP-free SOS''), but for larger inputs we additionally appeal to
    sparsity techniques in our SDP formulation.",
  paper = "Kalt08b.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt06b,
  author = "Kaltofen, Erich and Yang, Zhengfeng and Zhi, Lihong",
  title = {{Approximate greatest common divisors of several polynomials
           with linearly constrained coefficients and singular polynomials}},
  year = "2006",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'06",
  pages = "169--176",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/06/KYZ06.pdf}",
```

```
  abstract = "
    We consider the problem of computing minimal real or complex
    deformations to the coefficients in a list of relatively prime real or
    complex multivariate polynomials such that the deformed polynomials
    have a greatest common divisor (GCD) of a least a given degree $k$. In
    addition, we restrict the deformed coefficients by a given set of
    linear constraints, thus introducing the {\sl linearly constrained
    aproximate GCD} problem.  We present an algorithm based on a version
    of the structured total least norm (STLN) method and demonstrate, on a
    diverse set of benchmark polynomials, that the algorithm in practice
    computes globally minimal approximations. As an application of the
    linearly constrained approximate GCD problem, we present an STLN-based
    method that computes for a real or complex polynomial the nearest real
    or complex polynomial the nearest real or complex polynomial that has
    a root of multiplicity at least $k$.  We demonstrate that the
    algorithm in practice computes, on the benchmark polynomials given in
    the literate, the known globally optimal nearest singular
    polynomials. Our algorithms can handle, via randomized
    preconditioning, the difficult case when the nearest solution to a
    list of real input polynomials actually has non-real complex
    coefficients.",
  paper = "Kalt06b.pdf"
}
```

———-

— **axiom.bib** —

```
@InCollection{Kalt05,
  author = "Kaltofen, Erich and Yang, Zhengfeng and Zhi, Lihong",
  title = {{Structured Low Rank Approximation of a Sylvester Matrix}},
  booktitle = "Symbolic-Numeric Computation",
  publisher = "Springer",
  pages = "69--83",
  year = "2005",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/05/KYZ05.pdf}",
  abstract = "
    The task of determining the approximate greatest common divisor (GCD)
    of univariate polynomials with inexact coefficients can be formulated
    as computing for a given Sylvester matrix a new Sylvester matrix of
    lower rank whose entries are near the corresponding entries of that
    input matrix.  We solve the approximate GCD problem by a new method
    based on structured total least norm (STLN) algorithms, in our case
    for matrices with Sylvester structure. We present iterative algorithms
    that compute an approximate GCD and that can certify an approximate
    $\epsilon$-GCD when a tolerence $\epsilon$ is given on input. Each
    single iteration is carried out with a number of floating point
    operations that is of cubic order in the input degrees. We also
    demonstrate the practical performance of our algorithms on a diverse
    set of univariate pairs of polynomials.",
  paper = "Kalt05.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt03a,
  author = "Kaltofen, Erich and May, John",
  title =
    {{On Approximate Irreducibility of Polynomials in Several Variables}},
  year = "2003",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'03",
  pages = "161--168",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/03/KM03.pdf}",
  abstract = "
    We study the problem of bounding all factorizable polynomials away
    from a polynomial that is absolutely irreducible. Such separation
    bounds are useful for testing whether a numerical polynomial is
    absolutely irreducible, given a certain tolerance on its coefficients
    Using an absolute irreducibility criterion due to Ruppert, we are able
    to find useful separation bounds, in several norms, for bivariate
    polynomials.  We also use Ruppert's criterion to derive new, more
    effective Noether forms for polynomials of arbitrarily many
    variables. These forms lead to small separation bounds for polynomials
    of arbitrarily many variables.",
  paper = "Kalt03a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Gao04a,
  author = "Gao, Shuhong and Kaltofen, Erich and May, John P. and
            Yang, Zhengfeng and Zhi, Lihong",
  title = {{Approximate factorization of multivariate polynomials via
           differential equations}},
  year = "2004",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'04",
  pages = "167--174",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/04/GKMYZ04.pdf}",
  abstract = "
    The input to our algorithm is a multivariate polynomial, whose complex
    rational coefficient are considered imprecise with an unknown error
    that causes $f$ to be irreducible over the complex numbers {\bf C}.
    We seek to perturb the coefficients by a small quantity such that the
    resulting polynomial factors over {\bf C}. Ideally, one would like to
    minimize the perturbation in some selected distance measure, but no
    efficient algorithm for that is known. We give a numerical
    multivariate greatest common divisor algorithm and use it on a
    numerical variant of algorithms by W. M. Ruppert and S. Gao. Our
    numerical factorizer makes repeated use of singular value
    decompositions. We demonstrate on a significant body of experimental
    data that our algorithm is practical and can find factorizable
    polynomials within a distance that is about the same in relative
```

```
    magnitude as the input error, even when the relative error in the
    input is substantial ($10^{-3}$).",
  paper = "Gao04a.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt08,
  author = "Kaltofen, Erich and May, John and Yang, Zhengfeng and Zhi, Lihong",
  title = {{Approximate Factorization of Multivariate Polynomials Using
           Singular Value Decomposition}},
  year = "2008",
  journal = "Journal of Symbolic Computation",
  volume = "43",
  number = "5",
  pages = "359--376",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/07/KMYZ07.pdf}",
  paper = "Kalt08.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Hitz99,
  author = "Hitz, M.A. and Kaltofen, E. and Lakshman, Y.N.",
  title = {{Efficient Algorithms for Computing the Nearest Polynomial
           With A Real Root and Related Problems}},
  booktitle = "Proc. 1999 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "205--212",
  year = "1999",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/99/HKL99.pdf}",
  paper = "Hitz99.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Hitz98,
  author = "Hitz, M. A. and Kaltofen, E.",
  title = {{Efficient Algorithms for Computing the Nearest Polynomial
           with Constrained Roots}},
  booktitle = "Proc. 1998 Internat. Symp. Symbolic Algebraic Comput.",
  year = "1998",
  pages = "236--243",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/98/HiKa98.pdf}",
  paper = "Hitz98.pdf"
}
```

———-

1.10 Software Systems

— axiom.bib —

```
@InProceedings{Diaz91,
  author = "Diaz, A. and Kaltofen,E. and Schmitz, K. and Valente, T.",
  title = {{DSC A System for Distributed Symbolic Computation}},
  booktitle = "Proc. 1991 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "323--332",
  year = "1991",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/91/DKSV91.pdf}",
  paper = "Diaz91.pdf"
}
```

———-

— axiom.bib —

```
@InProceedings{Chan94,
  author = "Chan, K.C. and Diaz, A. and Kaltofen, E.",
  editor = "R. J. Lopez",
  title = {{A distributed approach to problem solving in Maple}},
  booktitle = "Maple V: Mathematics and its Application",
  pages = "13--21",
  publisher = {Birkh\"auser},
  year = "1994",
  series = "Proceedings of the Maple Summer Workshop and Symposium (MSWS'94)",
  address = "Boston",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/94/CDK94.ps.gz}",
  paper = "Chan94.ps"
}
```

———-

— axiom.bib —

```
@InProceedings{Duma02,
  author = "Dumas, J.-G. and Gautier, T. and Giesbrecht, M. and Giorgi, P.
            and Hovinen, B. and Kaltofen, E. and Saunders, B.D. and
            Turner, W.J. and Villard, G.",
  title = {{LinBox: A Generic Library for Exact Linear Algebra}},
  booktitle =     "Proc. First Internat. Congress Math. Software ICMS 2002,
                   Beijing, China",
  pages =         "40--50",
  year = "2002",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/02/Detal02.pdf}",
  paper = "Duma02.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@inproceedings{Einw95,
  author = "Einwohner, T. and Fateman, Richard J.",
  title = {{Searching Techniques for Integral Tables}},
  booktitle = "ISSAC 95",
  year = "1995",
  pages = "133-139",
  abstract =
    "We describe the design of data structures and a computer program for
    storing a table of symbolic indefinite or definite integrals and
    retrieving user-requested integrals on demand.  Typical times are so
    short that a preliminary look-up attempt prior to any algorithmic
    integration approach seems justified.  In one such test for a table
    with around 700 entries, matches were found requiring an average of
    2.8 milliseconds per request, on a Hewlett Packard 9000/712
    workstation.",
  paper = "Einw95.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt05a,
  author = "Kaltofen, Erich and Morozov, Dmitriy and Yuhasz, George",
  title = {{Generic Matrix Multiplication and Memory Management in LinBox}},
  year = "2005",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'05",
  pages = "216--223",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/05/KMY05.pdf}",
  paper = "Kalt05a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Diaz98,
  author = "Diaz, A. and Kaltofen, E.",
  title = {{FoxBox, a System for Manipulating Symbolic Objects in Black Box
           Representation}},
  booktitle = "Proc. 1998 Internat. Symp. Symbolic Algebraic Comput.",
  publisher = "ACM Press",
  year = "1998",
  pages = "30--37",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/98/DiKa98.pdf}",
```

```
  abstract =
    "The FOXBOX system puts in practice the black box representation of
    symbolic objects and provides algorithms for performing the symbolic
    calculus with such representations. Black box objects are stored as
    functions. For instance: a black box polynomial is a procedure that
    takes values for the variables as input and evaluates the polynomial
    at that given point. FOXBOX can compute the greatest common divisor
    and factorize polynomials in black box representation, producing as
    output new black boxes. It also can compute the standard sparse
    distributed representation of a black box polynomial, for example, one
    which was computed for an irreducible factor. We establish that the
    black box representation of objects can push the size of symbolic
    expressions far beyond what standard data structures could handle
    before.

    Furthermore, FOXBOX demonstrates the generic program design
    methodology. The FOXBOX system is written in C++. C++ template
    arguments provide for abstract domain types. Currently, FOXBOX can be
    compiled with SACLIB 1.1, Gnu-MP 1.0, and NTL 2.0 as its underlying
    field and polynomial arithmetic. Multiple arithmetic plugins can be
    used in the same computation. FOXBOX provides an MPI compliant
    distribution mechanism that allows for parallel and distributed
    execution of FOXBOX programs. Finally, FOXBOX plugs into a
    server/client-style Maple application interface.",
  paper = "Diaz98.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Diaz93,
  author = "Diaz, A. and Kaltofen, E. and Lobo, A. and Valente, T.",
  editor = "A. Miola",
  title = {{Process scheduling in DSC and the large sparse linear
           systems challenge}},
  booktitle = "Proc. DISCO '93",
  series = "Lect. Notes Comput. Sci.",
  pages = "66--80",
  year = "1993",
  volume = "722",
  publisher = "Springer-Verlag",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/93/DHKLV93.pdf}",
  paper = "Diaz93.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Diaz95a,
```

```
  author = "Diaz, A. and Hitz, M. and Kaltofen, E. and Lobo, A. and
            Valente, T.",
  title = {{Process scheduling in DSC and the large sparse linear
           systems challenge}},
  journal = "Journal of Symbolic Computing",
  year = "1995",
  volume = "19",
  number = "1--3",
  pages = "269--282",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/95/DHKLV95.pdf}",
  paper = "Diaz95a.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Free88,
  author = "Freeman, T.S. and Imirzian, G. and Kaltofen, E. and
            Yagati, Lakshman",
  title = {{DAGWOOD: A system for manipulating polynomials given by
           straight-line programs}},
  journal = "ACM Trans. Math. Software",
  year = "1988",
  volume = "14",
  number = "3",
  pages = "218--240",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/88/FIKY88.pdf}",
  paper = "Free88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Norm82,
  author = "Norman, Arthur",
  title = {{The Development of a Vector-Based Algebra System}},
  journal = "Lecture Notes in Computer Science",
  volume = "144",
  pages = "237-248",
  year = "1982",
  paper = "Norm82.pdf"
}
```

———-

— **axiom.bib** —

```
@inbook{Norm82a,
  author = "Norman, A.C.",
```

```
  title = {{Integration in Finite Terms}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "57-70",
  abstract =
    "A survey on algorithms for integration in finite terms is
    given. The emphasis is on indefinite integration. Systematic
    methods for rational, algebraic and elementary transcendental
    integrands are reviewed. Heuristic techniques for indefinite
    integration, and techniques for definite integration and ordinary
    differental equations are touched only briefly.",
  paper = "Buch82.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inbook{Norm82b,
  author = "Norman, A.C.",
  title = {{Computing in Transcendental Extensions}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "169-172",
  abstract =
    "Performing the rational operations in a field extended by a
    transcendental element is equivalent to performing arithmetic in
    the field of rational functions over the field. The computational
    difficulty associated with such extensions is in verifying that
    proposed extensions are transcendental. When the extensions being
    considered are functions, and where a differentiation operator can
    be defined for them, structure theorems can be used to determine
    the character of the extension and to exhibit a relationship
    between the adjoined element and existing quantities in case the
    adjoined element is not transcendental.",
  paper = "Buch82.pdf"
}
```

———-

1.11 The Seven Dwarfs

— **axiom.bib** —

```
@InCollection{Kalt10a,
```

```
  author = "Kaltofen, Erich L.",
  title = {{The ``Seven Dwarfs'' of Symbolic Computation}},
  booktitle =    "Numeric and Symbolic Scientific Computing
                  Progress and Prospects",
  publisher = "Springer",
  pages = "95--104",
  year = "2010",
  keywords = "survey",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/10/Ka10_7dwarfs.pdf}",
  paper = "Kalt10a.pdf"
}
```

———-

1.12 Solving Systems of Equations

— **axiom.bib** —

```
@inproceedings{Bro86,
  author = "Bronstein, Manuel",
  title = {{Gsolve: a faster algorithm for solving systems of algebraic
           equations}},
  booktitle = "Proc of 5th ACM SYMSAC",
  year = "1986",
  pages = "247-249",
  isbn = "0-89791-199-7",
  abstract = "
    We apply the elimination property of Gr{\"o}bner bases with respect to
    pure lexicographic ordering to solve systems of algebraic equations.
    We suggest reasons for this approach to be faster than the resultant
    technique, and give examples and timings that show that it is indeed
    faster and more correct, than MACSYMA's solve."
}
```

———-

— **axiom.bib** —

```
@inproceedings{Corl95,
  author = "Corless, Robert M. and Gianni, Patrizia, M. and Trager, Barry M.
            and Watt, Stephen M.",
  title = {{The Singular Value Decomposition for Polynomial Systems}},
  booktitle = "ISSAC 95",
  year = "1995",
  pages = "195-207",
  publisher = "ACM",
  abstract =
    "This paper introduces singular value decomposition (SVD) algorithms
    for some standard polynomial computations, in the case where the
    coefficients are inexact or imperfectly known. We first give an
```

```
    algorithm for computing univariate GCD's which gives {\sl exact}
    results for interesting {\sl nearby} problems, and give efficient
    algorithms for computing precisely how nearby. We generalize this to
    multivariate GCD computations. Next, we adapt Lazard's $u$-resultant
    algorithm for the solution of overdetermined systems of polynomial
    equations to the inexact-coefficent case. We also briefly discuss an
    application of the modified Lazard's method to the location of
    singular points on approximately known projections of algebraic curves.",
  paper = "Corl95.pdf",
  keywords = "axiomref",
}
```

———-

— **axiom.bib** —

```
@article{Lixx10,
  author = "Li, Xiaoliang and Mou, Chenqi and Wang, Dongming",
  title = {{Decomposing polynomial sets into simple sets over finite fields:
           The zero-dimensional case}},
  comment = "Provides clear polynomial algorithms",
  journal = "Computers and Mathematics with Applications",
  volume = "60",
  pages = "2983-2997",
  year = "2010",
  abstract =
    "This paper presents algorithms for decomposing any zero-dimensional
    polynomial set into simple sets over an arbitrary finite field, with
    an associated ideal or zero decomposition. As a key ingredient of
    these algorithms, we generalize the squarefree decomposition approach
    for univariate polynomials over a finite field to that over the field
    product determined by a simple set. As a subprocedure of the
    generalized squarefree decomposition approach, a method is proposed to
    extract the $p$th root of any element in the field
    product. Experiments with a preliminary implementation show the
    effectiveness of our algorithms.",
  paper = "Lixx10.pdf"
}
```

———-

1.13 Numerical Algorithms

— **axiom.bib** —

```
@book{Acto70,
  author = "Acton, F.S.",
  title = {{Numerical Methods that (Usually) Work}},
  year = "1970",
  publisher = "Harper and Row",
```

```
  address = "New York, USA"
}
```

———-

— **axiom.bib** —

```
@book{Acto96,
  author = "Acton, F.S.",
  title = {{Real Computing Made Real: Preventing Errors in Scientific
           and Engineering Calculations}},
  year = "1996",
  publisher = "Princeton University Press",
  address = "Princeton, N.J. USA",
  isbn = "0-691-03663-2"
}
```

———-

— **axiom.bib** —

```
@techreport{Ahre15,
  author = "Ahrens, Peter and Nguyen, Hong Diep and Demmel, James",
  title = {{Efficient Reproducible Floating Point Summation and BLAS}},
  institution = "University of California, Berkeley",
  year = "2015",
  month = "December",
  type = "technical report",
  number = "229",
  link = "\url{http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-229.pdf}",
  abstract =
    "We define reproducibility to mean getting bitwise identical results
    from multiple runs of the same program, perhaps with different
    hardware resources or other changes that should ideally not change the
    answer. Many users depend on reproducibility for debugging or
    correctness. However, dynamic scheduling of parallel computing
    resources, combined with nonassociativity of floating point addition,
    makes attaining reproducibility a challenge even for simple operations
    like summing a vector of numbers, or more complicated operations like
    Basic Linear Algebra Subprograms (BLAS). We describe an algorithm that
    computes a reproducible sum of floating point numbers independent of
    the order of summation. The algorithm depends only on a subset of the
    IEEE Floating Point Standard 754-2008. It is communication-optimal, in
    the sense that it does just one pass over the data in the sequential
    case, or one reduction operation in the parallel case, requiring an
    ‘‘accumulator’’ represented by just 6 floating point words (more can
    be used if higher precision is desired). Th arithmetic code with a
    6-word accumulator is $7n$ floating point additions to sum $n$ words,
    and (in IEEEE double precision) the final error bound can be up to
    $10^8$ times smaller than the error bound for conventional
    summation. We describe the basic summation algorithm, the software
    infrastructure used to build reproducible BLAS (ReproBLAS), and
```

```
    performance results. For example, when computing the dot product of
    4096 double precision floating point numbers, we get a $4x$ slowdown
    compared to Intel Math Kernel Library (MKL) running on an Intel Core
    i7-2600 CPU operating at 3.4 GHz and 256 KB L2 Cache.",
  paper = "Ahre15.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Alef00,
  author = "Alefeld, G. and Mayer, G.",
  title = {{Interval analysis: Theory and applications}},
  journal = "J. Comput. Appl. Math.",
  volume = "121",
  pages = "421-464",
  year = "2000"
}
```

———-

— **axiom.bib** —

```
@article{Anda94,
  author = "Anda, A.A. and Park, H.",
  title = {{Fast plane rotations with dynamic scaling}},
  journal = "SIAM J. matrix Anal. Appl.",
  volume = "15",
  year = "1994",
  pages = "162-174"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Anda95,
  author = "Anda, Andrew Allen",
  title = {{Self-Scaling Fast Plane Rotation Algorithms}},
  school = "University of Minnesota",
  year = "1995",
  month = "February",
  abstract =
    "A suite of {\sl self-scaling} fast circular plane rotation algorithms
    is developed which obviates the monitoring and periodic rescaling
    necessitated by the standard set of fast plane rotation
    algorithms. Self-Scaling fast rotations dynamically preserve the
    normed scaling of the diagonal factor matrix whereas standard fast
    rotations exhibit divergent scaling. Variations on standard fast
    rotation matrices are developed and algorithms which implement them
```

```
    are offered. Self-Scaling fast rotations are shown to be essentially
    as accurate as slow rotations and at least as efficient as standard
    fast rotations. Computational experimental results utilizing the
    Cray-2 illustrate the effectively stable scaling exhibited by
    self-scaling fast rotations. Jacobi-class algorithms with one-sided
    alterations are developed for the algebraic eigenvalue decomposition
    using self-scaling fast plane rotations and one-sided
    modifications. The new algorithms are shown to be both accurate and
    efficient on both vector and parallel architectures. The utility is
    described of applying fast plane rotations towards the rank-one update
    and downdate of least squares factorizations. The equivalence is
    illuminated of LINPACK, hyperbolic rotation, and fast negatively
    weighted plane rotation downdating. Algorithms are presented which
    apply self-scaling fast plane rotations to the QR factorization for
    stiff least squares problems. Both fast and standard Givens
    rotation-based algorithms are shown to produce accurate results
    regardless of row sorting even with extremely heavily row weighted
    matrices. Such matrices emanate, e.g. from equality constrainted least
    squares problems solved via the weighting method. The necessity of
    column sorting is emphasized. Numerical tests expose the Householder
    QR factorization algorithm to be sensitive to row sorting and it
    yields less accurate results for greater weights. Additionally, the
    modified Gram-Schmidt algorithm is shown to be sensitive to row
    sorting to a notably significant but lesser degree. Self-Scaling fast
    plane rotation algorithms, having competitive computational
    complexities, must therefore be the method of choice for the QR
    factorization of stiff matrices. Timing reults on the Cray 2 [XY]M/P,
    and C90, of rotations both in and out of a matrix factorization
    context are presented. Architectural features that can best exploit
    the advantagous features of the new fast rotations are subsequently
    discussed.",
  paper = "Anda95.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{LAPA99,
  author = "Anderson E. et al.",
  title = {{LAPACK User's Guide Third Addition}},
  year = "1999",
  month = "August",
  link = "\url{http://www.netlib.org/lapack/lug/}"
}
```

———-

— **axiom.bib** —

```
@book{Ande99,
  author = "Anderson, E. and Bai, Z. and Bischof, S. and Blackford, S. and
```

```
            Demmel, J. and Dongarra, J. J. and DuCroz, J. and Greenbaum, A.
            and Hammarling, S. and McKenney, A. Sorensen, D. C.",
  title = {{LAPACK Users' Guide}},
  publisher = "SIAM",
  year = "1999",
  isbn = "0-89871-447-8",
  link = "\url{http://www.netlib.org/lapack/lug/}"
}
```

———-

— **axiom.bib** —

```
@misc{Baud12,
  author = "Baudin, Michael and Smith, Robert L.",
  title = {{A Robust Complex Division in Scilab}},
  link = "\url{http://arxiv.org/pdf/1210.4539.pdf}",
  year = "2012",
  month = "October",
  abstract =
    "The most widely used algorithm for floating point complex division,
    known as Smith's method, may fail more often than expected. This
    document presents two improved complex division algorithms. We present
    a proof of robustness of the first improved algorithm. Numerical
    simulations show that this algorithm performs well in practice and is
    significantly more robust than other known implementations. By
    combining additional scaling methods with this first algorithm, we
    were able to create a second algorithm, which rarely fails.",
  paper = "Baud12.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bind02,
  author = "Bindel, D. and Demmel, J. and Kahan, W. and Marques, O.",
  title = {{On computing Givens rotations reliably and efficiently}},
  journal = "ACM Trans. Math. Software",
  volume = "28",
  pages = "206-238",
  year = "2002"
}
```

———-

— **axiom.bib** —

```
@article{Blac97,
  author = "Blackford, L. S. and Cleary, A. and Demmel, J. and Dhillon, I.
            and Dongarra, J. J. and Hammarling, S. and Petitet, A. and
```

```
           Ren, H. and Stanley, K. and Whaley, R. C.",
  title = {{Practical experience in the numerical dangers of heterogeneous
           computing}},
  journal = "ACM Trans. Math. Software",
  volume = "23",
  pages = "133-147",
  year = "1997"
}
```

———-

— **axiom.bib** —

```
@techreport{MMEF96,
  author = "Boisvert, Ronald F. and Pozo, Roldan and Remington, Karin A.",
  title = {{The Matrix Market Exchange Formats: Initial Design}},
  year = "1996",
  month = "December",
  institution = "National Institute of Standards and Technology",
  type = "Technical Report",
  link = "\url{http://math.nist.gov/MatrixMarket/reports/MMformat.ps}",
  abstract =
    "We propose elementary ASCII exchange formats for matrices. Specific
    instances of the format are defined for dense and sparse matrices with
    real, complex, integer and pattern entries, with special cases for
    symmetric, skew-symmetric and Hermitian matrices. Sparse matrices are
    represented in a coordinate storage format. The overall file structure
    is designed to allow future definition of other specialized matrix
    formats, as well as for objects other than matrices.",
  paper = "MMEF96.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bran97,
  author = "Brankin, R. W. and Gladwell, I.",
  title = {{rksuite\_90: Fortran 90 software for ordinary differential
           equation initial-value problems}},
  journal = "ACM Trans. Math. Software",
  volume = "23",
  pages = "402-415",
  year = "1997"
}
```

———-

— **axiom.bib** —

```
@techreport{Bran92,
```

```
  author = "Brankin, R. W. and Gladwell, I. and Shampine, L. F.",
  title = {{RKSUITE: A suite of runge-kutta codes for the initial value
           problem for ODEs}},
  year = "1992",
  institution = "Southern Methodist University, Dept of Math.",
  number = "Softreport 92-S1",
  type = "Technical Report"
}
```

———-

— **axiom.bib** —

```
@article{Bran93,
  author = "Brankin, R. W. and Gladwell, I. and Shampine, L. F.",
  title = {{RKSUITE: A suite of explicit runge-kutta codes}},
  journal = "Contributions ot Numerical Mathematics",
  pages = "41-53",
  publisher = "World Scientific",
  year = "1993"
}
```

———-

— **axiom.bib** —

```
@book{Brit92,
  author = "Britton, J. L.",
  title = {{Collected Works of A. M. Turing: Pure Mathematics}},
  publisher = "North-Holland",
  year = "1992",
  isbn = "0-444-88059-3"
}
```

———-

— **axiom.bib** —

```
@misc{Bron99,
  author = "Bronstein, Manuel",
  title = {{Fast Deterministic Computation of Determinants of Dense Matrices}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  abstract = "
    In this paper we consider deterministic computation of the exact
    determinant of a dense matrix $M$ of integers. We present a new
    algorithm with worst case complexity
    \[O(n^4(log n+ log \verb?||M||?)+x^3 log^2 \verb?||M||?)\],
    where $n$ is the dimension of the matrix
    and \verb?||M||? is a bound on the entries in $M$, but with
    average expected complexity
    \[O(n^4+m^3(log n + log \verb?||M||?)^2)\],
```

```
    assuming some plausible properties about the distribution of $M$.
    We will also describe a practical version of the algorithm and include
    timing data to compare this algorithm with existing ones. Our result
    does not depend on ‘‘fast’’ integer or matrix techniques.",
  paper = "Bron99.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Broo11,
  author = "Brookes, Mike",
  title = {{The Matrix Reference Manual}},
  year = "2011",
  link = "\url{http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html}"

}
```

———-

— **axiom.bib** —

```
@misc{Chel06,
  author = "Chellappa, Srinivas and Franchetti, Franz and Puschel, Markus",
  title = {{How To Write Fast Numerical Code: A Small Introduction}},
  year = "2006",
  institution = "Carnegie Mellon University",
  link = "\url{https://users.ece.cmu.edu/~franzf/papers/gttse07.pdf}",
  algebra = "\newline\refto{package BLAS1 BlasLevelOne}",
  abstract =
    "The complexity of modern computing platforms has made it extremely
    difficult to write numerical code that achieves the best possible
    performance. Straightforward implementations based on algorithms that
    minimize the operations count often fall short in performance by at
    least one order of magnitude. This tutorial introduces the reader to a
    set of general techniques to improve the performance of numerical
    code, focusing on optimizations for the computer's memory hierarchy.
    Further, program generators are discussed as a way to reduce the
    implementation and optimization effort. Two running examples are
    used to demonstrate these techniques: matrix-matrix multiplication
    and the discrete Fourier transform.",
  paper = "Chel06.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Chai96,
  author = "Chaitin-Chatelin, F. and Fraysse, V.",
```

```
  title = {{Lectures on Finite Precision Computations}},
  publisher = "SIAM",
  year = "1996",
  isbn = "0-89871-358-7"
}
```

———-

— **axiom.bib** —

```
@article{Chan83,
  author = "Chan, T. F. and Golub, G. H. and LeVeque, R. J.",
  title = {{Algorithms for computing the sample variance: Analysis and
           recommendations}},
  journal = "The American Statistician",
  volume = "37",
  pages = "242-247",
  year = "1983"
}
```

———-

— **axiom.bib** —

```
@article{Cool03,
  author = "Cools, R. and Haegemans, A.",
  title = {{Algorithm 824: CUBPACK: A package for automatic cubature;
           framework description}},
  journal = "ACM Trans. Math. Software",
  volume = "29",
  pages = "287-296",
  year = "2003"
}
```

———-

— **axiom.bib** —

```
@techreport{Coxx00,
  author = "Cox, M. G. and Dainton, M. P. and Harris, P. M.",
  title = {{Testing spreadsheets and other packages used in metrology:
           Testing functions for the calculation of standard deviation}},
  year = "2000",
  institution = "National Physical Lab, Teddington, Middlesex UK",
  type = "Technical Report",
  number = "NPL Report CMSC07/00"
}
```

———-

— **axiom.bib** —

```
@article{Davi11,
  author = "Davis, Timothy A. and Hu, Yifan",
  title = {{The University of Florida Sparse Matrix Collection}},
  journal = "ACM Trans. on Math. Software",
  volume = "38",
  number = "1",
  year = "2011",
  month = "November",
  link = "\url{http://yifanhu.net/PUB/matrices.pdf}",
  abstract =
    "We describe the Univerity of Florida Sparse Matrix Collection, a large
    and actively growing set of sparse matrices that arise in real
    applications. The Collection is widely used by the numerical linear
    algebra community for the development and performance evaluation of
    sparse matrix algorithms. It allows for robust and repeatable
    experiments: robust because performance results with artificially
    generated matrices can be misleading, and repeatable because matrices
    are curated and made publicly available in many formats. Its matrices
    cover a wide spectrum of domains, including those arising from
    problems with underlying 2D or 3D geometry (as structural engineering,
    computational fluid dynamics, model reduction, electromagnetics,
    semiconductor devices, thermodynamics, materials, acoustics, computer
    graphics/vision, robitics/kinematics, and other discretizations) and
    those that typically do not have such geometry (optimization, circuit
    simulation, economic and financial modeling, theoretical and quantum
    chemistry, chemical process simulation, mathematics and statistics,
    power networks, and other networks and graphs). We provide software
    for accessing and managing the Collection, from MATLAB, Mathematica,
    Fortran, and C, as well as an online search capability. Graph
    visualization of the matrices is provided, and a new multilevel
    coarsening scheme is proposed to facilitate this task.",
  paper = "Davi11.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Davi16,
  author = "Davis, Timothy and Rajamanickam, Sivasankaran and
            Sid-Lakhdar, Wissam M.",
  title = {{A survey of direct methods for sparse linear systems}},
  year = "2016",
  month = "April",
  institution = "Texas A and M",
  type = "Technical Report",
  link = "\url{http://faculty.cse.tamu.edu/davis/publications_files/survey_tech_report.pdf}",
  abstract =
    "Wilkinson defined a sparse matrix as one with enough zeros that it
    pays to take advantage of them. This informal yet practical definition
    captures the essence of the goal of direct methods for solving sparse
    matrix problems. They exploit the sparsity of a matrix to solve
```

```
    problems economically: much faster and using far less memory than if
    all the entries of a matrix were stored and took part in explicit
    computations. These methods form the backbone of a wide range of
    problems in computational science. A glimpse of the breadth of
    applications relying on sparse solvers can be seen in the origins of
    matrices in published matrix benchmark collections. The goal of this
    survey article is to impart a working knowledge of the underlying
    theory and practice of sparse direct methods for solving linear
    systems and least-squares problems, and to provide an overview of the
    algorithms, data structures, and software available to solve these
    problems, so that the reader can both understand the methods and know
    how best to use them.",
  paper = "Davi16.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Demm88,
  author = "Demmel, James and Kahan, W.",
  title = {{Computing Small Singular Values of Bidiagonal Matrices with
           Guaranteed High Relative Accuracy}},
  year = "1988",
  institution = "New York University",
  type = "Technical Report",
  number = "326",
  abstract =
    "Computing the singular values of a bidiagonal matrix is the final
    phase of the standard algorithm for the singular value decomposition
    of a general matrix. We present a new algorithm which computes all the
    singular values of a bidiagonal matrix to high relative accuracy
    independent of their magnitudes. In contrast, the standard algorithm
    for bidiagonal matrices may compute small singular values with no
    relative accuracy at all. Numerical experiments show that the new
    algorithm is comparable in speed to the standard algorithm, and
    frequently faster. We also show how to accurately compute tiny
    eigenvalues of some classes of symmetric trigiagonal matrices using
    the same technique.",
  paper = "Demm88.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Demm05,
  author = "Demmel, James and Hida, Yozo and Kahan, W. and Li, Xiaoye S.
           and Mukherjee, Soni and Riedy, E. Jason",
  title = {{Error Bounds from Extra Precise Iterative Refinement}},
  year = "2005",
  institution = "Univerity of California, Berkeley",
```

```
  type = "Technical Report",
  number = "165",
  link = "\url{http://www.netlib.org/lapack/lawnspdf/lawn165.pdf}",
  abstract =
    "We present the design and testing of an algorithm for iterative
    refinement of the solution of linear equations, were the residual is
    computed with extra precision. This algorithm was originally proposed
    in the 1960s as a means to compute very accurate solutions to all but
    the most ill-conditioned linear systems of equations. However two
    obstacles hae until now prevented its adoption in standard subroutine
    libraries like LAPACK: (1) There was no standard way to access the
    higher precision arithmetic needed to compute residuals, and (2) it
    was unclear how to compute a reliable error bound for the computed
    solution. The completion of the new BLAS Technical Forum Standard has
    recently removed the first obstacle. To overcome the second obstacle,
    we show how a single application of iterative refindment can be used
    to copute an error bound in any norm at small cost, and use this to
    compute both an error bound in the usual infinity norm, and a
    componentwise relative error bound.

    We report extensive test results on over 6.2 million matrices of
    dimension 5, 10, 100, and 1000. As long as a normwise
    (resp. componentwise) condition number computed by the algorithm is
    less than $1/max(10,\sqrt{n})\epsilon_w$, the computed normwise
    (resp. componentwise) error bound is at most
    $2max(10,\sqrt{n})\epsilon_w$, and indeed bounds the true error. Here,
    $n$ is the matrix dimension and $\epsilon_w$ is a single precision
    roundoff error. For worse conditioned problems, we get similarly small
    correct error bounds in over 89.4\% of cases.",
  paper = "Demm05.pdf"
}
```

———-

— axiom.bib —

```
@techreport{Demm08,
  author = "Demmel, James and Hoemmen, Mark and Hida, Yozo
            and Riedy, E. Jason",
  title = {{Non-Negative Diagonals and High Performance on Low-Profile
            Matrices from Householder QR}},
  year = "2008",
  institution = "Univerity of California, Berkeley",
  type = "Technical Report",
  number = "203",
  link = "\url{http://www.netlib.org/lapack/lawnspdf/lawn203.pdf}",
  abstract =
    "The Householder reflections used in LAPACK's QR factorization leave
    positive and negative real entries along R's diagonal. This is
    sufficient for most applications of QR factorizations, but a few
    require that R have a non-negative diagonal. This note provides a new
    Householder generation routine to produce a non-negative
    diagonal. Additionally, we find that scanning for trailing zeros in
```

```
    the generated reflections leads to large performance improvements when
    applying reflections with many trailing zeros. Factoring low-profile
    matrices, those with non-zero entries mostly near the diagonal (e.g
    band matrices), now requires far fewer operations. For example, QR
    factorization of matrices with profile width $b$ that are stored
    densely in an $n \cross n$ matrix improves form $O(n^3)$
    to $O(n^2+nb^2)$.",
  paper = "Demm08.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Demm90,
  author = "Demmel, James and Kahan, W.",
  title = {{Accurate Singular Values of Bidiagonal Matrices}},
  journal = "SIAM J. Sci. Stat. Comput.",
  volume = "11",
  number = "5",
  pages = "873-912",
  year = "1990",
  link = "\url{http://www.netlib.org/lapack/lawnspdf/lawn03.pdf}",
  abstract =
    "Computing the singular values of a bidiagonal matrix is the final
    phase of the standard algorithm for the singular value decomposition
    of a general matrix. We present a new algorithm which computes all the
    singular values of a bidiagonal matrix to high relative accuracy
    independent of their magnitudes. In contrast, the standard algorithm
    for bidiagonal matrices may compute small singular values with no
    relative accuracty at all. Numerical experiments show that the new
    algorithm is comparable in speed to the standard algorithm, and
    frequently faster.",
  paper = "Demm90.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Demm92,
  author = "Demmel, James and Veselic, Kresimir",
  title = {{Jacobi's Method is More Accurate than QR}},
  journal = "SIAM J. Matrix Anal. and Appl",
  volume = "13",
  number = "4",
  year = "1992",
  pages = "1204-1245",
  link = "\url{http://www.netlib.org/lapack/lawnspdf/lawn15.pdf}",
  abstract =
    "We show that Jacobi's method (with a proper stopping criterion)
    computes small eigenvalues of symmetric positive definite matrices
```

```
    with a uniformly better relative accuracy bound than QR, divide and
    conquer, traditional bisection, or any algorithm which first involves
    tridiagonalizing the matrix. In fact, modulo an assumption based on
    extensive numerical tests, we show that Jacobi's method is optimally
    accurate in the following sense: if the matrix is such that small
    relative errors in its entries cause small relative errors in its
    eigenvalues, Jacobi will compute them with nearly this accuracy. In
    other words, as long as the initial matrix has small relative errors
    in each component, even using infinite precision will not improve on
    Jacobi (modulo factors of dimensionality). We also show the
    eigenvectors are computed more accurately by Jacobi than previously
    thought possible. We prove similar results for using one-sided Jacobi
    for the singular value decomposition of a general matrix.",
  paper = "Demm92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Rijk98,
  author = "de Rijk, P.P.",
  title = {{A One-sided Jacobi Algorithm for Computing the Singular Value
           Decomposition on a vector computer}},
  journal = "SIAM J. Sci. Stat. Comput.",
  volume = "10",
  number = "2",
  month = "March",
  year = "1989",
  pages = "359-371",
  abstract =
    "An old algorithm for computing the singular value decomposition,
    which was first mentioned by Hestenes, has gained renewed interest
    because of its properties of parallelism and vectorizability. Some
    computational modifications are given and a comparison with the
    well-known Golub-Reinsch algorithm is made. Comparative experiments on
    a CYBER 205 are reported."
}
```

———-

— **axiom.bib** —

```
@phdthesis{Dhil97,
  author = "Dhillon, Inderjit Singh",
  title = {{A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal
           Eigenvalue/Eigenvector Problem}},
  school = "University of California, Berkeley",
  year = "1997",
  link =
    "\url{http://www.eecs.berkeley.edu/Pubs/TechRpts/1997/CSD-97-971.pdf}",
  abstract =
```

```
    "Computing the eigenvalues and orthogonal eigenvectors of an $n\times n$
    symmetric tridiagonal matrix is an important task that arises while
    solving any symmetric eigenproblem. All practical software requires
    $O(n^3)$ time to compute all the eigenvectors and ensure their
    orthogonality when eigenvalues are close. In the first part of this
    thesis we review earlier work and show how some existing
    implementations of inverse iteration can fail in surprising ways.

    The main contribution of this thesis is a new $O(n^2)$, easily
    parallelizable algorithm for solving the tridiagonal
    eigenproblem. Three main advances lead to our new algorithm. A
    tridiagonal matrix is traditionally represented by its diagonal and
    off-diagonal elements. Our most important advance is in recognizing
    that its bidiagonal factors are ``better'' for computational
    purposes. The use of bidiagonals enables us to invoke a relative
    criterion to judge when eigenvalues are ``close''. The second advance
    comes with using multiple bidiagonal factorizations in order to
    compute different eigenvectors independently of each other. Thirdly,
    we use carefully chosen dqds-like transformations as inner loops to
    compute eigenpairs that are highly accurate and ``faithful'' to the
    various bidiagonal representations. Orthogonality of the eigenvectors
    is a consequence of this accuracy. Only $O(n)$ work per eigenpair is
    neede by our new algorithm.

    Conventional wisdom is that there is usually a trade-off between speed
    and accuracy in numerical procedures, i.e., higher accuracy can be
    achieved only at the expense of greater computing time. An interesting
    aspect of our work is that increased accuracy in the eigenvalues and
    eigenvectors obviates the need for explicit orthogonalization and
    leads to greater speed.

    We present timing and accuracy results comparing a computer
    implementation of our new algorithm with four existing EISPACK and
    LAPACK software routines. Our test-bed contains a variety of
    tridiagonal matrices, some coming from quantum chemistry
    applications. The numerical results demonstrate the superiority of
    our new algorithm. For example, on a matrix of order 966 that occurs in
    the modeling of a biphenyl molecule our method is about 10 times
    faster than LAPACK's inverse iteration on a serial IBM RS/6000
    processor and nearly 100 times faster on a 128 processor IBM SP2
    parallel machine.",
  paper = "Dhil97.pdf"
}
```

———-

— axiom.bib —

```
@article{Dhil04a,
  author = "Dhillon, Inderjit S. and Parlett, Beresford N.",
  title = {{Multiple representations to compute orthogonal eigenvectors
           of symmetric tridiagonal matrices}},
  journal = "Linear Algebra and its Applications",
```

```
  volume = "387",
  number ="1",
  pages = "1-28",
  year = "2004",
  month = "August",
  abstract =
    "In this paper we present an $O(nk)$ procedure, Algorithm $MR^3$, for
    computing $k$ eigenvectors of an $n\times n$ symmetric tridiagonal
    matrix $T$. A salient feature of the algorithm is that a number of
    different $LDL^t$ products ($L$ unit lower triangular, $D$ diagonal)
    are computed. In exact arithmetic each $LDL^t$ is a factorization of a
    translate of $T$. We call the various $LDL^t$ productions
    {\sl representations} (of $T$) and, roughly speaking, there is a
    representation for each cluster of close eigenvalues. The unfolding of
    the algorithm, for each matrix, is well described by a
    {\sl representation tree}. We present the tree and use it to show that if
    each representation satisfies three prescribed conditions then the
    computed eigenvectors are orthogonal to working accuracy and have
    small residual norms with respect to the original matrix $T$.",
  paper = "Dhil04a.pdf"
}
```

———-

— axiom.bib —

```
@article{Dhil04,
  author = "Dhillon, Inderjit S. and Parlett, Beresford N.",
  title = {{Orthogonal Eigenvectors and Relative Gaps}},
  journal = "SIAM Journal on Matrix Analysis and Applications",
  volume = "25",
  year = "2004",
  abstract =
    "Let $LDL^t$ be the triangular factorization of a real symmetric
    $n\times n$ tridiagonal matrix so that $L$ is a unit lower bidiagonal
    matrix, $D$ is diagonal. Let $(\lambda,\nu)$ be an eigenpair,
    $\lambda \ne 0$, with the property that both $\lambda$ and $\nu$ are
    determined to high relative accuracy by the parameters in $L$ and $D$.
    Suppose also that the relative gap between $\lambda$ and its nearest
    neighbor $\mu$ in the spectrum exceeds $1/n; n|\lambda-\mu| > |\lambda|$.

    This paper presents a new $O(n)$ algorithm and a proof that, in the
    presence of round-off errors, the algorithm computes an approximate
    eigenvector $\hat{\nu}$ that is accurate to working precision
    $|sin \angle(\nu,\hat{\nu})| = O(n\epsilon)$, where $\epsilon$ is the
    round-off unit. It follows that $\hat{\nu}$ is numerically orthogonal to
    all the other eigenvectors. This result forms part of a program to
    compute numerically orthogonal eigenvectors without resorting to the
    Gram-Schmidt process.

    The contents of this paper provide a high-level description and
    theoretical justification for LAPACK (version 3.0) subroutine DLAR1V.",
  paper = "Dhil04.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Dods83,
  author = "Dodson, D. S.",
  title = {{Corrigendum: Remark on 'Algorithm 539: Basic Linear Algebra
           Subroutines for FORTRAN usage}},
  journal = "ACM Trans. Math. Software",
  volume = "9",
  pages = "140",
  year = "1983"
}
```

———-

— **axiom.bib** —

```
@article{Dods82,
  author = "Dodson, D. S. and Grimes, R. G.",
  title = {{Remark on algorithm 539: Basic Linear Algebra Subprograms for
           Fortran usage}},
  journal = "ACM Trans. Math. Software",
  volume = "8",
  pages = "403-404",
  year = "1982"
}
```

———-

— **axiom.bib** —

```
@article{Dong88,
  author = "Dongarra, J. J. and DuCroz, J. and Hammarling, S. and
            Hanson, R. J.",
  title = {{An extended set of FORTRAN Basic Linear Algebra Subprograms}},
  journal = "ACM Trans. Math. Software",
  volume = "14",
  pages = "1-32",
  year = "1988"
}
```

———-

— **axiom.bib** —

```
@article{Dong88a,
  author = "Dongarra, J. J. and DuCroz, J. and Hammarling, S. and
            Hanson, R. J.",
```

```
  title = {{Corrigenda: 'An extended set of FORTRAN Basic Linear Algebra
           Subprograms}},
  journal = "ACM Trans. Math. Software",
  volume = "14",
  pages = "399",
  year = "1988"
}
```

———-

— **axiom.bib** —

```
@article{Dong90,
  author = "Dongarra, J. and DuCroz, J. and Duff, I. S. and Hammarling, S.",
  title = {{A set of Level 3 Basic Linear Algebra Subprograms}},
  journal = "ACM Trans. Math. Software",
  volume = "16",
  pages = "1-28",
  year = "1990"
}
```

———-

— **axiom.bib** —

```
@article{Dong16,
  author = "Dongarra, Jack",
  title = {{With Extreme Scale Computing, the Rules Have Changed}},
  journal = "LNCS",
  volume = "9725",
  pages = "3-6",
  year = "2016",
  paper = "Dong16.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Drma97,
  author = "Drmac, Zlatko",
  title = {{Implementation of Jacobi Rotations for Accurate Singular Value
           Computation in Floating Point Arithmetic}},
  journal = "SIAM Journal on Scientific Computing",
  volume = "18",
  number = "4",
  year = "1997",
  month = "July",
  pages = "1200-1222",
  abstract =
    "In this paper we consider how to compute the singular value
```

```
    decomposition (SVD) $A = U\Sigma{}T^{\tau}$ of
    $A=[a_1,a_2] \in R^{m\times 2}$
    accurately in floating point arithmetic. It is shown
    how to compute the Jacobi rotation $V$ (the right singular vector
    matrix) and how to compute $AV=U\Sigma$ even if the floating point
    representation of $V$ is the identity matrix. In the case
    underflow can produce the identity matrix as
    the floating point value of $V$ even for $a1$, $a2$ that are far from
    being mutually orthogonal. This can cause loss of accuracy and failure
    of convergence of the floating point implementation of the Jacobi
    method for computing the SVD. The modified Jacobi method recommended
    in this paper can be implemented as a reliable and highly accurate
    procedure for computing the SVD of general real matrices whenever the
    exact singular values do not exceed the underflow or overflow limits."
}
```

———-

— axiom.bib —

```
@article{Drma08a,
  author = "Drmac, Zlatko and Veselic, Kresimir",
  title = {{New fast and accurate Jacobi SVD algorithm I}},
  journal = "SIAM J. Matrix Anal. Appl.",
  volume = "35",
  number = "2",
  year = "2008",
  pages = "1322-1342",
  comment = "LAPACK Working note 169",
  link = "\url{http://www.netlib.org/lapack/lawnspdf/lawn169.pdf}",
  abstract =
    "This paper is the result of contrived efforts to break the barrier
    between numerical accuracy and run time efficiency in computing the
    fundamental decomposition of numerical linear algebra - the singular
    value decomposition (SVD) of a general dense matrix. It is an
    unfortunate fact that the numerically most accurate one-sided Jacobi
    SVD algorithm is several times slower than generally less accurate
    bidiagonalization based methods such as the QR or the divide and
    conquer algorithm. Despite its sound numerical qualities, the Jacobi
    SVD is not included in the state of the art matrix computation
    libraries and it is even considered obsolete by some leading
    researchers. Our quest for a highly accurate and efficient SVD
    algorithm has led us to a new, superior variant of the Jacobi
    algorithm. The new algorithm has inherited all good high accuracy
    properties, and it outperforms not only the best implementations of
    the one-sided Jacobi algorithm but also the QR algorithm. Moreoever,
    it seems that the potential of the new approach is yet to be fully
    exploited.",
  paper = "Drma08a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Drma08b,
  author = "Drmac, Zlatko and Veselic, Kresimir",
  title = {{New fast and accurate Jacobi SVD algorithm II}},
  journal = "SIAM J. Matrix Anal. Appl.",
  volume = "35",
  number = "2",
  year = "2008",
  pages = "1343-1362",
  comment = "LAPACK Working note 170",
  link = "\url{http://www.netlib.org/lapack/lawnspdf/lawn170.pdf}",
  abstract =
    "This paper presents new implementation of one-sided Jacobi SVD for
    triangular matrices and its use as the core routine in a new
    preconditioned Jacobi SVD algorithm, recently proposed by the
    authors. New pivot strategy exploits the triangular form and uses the
    fact that the input triangular matrix is the result of rank revealing
    QR factorization. If used in the preconditioned Jacobi SVD algorithm,
    described in the first part of this report, it delivers superior
    performance leading to the currently fastest method for computing SVD
    decomposition with high relative accuracy. Furthermore, the efficiency
    of the new algorithm is comparable to the less accurate
    bidiagonalization based methods. The paper also discusses underflow
    issues in floating point implementation, and shows how to use
    perturbation theory to fix the imperfectness of machine arithmetic on
    some systems.",
  paper = "Drma08b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Drma08c,
  author = "Drmac, Zlatko and Bujanovic, Zvonimir",
  title = {{On the failure of rank-revealing QR factorization software -
           a case study.}},
  journal = "ACM Trans. math. Softw.",
  volume = "35",
  number = "2",
  year = "2008",
  pages = "1-28",
  comment = "LAPACK Working note 176",
  link = "\url{http://www.netlib.org/lapack/lawnspdf/lawn176.pdf}",
  abstract =
    "This note reports an unexpected and rather erratic behavior of the
    LAPACK software implementation of the QR factorization with
    Businger-Golub column pivoting. It is shown that, due to finite
    precision arithmetic, software implementation of the factorization can
    catastrophically fail to produce triangular factor with the structure
    characteristic to the Businger-Golub pivot strategy. The failure of
    current {\sl state of the art} software, and a proposed alternative
```

```
    implementations are analyzed in detail.",
  paper = "Drma08c.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dubr83,
  author = "Dubrulle, A. A.",
  title = {{A class of numerical methods for the computation of Pythagorean
           sums}},
  journal = "IBM J. Res. Develop.",
  volume = "27",
  number = "6",
  pages = "582-589",
  year = "1983"
}
```

———-

— **axiom.bib** —

```
@book{Eina05,
  author = "Einarsson, B.",
  title = {{Accuracy and Reliability in Scientific Computing}},
  publisher = "SIAM",
  year = "2005",
  isbn = "0-89871-584-9",
  link = "\url{http://www.nsc.liu.se/wg25/book/}"
}
```

———-

— **axiom.bib** —

```
@article{Elmr00,
  author = "Elmroth, E. and Gustavson, F. G.",
  title = {{Applying recursion to serial and parallel QR factorization leads
           to better performance}},
  journal = "IBM Journal of Research and Development",
  volume = "44",
  number = "4",
  month = "July",
  year = "2000",
  pages = "605--624",
  doi = "10.1.1.33.1820",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.1820}",
  abstract =
    "We present new recursive serial and parallel algorithms for QR
    factorization of an $m$ by $n$ matrix. They improve performance. The
```

```
    recursion leads to an automatic variable blocking, and it also
    replaces a Level 2 part in a standard aglorithm with Level 3
    operations. However, there are significant additional costs for
    creating and performing the updates, which prohibit the efficient use
    of the recursion for large $n$. We present a quantitative analysis of
    these extra costs. This analysis leads us to introduce a hybrid
    recursive algorithm that outperforms the LAPACK algorithm DGEQRF by
    about 20\% for large square matrices up to almost a factor of 3 for
    tall thin matrices. Uniprocessor performance results are presented for
    two IBM RS/6000 SP nodes -- a 120-Mhz IBM POWER2 node and one
    processor of a four-way 332-Mhz IBM PowerPC 604e SMP node. The hybrid
    recursive algorithm reaches more than 90\% of the theoretical peak
    performance of the POWER2 node. Compared to standard block algorithms,
    the recursive approach also shows a significant advantage in the
    automatic tuning obtained from its automatic variable blocking. A
    successful parallel implementation on a four-way 332-MHz IBM PPC604e
    SMP node based on dynamic load balancing is presented. For two, three,
    and four processors it shows speedups of up to 1.97, 299, and 3.97."
}
```

———-

— axiom.bib —

```
@misc{Fate13,
  author = "Fateman, Richard J.",
  title = {{Interval Arithmetic, Extended Numbers and Computer Algebra
           Systems}},
  year = "2013",
  link = "\url{http://www.cs.berkeley.edu/~fateman/papers/interval.pdf}",
  abstract =
    "Many ambitious computer algebra systems were initially designed in a
    flush of enthusiasm, with the goal of automating any symbolic
    mathematical manipulation ``correctly''. Historically, this approach
    results in programs that implicitly used certain identities to
    simplify expressions. These identities, which very likely seemed
    universally true to the programmers in the heat of writing the CAS
    (and often were true in well-known abstract algebraic domains) later
    neede re-examination when such systems were extended for dealing with
    kinds of objects unanticipated in the original design. These new
    objects are generally introduced to the CAS by extending
    ``generically'' the arithmetic of other operations. For example,
    approximate floats do not have the mathematical properties of exact
    integers or rationals. Complex numbers may strain a system designed
    for real-valued variables. In the situation examined here, we consider
    two categories of ``extended'' numbers: $\infty$ and {\sl undefined},
    and real intervals. We comment on issues raised by these two
    troublesome notions, how their introduction into a computer algebra
    system may require a (sometimes painful) reconsideration and redesign
    of parts of the program, and how they are related. An alternative
    (followed most notably by the Axiom system is to essentially envision
    a ``meta'' CAS defined in terms of categories and inheritance with
    only the most fundamental built-in concepts; from these one can build
```

```
    many variants of specific CAS features. This approach is appealing but
    can fail to accommodate extensions that violate some mathematical
    tenets in the cause of practicality.",
  paper = "Fate13.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Fern92,
  author = "Fernando, K. Vince and Parlett, Beresford N.",
  title = {{Accurate Singular Values and Differential qd Algorithms}},
  year = "1992",
  institution = "University of California, Berkeley",
  type = "Technical Report",
  number = "PAM-554",
  link = "\url{http://www.dtic.mil/dtic/tr/fulltext/u2/a256582.pdf}",
  abstract =
    "We have discovered a new implementation of the qd algorithm that has
    a far wider domain of stability than Rutishauser's version. Our
    algorithm was developed from an examination of the LR-Cholesky
    transformation and can be adapted to parallel computation in stark
    contrast to traditional qd. Our algorithm also yields useful a
    posteriori upper and lower bounds on the smallest singular value of a
    bidiagonal matrix.

    The zero-shift bidiagonal QR of Demmel and Kahan computes the smallest
    singlar values to maximal relative accuracy and the others to maximal
    absolute accuracy with little or no degradation in efficiency when
    compared with the LINPACK code. Our algorithm obtains maximal relative
    accuracy for all the singluar values and runs at least four times
    faster than the LINPACK code.",
  paper = "Fern92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Fors70,
  author = "Forsythe, G. E.",
  title = {{Pitfalls in computations, or why a math book isn't enough}},
  journal = "Amer. Math. Monthly",
  volume = "9",
  pages = "931-995",
  year = "1970"
}
```

———-

— **axiom.bib** —

```
@incollection{Fors69,
  author = "Forsythe, G. E.",
  title = {{What is a satisfactory quadratic equation solver}},
  booktitle = "Constructive Aspects of the Fundamental Theorem of Algebra",
  pages = "53-61",
  publisher = "Wiley",
  year = "1969"
}
```

———-

— **axiom.bib** —

```
@article{Foxx71,
  author = "Fox, L.",
  title = {{How to get meaningless answers in scientific computations (and
           what to do about it)}},
  journal = "IMA Bulletin",
  volume = "7",
  pages = "296-302",
  year = "1971"
}
```

———-

— **axiom.bib** —

```
@article{Gent74,
  author = "Gentlman, W. Morven and Marovich, Scott B.",
  title = {{More on algorithms that reveal properties of floating point
           arithmetic units.}},
  journal = "Comm. of the ACM",
  year = "1974",
  month = "April",
  volume = "17",
  number = "5",
  pages = "276-277",
  abstract =
    "In the interests of producing portable mathematical software, it is
    highly desirable for a program to be able directly to obtain
    fundamental properties of the environment in which it is to run. The
    installer would then not be obliged to change appropriate magic
    constants in the source code, and the user would not have to provide
    information he may very well not understand. Until the standard
    definitions of programming languages are changed to require builtin
    functions that provide this information, we will have to resort to
    writing routines that discover it."
}
```

———-

— **axiom.bib** —

```
@techreport{Give54,
  author = "Givens, W.",
  title = {{Numerical computation of the characteristic values of a real
           symmetric matrix}},
  year = "1954",
  institution = "Oak Ridge National Laboratory",
  type = "Technical Report",
  number = "ORNL-1574"
}
```

———-

— **axiom.bib** —

```
@article{Golu65,
  author = "Golub, G.H.",
  title = {{Numerical methods for solving linear least squares problems}},
  journal = "Numer. Math.",
  volume = "7",
  pages = "206-216",
  year = "1965"
}
```

———-

— **axiom.bib** —

```
@book{Golu89,
  author = "Golub, Gene H. and Van Loan, Charles F.",
  title = {{Matrix Computations}},
  publisher = "Johns Hopkins University Press",
  year = "1989",
  isbn = "0-8018-3772-3"
}
```

———-

— **axiom.bib** —

```
@book{Golu96,
  author = "Golub, Gene H. and Van Loan, Charles F.",
  title = {{Matrix Computations}},
  publisher = "Johns Hopkins University Press",
  isbn = "978-0-8018-5414-9",
  year = "1996"
}
```

———-

— **axiom.bib** —

```
@article{Hamm85,
  author = "Hammarling S.",
  title = {{The Singular Value Decomposition in Multivariate Statistics}},
  journal = "ACM Signum Newsletter",
  volume = "20",
  number = "3",
  pages = "2--25",
  year = "1985"
}
```

———-

— **axiom.bib** —

```
@book{Hamm05,
  author = "Hammarling, Sven",
  title = {{An Introduction to the Quality of Computed Solutions}},
  booktitle = "Accuracy and Reliability in Scientific Computing",
  year = "2005",
  publisher = "SIAM",
  pages = "43-76",
  link = "\url{http://eprints.ma.man.ac.uk/101/}",
  paper = "Hamm05.pdf"
}
```

———-

— **axiom.bib** —

```
@mastersthesis{Harg02,
  author = "Hargreaves, G.",
  title = {{Interval analysis in MATLAB}},
  school = "University of Manchester, Dept. of Mathematics",
  year = "2002"
}
```

———-

— **axiom.bib** —

```
@book{High05,
  author = "Higham, D. J. and Higham, N. J.",
  title = {{MATLAB Guide}},
  publisher = "SIAM",
  year = "2002",
  isbn = "0-89871-521-0"
}
```

———-

— **axiom.bib** —

```
@article{High88,
  author = "Higham, Nicholas J.",
  title = {{FORTRAN codes for estimating the one-norm of a real or complex
           matrix, with applications to condition estimation}},
  journal = "ACM Trans. Math. Soft",
  volume = "14",
  number = "4",
  pages = "381-396",
  year = "1988"
}
```

———-

— **axiom.bib** —

```
@misc{High98,
  author = "Higham, Nicholas J.",
  title = {{Can you 'count' on your computer?}},
  link = "\url{http://www.maths.man.ac.uk/~higham/talks/}",
  year = "1998"
}
```

———-

— **axiom.bib** —

```
@book{High02,
  author = "Higham, Nicholas J.",
  title = {{Accuracy and stability of numerical algorithms}},
  publisher = "SIAM",
  isbn = "0-89871-521-0",
  year = "2002"
}
```

———-

— **axiom.bib** —

```
@article{High86,
  author = "Higham, Nicholas J.",
  title = {{Efficient Algorithms for Computing the Condition Number of a
           Tridiagonal Matrix}},
  journal = "SIAM J. Sci. Stat. Comput.",
  volume = "7",
  number = "1",
  year = "1986",
```

```
  month = "January",
  abstract =
    "Let $A$ be a tridiagonal matrix of order $n$. We show that it is
    possible to compute $||A^{-1}||_{\infty}$ and hence
    ${\rm cond}_{\infty}(A)$ in $O(n)$ operations. Several algorithms
    which perform this task are given and their numerical properties are
    investigated.

    If $A$ is also positive definite then $||A^{-1}||_{\infty}$ can be
    computed as the norm of the solution to a positive definite
    tridiagonal linear system whose coefficient matrix is closely related
    to $A$. We show how this computation can be carried out in parallel
    with the solution of a linear system $Ax=b$. In particular we describe
    some simple modifications to the LINPACK routine SPTSL which enable
    this routine to compute ${\rm cond}_1(A)$, efficiently, in addtion to
    solving $Ax=b$.",
  paper = "High86.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{IEEE85,
  author = "IEEE",
  title = {{ANSI/IEEE Standard for Binary Floating Point Arithmetic:
           Std 754-1985}},
  publisher = "IEEE Press",
  year = "1985"
}
```

———-

— **axiom.bib** —

```
@misc{IEEE87,
  author = "IEEE",
  title = {{ANSI/IEEE Standard for Radix Independent Floating Point Arithmetic:
           Std 854-1987}},
  publisher = "IEEE Press",
  year = "1987"
}
```

———-

— **axiom.bib** —

```
@book{Isaa94,
  author = "Isaacson, E. and Keller, H. B.",
  title = {{Analysis of Numerical Methods}},
  publisher = "Dover",
```

```
  year = "1994",
  isbn = "0-486-68029-0"
}
```

———-

— **axiom.bib** —

```
@article{Kags89,
  author = "Kagstrom, Bo and Westin, L.",
  title = {{Generalized Schur Methods with Condition Estimators for Solving
           the Generalized Sylvester Equation}},
  journal = "IEEE Transactions on Automatic Control",
  volume = "34",
  number = "7",
  year = "1989",
  month = "July",
  pages = "745-751",
  abstract =
    "Stable algorithms are presented for solving the generalized Sylvester
    equation. They are based on orthogonal equivalence transformations of
    the original problem. Perturbation theory and rounding error analysis
    are included. Condition estimators (${\rm Dif}^{-1}$-estimators) are
    developed which when substituted into derived error bounds give
    accuracy estimates of a computed solution. Results from numerical
    experiments on well-conditioned and ill-conditioned problems are
    reported."
}
```

———-

— **axiom.bib** —

```
@book{Kags93a,
  author = "Kagstrom, B.",
  title = {{A Direct Method for Reordering Eigenvalues in the
           Generalized Real Schur Form of a Regular Matrix Pair (A, B)}},
  year = "1993",
  pages = "195-218",
  booktitle = "Linear Algebra for Large Scale and Real-Time Applications",
  publisher = "NATO",
  volume = "232",
  journal = "NATO ASI Series",
  institution = "NATO ASI Series",
  isbn = "978-90-481-4246-0",
  abstract =
    "A direct orthogonal equivalence transformation method for reordering
    the eigenvalues along the diagonal in the generalized real Schur form
    of a regular matrix pair $(A,B)$ is presented. Each swap of two
    adjacent eigenvalues (real, or complex conjugate pairs) involves
    solving a generalized Sylvester equation and the construction of two
    orthogonal transformation matrices from certain eigenspaces associated
```

```
    with the corresponding diagonal blocks. An error analysis of the
    direct reordering method is presented. Results from numerical
    experiments on well-conditionsed as well as ill-conditioned problems
    illustrate the stability and the accuracy of the method. Finally, a
    direct reordering algorithm with controlled backward error is described."
}
```

———-

— **axiom.bib** —

```
@techreport{Kags93,
  author = "Kagstrom, Bo and Poromaa, Peter",
  title = {{LAPACK-Style Algorithms and Software for Solving the Generalized
           Sylvester Equation and Estimating the Separation between
           Regular Matrix Pairs}},
  year = "1993",
  month = "December",
  link = "\url{http://www.netlib.org/lapack/lawnspdf/lawn75.pdf}",
  comment = "LAPACK Working Note 75",
  institution = "NETLIB",
  abstract =
    "Level 3 algorithms for solving the generalized Sylvester equation
    $(AR-LB,DR-LE)=(C,F)$ and the transposed analogue
    $(A^TU+D^TV,-UB^T-VE^T)=(C,F)$ are presented. These blocked algorithms
    permit reuse of data in complex memory hierarchies of current advanced
    computer architectures. The separation of two regular matrix pairs
    $(A,D)$ and $(B,E)$, Dif[$(A,D)$,$(B,E)$], is defined in terms of the
    generalized Sylvester operator $(AR-LB,DR-LE)$. Robust, efficient and
    reliable Dif-estimators are presented. The basic problem is to find a
    lower bound on Dif$^{-1}$, which can be done by solving generalized
    Sylvester equations in triangular form. Frobenius norm-based and one
    norm based Dif estimators are described and evaluated. These estimates
    lead to computable error bounds for the generalized Sylvester
    equation. The one-norm-based estimator makes the condition estimation
    uniform with LAPACK. Fortran 77 software that implements our
    algorithms for solving generalized Sylvester equations, and for
    computing error bounds and Dif-estimators are presented. Computational
    experiments that illustrate the accuracy, efficiency and reliability
    of our software are also described.",
  paper = "Kags93.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Kags94,
  author = "Kagstrom, Bo and Poromaa, Peter",
  title = {{Computing Eigenspaces with Specified Eigenvalues of a Regular
           Matrix Pair (A, B) and Condition Estimation: Theory,
           Algorithms and Software}},
```

```
  year = "1994",
  month = "April",
  link = "\url{http://www.netlib.org/lapack/lawns/lawn87.ps}",
  abstract =
    "Theory, algorithms and LAPACK style software for computing a pair of
    deflating subspaces with specified eigenvalues of a regular matrix
    pair $(A,B)$ and error bounds for computed quantities (eigenvalues and
    eigenspaces) are presented. The {\sl reordering} of specified
    eigenvalues is performed with a direct orthogonal transformation
    method with guaranteed numerical stability. Each swap of two adjacent
    diagonal blocks in the real generalized Schur form, where at least one
    of them corresponds to a complex conjugate pair of eigenvalues,
    involves solving a generalized Sylvester equation and the construction
    of two orthogonal transformation matrices from certain eigenspaces
    associated with the diagonal blocks. The swapping of two $1\cross 1$
    blocks is performed using orthogonal (unitary) Givens rotations. The
    {\sl error bounds} are based on estimates of condition numbers for
    eigenvalues and eigenspaces. The software computes reciprocal values
    of a condition number for an individual eigenvalue (or a cluster of
    eigenvalues), a condition number for an eigenvector (or eigenspace),
    and spectral projectors onto a selected cluster. By computing
    reciprocal values we avoid overflow. Changes in eigenvectors and
    eigenspaces are measured by their change in angle. The condition
    numbers yield both {\sl asymptotic} and {\sl global} error bounds. The
    asymptotic bounds are only accurate for small perturbations $(E,F)$ of
    $(A,B)$, while the global bounds work for all $||(E,F)||$ up to a
    certain bound, whose size is determined by the conditioning of the
    problem. It is also shown how these upper bounds can be
    estimated. Fortran 77 {\sl software} that implements our algorithms
    for reordering eigenvalues, computing (left and right) deflating
    subspaces with specified eigenvalues and condition number estimation
    are presented. Computational experiments that illustrate the accuracy,
    efficiency and reliability of our software are also described.",
  paper = "Kags94.pdf"
}
```

———-

— axiom.bib —

```
@article{Kags94b,
  author = "Kagstrom, Bo",
  title = {{A Perturbation Analysis of the Generalized Sylvester Equation
           $(AR-LB,DR-LE)=(C,F)$}},
  journal = "SIAM J. Matrix Anal. and Appl.",
  volume = "15",
  number = "4",
  pages = "1045-1060",
  year = "1994",
  abstract =
    "Perturbation and error bounds for the generalized Sylvester equation
    $(AR-LB,DR-LE)=(C,F)$ are presented. An explicit expression for the
    normwise relative backward error associated with an approximate
```

```
    solution of the generalized Sylvester equation is derived and
    conditions when it can be much greater than the relative residual are
    given. This analysis is applicable to any method that solves the
    generalized Sylvester equation. A condition number that reflects the
    structure of the problem and a normwise forward error bound based on
    ${\rm Dif}^{-1}[(A,D),B,E)]$ and the residual are derived. The
    structure-preserving condition number can be arbitrarily smaller than
    a ${\rm Dif}^{-1}$-based condition number. The normwise error bound
    can be evaluated robustly and at moderate cost by using a reliable
    ${\rm Dif}^{-1}$ estimator. A componentwise LAPACK-style forward error
    bound that can be stronger than the normwise error bound is
    presented. A componentwise approximate error bound that can be
    evaluated to a much lower cost is also proposed. Finally, some
    computational experiments that validate and evaluate the perturbation
    and error bounds are presented."
}
```

———-

— **axiom.bib** —

```
@techreport{Kaha66,
  author = "Kahan, W.",
  title = {{Accurate Eigenvalues of a Symmetric Tri-Diagonal Matrix}},
  year = "1966",
  month = "July",
  institution = "Stanford University",
  type = "Technical Report",
  number = "CS41",
  abstract =
    "Having established tight bounds for the quotient of two different
    lub-norms of the same tri-diagonal matrix J, the author observes that
    these bounds could be of use in an error-analysis provided a suitable
    algorithm were found. Such an algorithm is exhibited, and its errors
    are thoroughly accounted for, including the effects of scaling,
    over/underflow and roundoff. A typical result is that, on a computer
    using rounded floating point binary arithmetic, the biggest eigenvalue
    of J can be computed easily to within 2.5 units in its last place, and
    the smaller eigenvalues will suffer absolute errors which are no
    larger. These results are somewhat stronger than had been known before.",
  paper = "Kaha66.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Kels00,
  author = "Kelsey, Tom",
  title = {{Exact Numerical Computation via Symbolic Computation}},
  link = "\url{http://tom.host.cs.st-andrews.ac.uk/pub/ccapaper.pdf}",
  year = "2000",
```

```
  abstract = "
    We provide a method for converting any symbolic algebraic expression
    that can be converted into a floating point number into an exact
    numeric representation. We use this method to demonstrate a suite of
    procedures for the representation of, and arithmetic over, exact real
    numbers in the Maple computer algebra system. Exact reals are
    represented by potentially infinite lists of binary digits, and
    interpreted as sums of negative powers of the golden ratio.",
  paper = "Kels00.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@article{Knus98,
  author = {Kn\"usel, L.},
  title = {{On the accuracy of statistical distributions in Microsoft
           Excel 97}},
  journal = "Comput. Statist. Data Anal.",
  volume = "26",
  pages = "375-377",
  year = "1998"
}
```

———-

— **axiom.bib** —

```
@misc{Kohl14,
  author = {K\"ohler, Martin and Saak, Jens},
  title = {{On BLAS Level-3 Implementations of Common Solvers for
           (Quasi-)Triangular Generalized Lyapunov Equations}},
  year = "2014",
  link = "\url{http://slicot.org/objects/software/reports/SLWN2014_1.pdf}",
  note = "SLICOT Working Note 2014-1",
  abstract =
    "The solutions of Lyapunov and generalized Lyapunov equations are a
    key player in many applications in systems and control theory. Their
    stable numerical computation, when the full solution is sought, is
    considered solved since the seminal work of Bartels and Stewart. A
    number of variants of their algorithm have been proposed, but none of
    them goes beyond BLAS level-2 style implementation. On modern
    computers, however, the formulation of BLAS level-3 type
    implementations is crucial to enable optimal usage of cache
    hierarchies and modern block scheduling methods based on directed
    acyclic graphs describing the interdependence of single block
    computations. Our contribution closes this gap by a transformation of
    the aforementioned level-2 variants to level-3 versions and a
    comparison on a standard multicore machine.",
  paper = "Kohl14.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Krei05,
  author = "Kreinovich, V.",
  title = {{Interval cmoputations}},
  year = "2005",
  link = "\url{http://www.cs.utep.edu/interval-comp/}"
}
```

———-

— **axiom.bib** —

```
@article{Kuki72a,
 author = "Kuki, Hirondo",
 title = {{Complex Gamma Function with Error Control}},
 year = "1972",
 publisher = "ACM",
 journal = "Communications of the ACM",
 volume = "15",
 number = "4",
 pages = "262-267",
 abstract =
  "An algorithm to compute the gamma function and the loggamma function
   of a complex variable is presented. The standard algorithm is modified
   in several respects to insure the continuity of the function value and
   to reduce accumulation of round-off errors. In addition to computation
   of function values, this algorithm includes an object-time estimation
   of round-off errors. Experimental data with regard to the effectiveness
   of this error control are presented. A Fortran program for the algorithm
   appears in the algorithms section of this issue."
}
```

———-

— **axiom.bib** —

```
@article{Kuki72b,
 author = "Kuki, Hirondo",
 title = {{Algorithm 421: Complex Gamma Function with Error Control}},
 year = "1972",
 publisher = "ACM",
 journal = "Communications of the ACM",
 volume = "15",
 number = "4",
 pages = "271-272",
 abstract =
```

```
  "This Fortran program computes either the gamma function or the
   loggamma function of a complex variable in double precision.
   In addition, it provides an error estimate of the computed answer.
   The calling sequences are:
   \verb|CALL CDLGAM (X, W, E, 0)|
   for the loggamma, and
   \verb|CALL CDLGAM (X, W, E, 1)|
   for the gamma, where Z is the double precision argument, W is the
   answer of the same type, and E is a single precision real variable.
   Before the call, the value of E is an estimate of the error in Z,
   and after the call, it is an estimate of the error in W."
}
```

———-

— **axiom.bib** —

```
@book{Laws74,
  author = "Lawson, C. L. and Hanson, R. J.",
  title = {{Solving Least Squares Problems}},
  publisher = "Prentice-Hall",
  year = "1974"
}
```

———-

— **axiom.bib** —

```
@book{Laws95,
  author = "Lawson, C. L. and Hanson, R. J.",
  title = {{Solving Least Squares Problems}},
  publisher = "SIAM",
  isbn = "0-89871-356-0",
  year = "1995"
}
```

———-

— **axiom.bib** —

```
@article{Livn04,
  author = "Livne, Oren E. and Golub, Gene H.",
  title = {{Scaling by Binormalization}},
  journal = "Numerical Algorithms",
  volume = "35",
  number = "1",
  pages = "97-120",
  year = "2004",
  month = "January",
  abstract =
    "We present an interative algorithm (BIN) for scaling all the rows and
```

```
    columns of a real symmetric matrix to unit 2-norm. We study the
    theoretical convergence properties and its relation to optimal
    conditioning. Numerical experiments show that BIN requires 2-4
    matrix-vector multiplications to obtain an adequate scaling, and in
    many cases significantly reduces the condition number, more than other
    scaling algorithms. We present generalizations to complex,
    non-symmetric and rectangular matrices.",
  paper = "Livn04.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Malc72,
  author = "Malcolm, Michael A.",
  title = {{Algorithms to reveal properties of floating-point arithmetic}},
  journal = "Comms of the ACM",
  volume = "15",
  year = "1972",
  pages = "949-951",
  link = "\url{http://www.dtic.mil/dtic/tr/fulltext/u2/727104.pdf}",
  abstract =
    "Two algorithms are presented in the form of Fortran subroutines. Each
    subroutine computes the radix and number of digits of the floating
    point numbers and whether rounding or chopping is done by the machine
    on which it is run. The methods are shown to work on any ‘‘reasonable’’
    floating-point computer.",
  paper = "Malc72.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Marq06,
  author = "Marques, Osni A. and Reidy, Jason and Vomel, Christof",
  title = {{Benefits of IEEE-754 Features in Modern Symmetric Tridiagonal
           Eigensolvers}},
  journal = "SIAM Journal on Scientific Computing",
  volume = "28",
  number = "5",
  year = "2006",
  link = "\url{http://www.netlib.org/lapack/lawnspdf/lawn172.pdf}",
  abstract =
    "Bisection is one of the most common methods used to compute the
    eigenvalues of symmetric tridiagonal matrices. Bisection relies on the
    {\sl Sturm count}: for a given shift $\sigma$, the number of negative
    pivots in the factorization $T-\sigma{}I=LDL^T$ equals the number of
    eigenvalues of $T$ that are smaller than $\sigma$. In IEEE-754
    arithmetic, the value $\infty$ permits the computation to continue
    past a zero pivot, producing a correct Sturm count when $T$ is
```

```
    unreduced. Demmel and Li showed in the 90s that using $\infty$ rather
    than testing for zero pivots within the loop could improve performance
    significantly on certain architectures.

    When eigenvalues are to be computed to high relative accuracy, it is
    often preferable to work with $LDL^T$ factorizations instead of the
    original tridiagonal $T$, see for example the MRRR algorithm. In these
    cases, the Sturm count has to be computed from $LDL^T$. The
    differential stationary and progressive qds algorithms are the method
    of choice.

    While it seems trivial to replace $T$ by $LDL^T$, in reality these
    algorithms are more complicated: in IEEE-754 arithmetic, a zero pivot
    produces an overflow, followed by an invalid exception (NaN), that
    renders the Sturm count incorrect.

    We present alternative, safe formulations that are guaranteed to
    produce the correct result.

    Benchmarking these algorithms on a variety of platforms shows that the
    original formulation without tests is always faster provided no
    exception occurs. The transforms see speed-ups of up to 2.6x over the
    careful formulation.

    Tests on industrial matrices show that encountering exceptions in
    practice is rare. This leads to the following design: First, compute
    the Sturm count by the fast but unsafe algorithm. Then, if an
    exception occured, recompute the count by a safe, slower alternative.

    The new Sturm count algorithms improve the speed of bisection by up to
    2x on our test matrices. Furthermore, unlike the traditional
    tiny-pivot substitutions, proper use of IEEE-754 features provides a
    careful formulation that imposed no input range restrictions.",
  paper = "Marq06.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mart68,
  author = "Martin, R. S. and Wilkinson, J. H.",
  title = {{Similarity reduction ofa general matrix to Hessenberg form}},
  journal = "Numer. Math.",
  volume = "12",
  pages = "349-368",
  year = "1968"
}
```

———-

— **axiom.bib** —

```
@misc{Math05,
  author = "MathWorks",
  title = {{MATLAB}},
  publisher = "The Mathworks, Inc.",
  link = "\url{http://www.mathworks.com}"
}
```

———-

— **axiom.bib** —

```
@article{Mccu02,
  author = "McCullough, B. D. and Wilson, B.",
  title = {{On the accuracy of statistical procedures in Microsoft Excel
           2000 and Excel XP}},
  journal = "Comput. Statist. Data Anal.",
  volume = "40",
  pages = "713-721",
  year = "2002"
}
```

———-

— **axiom.bib** —

```
@article{Mccu99,
  author = "McCullough, B. D. and Wilson, B.",
  title = {{On the accuracy of statistical procedures in Microsoft Excel 97}},
  journal = "Comput. Statist. Data Anal.",
  volume = "31",
  pages = "27-37",
  year = "1999"
}
```

———-

— **axiom.bib** —

```
@book{Metc96,
  author = "Metcalf, M. and Reid, J. K.",
  title = {{Fortran 90/95 Explained}},
  publisher = "Oxford University Press",
  year = "1996"
}
```

———-

— **axiom.bib** —

```
@book{Metc04,
  author = "Metcalf, M. and Reid, J. K. and Cohen, M.",
```

```
  title = {{Fortran 95/2003 Explained}},
  publisher = "Oxford University Press",
  year = "2004",
  isbn = "0-19-852693-8"
}
```

———-

— **axiom.bib** —

```
@article{Mole83,
  author = "Moler, C. and Morrison, D.",
  title = {{Replacing square roots by Pythagorena sums}},
  journal = "IBM J. Res. Develop.",
  volume = "27",
  number = "6",
  pages = "577-581",
  year = "1983"
}
```

———-

— **axiom.bib** —

```
@article{Mole71,
  author = "Moler, C.B. and Stewart, G.W.",
  title = {{An Algorithm for Generalized Matrix Eigenvalue Problems}},
  journal = "SIAM J. Numer. Anal",
  volume = "10",
  year = "1973",
  pages = "241--256",
  abstract =
    "A new method, called the QZ algorithm, is presented for the solution
    of the matrix eigenvalue problem $Ax=\lambda{}Bx$ with general square
    matrices $A$ and $B$. Particluar attention is paid to the degeracies
    which result when $B$ is singular. No inversions of $B$ or its
    submatrices are used. The algorithm is a generalization of the QR
    algorithm, and reduces to it when $B=I$. A Fortran program and some
    illustrative examples are included."
}
```

———-

— **axiom.bib** —

```
@book{Moon93,
  author = "Moonen, Marc S. and Golub, Gene H. and De Moor, Bart L.R.",
  title = {{Linear Algebra and Large Scale and Real-Time Applications}},
  year = "1993",
  publisher = "NATO ASI Series",
  isbn = "978-904814246-0"
```

```
}
```

———-

— **axiom.bib** —

```
@book{Moor79,
  author = "Moore, R. E.",
  title = {{methods and Applications of Interval Analysis}},
  publisher = "SIAM",
  year = "1979"
}
```

———-

— **axiom.bib** —

```
@misc{NAGa05,
  author = "Numerical Algorithms Group",
  title = {{The NAG Library}},
  link = "\url{http://www.nag.co.uk/numeric}",
  year = "2005"
}
```

———-

— **axiom.bib** —

```
@misc{NAGb05,
  author = "Numerical Algorithms Group",
  title = {{The NAG Fortran Library Manual}},
  link = "\url{http://www.nag.co.uk/numeric/fl/manual/html/FLlibrarymanual.asp}",
  year = "2005"
}
```

———-

— **axiom.bib** —

```
@book{Over01,
  author = "Overton, M. L.",
  title = {{Numerical Computing with IEEE Floating Point Arithmetic}},
  publisher = "SIAM",
  year = "2001",
  isbn = "0-89871-482-6"
}
```

———-

,

— **axiom.bib** —

```
@book{Pies83,
  author = {Piessens, R. and de Doncker-Kapenga, E. and \"Uberhuber, C. W.
            and Kahaner, D. K.},
  title = {{QUADPACK - A Subroutine Package for Automatic Integration}},
  publisher = "Springer-Verlag",
  year = "1983"
}
```

———-

— **axiom.bib** —

```
@misc{Pete12,
  author = "Petersen, Kaare Brandt and Pedersen, Michael Syskind",
  title = {{The Matrix Cookbook}},
  link = "\url{http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf}",
  year = "2012",
  month = "November"
}
```

———-

— **axiom.bib** —

```
@article{Prie04,
  author = "Priest, D. M.",
  title = {{Efficient scaling for complex division}},
  journal = "ACM Trans. Math. Software",
  volume = "30",
  pages = "389-401",
  year = "2004"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Rump99,
  author = "Rump, S. M.",
  title = {{INTLAB - INTerval LABoratory}},
  booktitle = "Developments in Reliable Computing",
  pages = "77-104",
  publisher = "Kluwer Academic",
  year = "1999"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Sham92,
  author = "Shampine, L. F. and Gladwell, I.",
  title = {{The next generation of runge-kutta codes}},
  booktitle = "Computational Ordinary Differential Equations",
  pages = "145-164",
  publisher = "Oxford University Press",
  year = "1992"
}
```

———-

— **axiom.bib** —

```
@article{Smit62,
  author = "Smith, R. L.",
  title = {{Algorithm 116: Complex division}},
  journal = "Communs. Ass. comput. Mach.",
  volume = "5",
  pages = "435",
  year = "1962"
}
```

———-

— **axiom.bib** —

```
@book{Stew98,
  author = "Stewart, G. W.",
  title = {{Matrix Algorithms: Basic Decompositions, volume I}},
  publisher = "SIAM",
  year = "1998",
  isbn = "0-89871-414-1"
}
```

———-

— **axiom.bib** —

```
@article{Stew85,
  author = "Stewart, G. W.",
  title = {{A note on complex division}},
  journal = "ACM Trans. Math. Software",
  volume = "11",
  pages = "238-241",
  year = "1985"
}
```

———-

— **axiom.bib** —

```
@book{Stew90,
  author = "Stewart, G. W. and Sun, J.",
  title = {{Matrix Perturbation Theory}},
  publisher = "Academic Press",
  year = "1990"
}
```

———-

— **axiom.bib** —

```
@article{Stou07,
  author = "Stoutemyer, David R.",
  title = {{Useful Computations Need Useful Numbers}},
  year = "2007",
  publisher = "ACM",
  journal = "Communications in Computer Algebra",
  volume = "41",
  number = "3",
  abstract =
    "Most of us have taken the exact rational and approximate numbers in
    our computer algebra systems for granted for a long time, not thinking
    to ask if they could be significantly better. With exact rational
    arithmetic and adjustable-precision floating-point arithmetic to
    precision limited only by the total computer memory or our patience,
    what more could we want for such numbers?  It turns out that there is
    much that can be done that permits us to obtain exact results more
    often, more intelligible results, approximate results guaranteed to
    have requested error bounds, and recovery of exact results from
    approximate ones."
}
```

———-

— **axiom.bib** —

```
@article{Sutt13,
  author = "Sutton, Brian D.",
  title = {{Computing the Complete CS Decomposition}},
  journal = "Numerical Algorithms",
  volume = "50",
  pages = "33-65",
  year = "2013",
  month = "February",
  link = "\url{http://arxiv.org/pdf/0707.1838v3.pdf}",
  abstract =
    "An algorithm is developed to compute the complete CS decomposition
    (CSD) of a partitioned unitary matrix. Although the existence of the
    CSD has been recognized since 1977, prior algorithms compute only a
    reduced version (the 2-by-1 CSD) that is equivalent to two
    simultaneous singular value decompositions. The algorithm presented
    here computes the complete 2-by-2 CSD, which requires the simultaneous
```

```
    diagonalization of all four blocks of a unitary matrix partitioned
    into a 2-by-2 block structure. The algorithm appears to be the only
    fully specified algorithm available. The computation occurs in two
    phases. In the first phase, the unitary matrix is reduced to
    bidiagonal block form, as described by Sutton and Edelman. In the
    second phase, the blocks are simultaneously diagonalized using
    techniques from bidiagonal SVD algorithms of Golub, Kahan, and
    Demmel. The algorithm has a number of desirable numerical features.",
  paper = "Sutt13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Turi48a,
  author = "Turing, A. M.",
  title = {{Rounding-off errors in matrix processes}},
  journal = "Q. J. Mech. Appl. Math.",
  volume = "1",
  pages = "287-308",
  year = "1948",
  paper = "Turi48a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Vign93,
  author = "Vignes, J.",
  title = {{A stochastic arithmetic for reliable scientific computation}},
  journal = "Math. and Comp. in Sim.",
  volume = "25",
  pages = "233-261",
  year = "1993"
}
```

———-

— **axiom.bib** —

```
@article{Ward81,
  author = "Ward, Robert C.",
  title = {{Balancing the generalized eigenvalue problem}},
  journal = "SIAM J. Sci. and Stat. Comput.",
  volume = "2",
  number = "2",
  year = "1981",
  pages = "141-152",
  abstract =
```

```
    "An algorithm is presented for balancing the $A$ and $B$ matirces
    prior to computing the eigensystem of the generalized eigenvalue
    problem $Ax=\lambda Bx$. The three-step algorithm is specifically
    designed to preceed the $QZ$-type algorithms, but improved performance
    is expected from most eigensystem solvers. Permutations and two-sided
    diagonal transformations are applied to $A$ and $B$ to produce
    matrices with certain desirable properties. Test cases are presented
    to illustrate the improved accuracy of the computed eigenvalues."
}
```

———-

— **axiom.bib** —

```
@book{Wilk63,
  author = "Wilkinson, J. H.",
  title = {{Rounding Errors in Algebraic Processes}},
  publisher = "HMSO",
  series = "Notes on Applied Science, No. 32",
  year = "1963"
}
```

———-

— **axiom.bib** —

```
@book{Wilk65,
  author = "Wilkinson, J. H.",
  title = {{The Algebraic Eigenvalue Problem}},
  publisher = "Oxford University Press",
  year = "1965"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Wilk84,
  author = "Wilkinson, J. H.",
  title = {{The perfidious polynomial}},
  booktitle = "Studies in Numerical Analysis",
  volume = "24",
  chapter = "1",
  pages = "1-28",
  year = "1984"
}
```

———-

— **axiom.bib** —

```
@article{Wilk86,
  author = "Wilkinson, J. H.",
  title = {{Error analysis revisited}},
  journal = "IMA Bulletin",
  volume = "22",
  pages = "192-200",
  year = "1986"
}
```

———-

— **axiom.bib** —

```
@article{Wilk61,
  author = "Wilkinson, J. H.",
  title = {{Error analysis of diret methods of matrix inversion}},
  journal = "J. ACM",
  volume = "8",
  pages = "281-330",
  year = "1961"
}
```

———-

— **axiom.bib** —

```
@article{Wilk85,
  author = "Wilkinson, J. H.",
  title = {{The state of the art in error analysis}},
  journal = "NAG Newsletter",
  volume = "2/85",
  pages = "5-28",
  year = "1985"
}
```

———-

— **axiom.bib** —

```
@article{Wilk60,
  author = "Wilkinson, J. H.",
  title = {{Error analysis of floating-point computation}},
  journal = "Numer. Math.",
  volume = "2",
  pages = "319-340",
  year = "1960"
}
```

———-

— **axiom.bib** —

```
@book{Wilk71,
  author = "Wilkinson, J. H.",
  title = {{Handbook for Automatic Computation, V2, Linear Algebra}},
  publisher = "Springer-Verlag",
  year = "1971"
}
```

———-

— **axiom.bib** —

```
@misc{Yang14,
  author ="Yang, Xiang and Mittal, Rajat",
  title = {{Acceleration of the Jacobi iterative method by factors exceeding
           100 using scheduled relation}},
  link = "\url{http://engineering.jhu.edu/fsag/wp-content/uploads/sites/23/2013/10/JCP_revised_WebPost.pdf}",
  paper = "Yang14.pdf"
}
```

———-

1.14 Special Functions

— **axiom.bib** —

```
@inproceedings{Badd94,
  author = "Baddoura, Jamil",
  title = {{A Conjecture On Integration in Finite Terms with Elementary
           Functions and Polylogarithms}},
  booktitle = "ISSAC 94",
  year = "1994",
  pages = "158-162",
  isbn = "0-89791-638-7",
  abstract =
    "In this abstract, we report on a conjecture that gives the form of an
    integral if it can be expressed using elementary functions and
    polylogarithms. The conjecture is proved by the author in the cases of
    the dilogarithm and the trilogarithm [3] and consists of a
    generalization of Liouville's theorem on integration in finite terms
    with elementary functions. Those last structure theorems, for the
    dilogarithm and the trilogarithm, are the first case of structure
    theorems where logarithms can appear with non-constant
    coefficients. In order to prove the conjecture for higher
    polylogarithms we need to find the functional identities, for the
    polylogarithms that we are using, that characterize all the possible
    algebraic relations among the considered polylogarithms of functions
    that are built up from the rational functions by taking the considered
    polylogarithms, exponentials, logarithms and algebraics. The task of
    finding those functional identities seems to be a difficult one and is
```

```
    an unsolved problem for the most part to this date.",
  paper = "Badd94.pdf",
}
```

———-

— **axiom.bib** —

```
@article{Badd06,
  author = "Baddoura, Jamil",
  title = {{Integration in Finite Terms with Elementary Functions and
           Dilogarithms}},
  journal = "J. Symbolic Computation",
  volume = "41",
  number = "8",
  year = "2006",
  pages = "909-942",
  abstract =
    "In this paper, we report on a new theorem that generalizes
    Liouvilles theorem on integration in finite terms. The new theorem
    allows dilogarithms to occur in the integral in addition to
    transcendental elementary functions. The proof is based on two
    identities for the dilogarithm, that characterize all the possible
    algebraic relations among dilogarithms of functions that are built up
    from the rational functions by taking transcendental exponentials,
    dilogarithms, and logarithms. This means that we assume the integral
    lies in a transcendental tower.",
  paper = "Badd06.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Barn89,
  author = "Barnett, Michael P.",
  title = {{Using Partial Fraction Formulas to Sum some slowly convergent
           series analytically for molecular integral calculations}},
  journal = "ACM SIGSAM",
  volume = "23",
  number = "3",
  year = "1989",
  abstract =
    "Two sets of rational expressions, needed for quantum chemical
    calculations, have been constructed by mechanical application of
    partial fraction and polynomial operations on a CYBER 205. The
    algorithms were coded in FORTRAN, using simple array manipulation. The
    results suggest extensions that could be tackled with general
    algebraic manipulation programs.",
  paper = "Barn89.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bert94,
  author = "Bertrand, Laurent",
  title = {{On the Implementation of a new Algorithm for the Computation
           of Hyperelliptic Integrals}},
  booktitle = "ISSAC 94",
  isbn = "0-89791-638-7",
  pages = "211-215",
  year = "1994",
  abstract =
    "In this paper, we present an implementation in Maple of a new
    aJgorithm for the algebraic function integration problem in the
    particular case of hyperelliptic integrals.  This algorithm is based
    on the general algorithm of Trager [9] and on the arithmetic in the
    Jacobian of hyperelliptic curves of Cantor [2].",
  paper = "Bert94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Brow69,
  author = "Brown, W.S.",
  title = {{Rational Exponential Expressions and a Conjecture Concerning
           $\pi$ and $e$}},
  journal = "The American Mathematical Monthly",
  volume = "76",
  number = "1",
  year = "1969",
  pages = "28-34",
  abstract =
    "One of the most controversial and least well defined of mathematical
    problems is the problem of simplification.  The recent upsurge
    of interest in mechanized mathematics has lent new urgency to this
    problem, but so far very little has been accomplished.  This paper
    attempts to shed light on the situation by introducing the class of
    rational exponential expressions, defining simplification within
    this class, and showing constructively how to achieve it. It is shown
    that the only simplified rational exponential expression equivalent to
    0 is 0 itself, provided that an easily stated conjecture is true.
    However the conjecture, if true, will surely be difficult to prove,
    since it asserts as a special case that $\pi$ and $e$ are algebraically
    independent, and no one has yet been able to prove even the much
    weaker conjecture that $\pi+e$ is irrational.",
  paper = "Brow69.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Clar89,
  author = "Clarkson, M.",
  title = {{MACSYMA's inverse Laplace transform}},
  journal = "ACM SIGSAM Bulletin",
  volume = "23",
  number = "1",
  year = "1989",
  pages = "33-38",
  abstract =
    "The inverse Laplace transform capability of MACSYMA has been improved
    and extended. It has been extended to evaluate certain limits, sums,
    derivatives and integrals of Laplace transforms. It also takes
    advantage of the inverse Laplace transform convolution theorem, and
    can deal with a wider range of symbolic parameters.",
  paper = "Clar89.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Corl05,
  author = "Corless, Robert M. and Jeffrey, David J. and Watt, Stephen M.
            and  Bradford, Russell and Davenport, James H.",
  title = {{Reasoning about the elementary functions of complex analysis}},
  link = "\url{http://www.csd.uwo.ca/~watt/pub/reprints/2002-amai-reasoning.pdf}",
  abstract = "
    There are many problems with the simplification of elementary
    functions, particularly over the complex plane. Systems tend to make
    ``howlers'' or not to simplify enough. In this paper we outline the
    ``unwinding number'' approach to such problems, and show how it can be
    used to prevent errors and to systematise such simplification, even
    though we have not yet reduced the simplification process to a
    complete algorithm.  The unsolved problems are probably more amenable
    to the techniques of artificial intelligence and theorem proving than
    the original problem of complex-variable analysis.",
  paper = "Corl05.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Erde56,
  author = {Erd\'elyi, A.},
  title = {{Asymptotic Expansions}},
  year = "1956",
  isbn = "978-0-486-15505-0",
  publisher = "Dover Publications"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kaue08b,
  author = "Kauers, Manuel",
  title = {{Computer Algebra for Special Function Inequalities}},
  booktitle = "Tapas in Experimental Mathematics",
  pages = "215-235",
  year = "2008",
  abstract =
    "Recent coputer proofs for some special function inequalities are
    presented. The algorithmic ideas underlying these computer proofs
    are described, and the conceptual difference to existing
    algorithms for proving special function identities is discussed.",
  paper = "Kaue08b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Ng68,
  author = "Ng, Edward W. and Geller, Murray",
  title = {{A Table of Integrals of the Error functions}},
  link =
    "\url{http://nvlpubs.nist.gov/nistpubs/jres/73B/jresv73Bn1p1_A1b.pdf}",
  abstract = "
    This is a compendium of indefinite and definite integrals of products
    of the Error functions with elementary and transcendental functions.",
  paper = "Ng68.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Nort80,
  author = "Norton, Lewis M.",
  title = {{A Note about Laplace Transform Tables for Computer use}},
  journal = "ACM SIGSAM",
  volume = "14",
  number = "2",
  year = "1980",
  pages = "30-31",
  abstract =
    "The purpose of this note is to give another illustration of the fact
    that the best way for a human being to represent or process
    information is not necessarily the best way for a computer. The
```

```
    example concerns the use of a table of inverse Laplace transforms
    within a program, written in the REDUCE language [1] for symbolic
    algebraic manipulation, which solves linear ordinary differential
    equations with constant coefficients using Laplace transform
    methods. (See [2] for discussion of an earlier program which solved
    such equations.)",
  paper = "Nort80.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Piqu89,
  author = "Piquette, J. C.",
  title = {{Special Function Integration}},
  journal = "ACM SIGSAM Bulletin",
  volume = "23",
  number = "2",
  year = "1989",
  pages = "11-21",
  abstract =
    "This article describes a method by which the integration capabilities
    of symbolic-mathematics computer programs can be extended to include
    integrals that contain special functions. A summary of the theory that
    forms the basis of the method is given in Appendix A. A few integrals
    that have been evaluated using the method are presented in Appendix
    B. A more thorough development and explanation of the method is given
    in Piquette, in review (b)."
}
```

———-

1.15 Exponential Integral $E_1(x)$

— **axiom.bib** —

```
@misc{Gell69,
  author = "Geller, Murray and Ng, Edward W.",
  title = {{A Table of Integrals of the Exponential Integral}},
  link =
    "\url{http://nvlpubs.nist.gov/nistpubs/jres/73B/jresv73Bn3p191_A1b.pdf}",
  abstract = "
    This is a compendium of indefinite and definite integrals of products
    of the Exponential Integral with elementary or transcendental functions.",
  paper = "Gell69.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Segl98,
  author = "Segletes, S.B.",
  title = {{A compact analytical fit to the exponential integral $E_1(x)$}},
  year = "1998",
  institution = "U.S. Army Ballistic Research Laboratory,
                 Aberdeen Proving Ground, MD",
  type = "Technical Report",
  number = "ARL-TR-1758",
  algebra = "\newline\refto{package DFSFUN DoubleFloatSpecialFunctions}",
  abstract = "
    A four-parameter fit is developed for the class of integrals known as
    the exponential integral (real branch). Unlike other fits that are
    piecewise in nature, the current fit to the exponential integral is
    valid over the complete domain of the function (compact) and is
    everywhere accurate to within $\pm 0.0052\%$ when evaluating the first
    exponential integral, $E_1$. To achieve this result, a methodology
    that makes use of analytically known limiting behaviors at either
    extreme of the domain is employed. Because the fit accurately captures
    limiting behaviors of the $E_1$ function, more accuracy is retained
    when the fit is used as part of the scheme to evaluate higher-order
    exponential integrals, $E_n$, as compared with the use of brute-force
    fits to $E_1$, which fail to accurately model limiting
    behaviors. Furthermore, because the fit is compact, no special
    accommodations are required (as in the case of spliced piecewise fits)
    to smooth the value, slope, and higher derivatives in the transition
    region between two piecewise domains. The general methodology employed
    to develop this fit is outlined, since it may be used for other
    problems as well.",
  paper = "Segl98.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Se09,
  author = "Segletes, S.B.",
  title = {{Improved fits for $E_1(x)$ {\sl vis-\'a-vis} those presented
           in ARL-TR-1758}},
  type = "Technical Report",
  number = "ARL-TR-1758",
  institution ="U.S. Army Ballistic Research Laboratory,
                Aberdeen Proving Ground, MD",
  year = "1998",
  month = "September",
  abstract = "
    This is a writeup detailing the more accurate fits to $E_1(x)$,
    relative to those presented in ARL-TR-1758.  My actual fits are to
    \[F1 =[x\ exp(x) E_1(x)]\] which spans a functional range from 0 to 1.
    The best accuracy I have been yet able to achieve, defined by limiting
    the value of \[[(F1)_{fit} - F1]/F1\] over the domain, is
```

```
    approximately 3.1E-07 with a 12-parameter fit, which unfortunately
    isn't quite to 32-bit floating-point accuracy.  Nonetheless, the fit
    is not a piecewise fit, but rather a single continuous function over
    the domain of nonnegative x, which avoids some of the problems
    associated with piecewise domain splicing.",
  paper = "Se09.pdf"
}
```

1.16 Proving Axiom Correct – The Project

1.16.1 A

1.16.2 B

1.16.3 C

1.16.4 D

1.16.5 E

1.16.6 F

1.16.7 G

1.16.8 H

1.16.9 I

1.16.10 J

1.16.11 K

1.16.12 L

1.16.13 M

1.16.14 N

1.16.15 O

1.16.16 P

1.16.17 Q

1.16.18 R

1.16.19 S

1.16.20 T

1.16.21 U

1.16.22 V

1.16.23 W

— **axiom.bib** —

```
@misc{Wadl00,
  author = "Wadler, Philip",
  title = {{Proofs are Programs: 19th Century Logic and 21st Century
            Computing}},
  link = "\url{https://homepages.inf.ed.ac.uk/wadler/papers/frege/frege.pdf}",
  year = "2000",
  paper = "Wadl00.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Wadl03,
  author = "Wadler, Philip",
  title = {{The Girard-Reynolds Isomorphism}},
  journal = "Information and Computation",
  volume = "186",
  number = "2",
  pages = "260-280",
  year = "2003",
  abstract =
    "The second-order polymorphic lambda calculus, F2, was
    independently discovered by Girard and Reynolds. Girard
    additionally proved a {\sl representation} theorem: every function
    on natural numbers that can be proved total in second-order
    intutionistic propositional logic, P2, can be represented in
    F2. Reynolds additionally proved an {\sl abstraction} theorem: for
    a suitable notion of logical relation, every term in F2 takes
    related arguments into related results. We observe that the
    essence of Girard's result is a projection from P2 into F2, and
    that the essence of Reynolds's result is an embedding of F2 into
    P2, and that the Reynolds embedding followed by the Girard
    projection is the identity. The Girard projection discards all
    first-order quantifiers, so it seems unreasonable to expect that
    the Girard projection followed by the Reynolds embedding should
    also be the identity. However, we show that in the presence of
    Reynolds's parametricity property that this is indeed the case,
    for propositions corresponding to inductive definitions of
    naturals, products, sums, and fixpoint types.",
  paper = "Wadl03.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Wall04,
  author = "Wallenburg, Angela",
```

```
  title = {{Inductive Rules for Proving Correctness of Imperative Programs}},
  school = "Goteborg University",
  link = "\url{http://www.cse.chalmers.se/~angelaw/papers/lic.pdf}",
  year = "2004",
  abstract =
    "This thesis is aimed at simplifying the user-interaction in
    semi-interactive theorem proving for imperative programs.  More
    specically , we describe the creation of customised induction rules
    that are tailor-made for the specific program to verify and thus make
    the resulting proof simpler.  The concern is in user interaction,
    rather than in proof strength.  To achiev e this, two different
    verication techniques are used.

    In the first approach, we develop an idea where a software testing
    technique, partition analysis, is used to compute a partition of the
    domain of the induction variable, based on the branch predicates in
    the program we wish to prove correct.  Based on this partition we
    derive mechanically a partitioned induction rule, which then inherits
    the divide-and-conquer style of partition analysis, and (hopefully) is
    easier to use than the standard (Peano) induction rule.

    The second part of the thesis continues with a more thorough
    development of the method.  Here the connection to software testing is
    completely removed and the focus is on inductive theorem proving only.
    This time, we make use of failed proof attempts in a theorem prover
    to gain information about the problem structure and create the
    partition.  Then, based on the partition we create an induction rule,
    in destructor style, that is customised to make the proving of the
    loop simpler.

    With the customised induction rules, in comparison to standard (Peano)
    induction or Noetherian induction, the required user interaction is
    moved to an earlier point in the proof which also becomes more
    modularised.  Moreover, by using destructor style induction we
    circumvent the problem of creating inverses of functions.  The
    soundness of the customised induction rules created by the method is
    shown. Furthermore, the machinery of the theorem prover (KeY) is used
    to make the method automatic.  The induction rules are developed to
    prove the total correctness of loops in an object-oriented language
    and we concentrate on integers.",
  paper = "Wall04.pdf",
  keywords = "printed, DONE"
}
```

———-

— axiom.bib —

```
@misc{Wall09,
  author = "Wallenburg, Angela",
  title = {{Generalisation of Inductive Formulae based on Proving by
            Symbolic Execution}},
  year = "2009",
```

```
  link = "\url{http://www.cse.chalmers.se/~angelaw/papers/awWING2009.pdf}",
  abstract =
    "Induction is a powerful method that can be used to prove the total
    correctness of program loops.  Unfortunately the induction proving
    process in an interactive theorem prover is often very cumbersome.  In
    particular it can be difficult to find the right induction formula.
    We describe a method for generalising induction formulae by analysing
    a symbolic proof attempt in a semi-interactive first-order theorem
    prover.  Based on the proof attempt we introduce universally
    quantified variables, meta-variables and sets of constraints on these.
    The constraints describe the conditions for a successful proof.  By
    the help of examples, we outline some classes of problems and their
    associated constraint solutions, and possible ways to automate the
    constraint solving.",
  paper = "Wall09.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@mastersthesis{Walt12,
  author = "van der Walt, Paul",
  title = {{Reflection in Agda}},
  school = "Utrecht University",
  year = "2012",
  abstract =
    "This project explores the recent addition to Agda enabling
    reflection, in the style of Lisp, MetaML, and Template Haskell. It
    illustrates several possible applications of reflection that arise in
    dependently typed programming, and details the limitations of the
    current implementation of reflection. Examples of type-safe
    metaprograms are given that illustrate the power of reflection coupled
    with a dependently typed language. Among other things the limitations
    inherent in having quote and unquote implemented as keywords are
    highlighted. The fact that lambda terms are returned without typing
    information is discussed, and a solution is presented. Also provided
    is a detailed users guide to the reflection API and a library of
    working code examples to illustrate how various common tasks can be
    performed, along with suggestions for an updated reflection API in a
    future version of Agda.",
  paper = "Walk12.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Wang78,
  author = "Wang, Paul S.",
```

```
  title = {{An Improved Multivariate Polynomial Factoring Algorithm}},
  journal = "Mathematics of Computation",
  volume = "32",
  number = "144",
  year = "1978",
  pages = "1215-1231",
  link = "\url{http://www.ams.org/journals/mcom/1978-32-144/S0025-5718-1978-0568284-3/S0025-5718-1978-0568284-3.
  abstract = "
    A new algorithm for factoring multivariate polynomials over the
    integers based on an algorithm by Wang and Rothschild is described.
    The new algorithm has improved strategies for dealing with the known
    problems of the original algorithm, namely, the leading coefficient
    problem, the bad-zero problem and the occurence of extraneous factors.
    It has an algorithm for correctly predetermining leading coefficients
    of the factors. A new and efficient p-adic algorith named EEZ is
    described. Basically it is a linearly convergent variable-by-variable
    parallel construction. The improved algorithm is generally faster and
    requires less store than the original algorithm. Machine examples with
    comparative timing are included.",
  paper = "Wang78.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wang80,
  author = "Wang, Paul S.",
  title = {{The EEZ-GCD Algorithm}},
  journal = "SIGSAM Bulletin",
  volume = "14",
  number = "2",
  pages = "50-60",
  year = "1980",
  abstract =
    "An enhanced gcd algorithm based on the EX-GCD algorithm is
    described. Implementational aspects are emphasized. It is
    generally faster and is particularly suited for computing gcd of
    sparse multivariate polynomials. The EEZ-GCD algorithm is
    characterized by the following features:
    \begin{enumerate}
    \item avoiding unlucky evaluations,
    \item predetermining the correct leading coefficient of the
    desired gcd,
    \item using the sparsity of the given polynomials to determine
    terms in the gcd and
    \item direct methods for dealing with the ``common divisor problem.''
    \end{enumerate}
    The common divisor problem occurs when the gcd has a different
    common divisor with each of the cofactors. The EZ-GCD algorithm
    does a square-free decomposition in this case. It can be avoided
    resulting in increased speed. One method is to use parallel p-adic
    construction of more than two factors. Machine examples with
```

```
    timing data are included.",
  paper = "Wang80.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Wang19a,
  author = "Wang, Qingxiang and Brown, Chad and Kaliszyk, Cezary and
            Urban, Josef",
  title = {{Exploration of Neural Machine Translation in
            Autoformalization of Mathematics in Mizar}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1912.02636.pdf}",
  abstract =
    "In this paper we share several experiments trying to
    automatically translate informal mathematics into formal
    mathematics. In our context informal mathematics refers to
    human-written mathematical sentences in the LaTeX format; and
    formal mathematics refers to statements in the Mizar language. We
    conducted our experiments against three established neural
    network-based machine translation models that are know to deliver
    competitive results on translating between natural languages. To
    train these models we also prepared four informal-to-formal
    datasets. We compare and analyze our results according to whether
    the model is supervised or unsupervised. In order to augment the
    data available for auto-formalization and improve the results, we
    develop a custom type-elaboration mechanism and integrate it into
    the supervised translation.",
  paper = "Wang19a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Wang19,
  author = "Wang, Ke",
  title = {{Learning Scalable and Precise Representation of Program
            Semantics}},
  year = "2019",
  abstract =
    "Neural program embedding has shown potential in aiding the
    analysis of large-scale, complicated software. Newly proposed deep
    neural architectures pride themselves on learning program
    semantics rather than superficial syntactic features. However, by
    considering the source code only, the vast majority of neural
    networks do not capture a deep, precise representation of program
    semantics. In this paper, we present DYPRO, a novel deep neural
    network that learns from program execution traces. Compared to the
```

```
    prior dynamic models, not only is DYPRO capable of generalizing
    across multiple executions for learning a program's dynamic
    semantics in its entirety, but DYPRO is also more efficient when
    dealing with programs yielding long execution traces. For
    evaluation, we task DYPRO with semantic classification
    (i.e. categorizing programs based on their semantics) and compared
    it against two prominent static models: Gated Graph Neural Network
    and TreeLSTM. We find that DYPRO achieves the highest prediction
    accuracy among all models. To further reeal the capacity of all
    aforementioned deep neural architectures, we examine if the models
    can learn to detect deeper semantic properties of a program. In
    particular given a task of recognizing loop invariants, we show
    DYPRO beats all static models by a wide margin.",
  paper = "Wang19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Warn16,
  author = "Warne, Henrik",
  title = {{More Good Programming Quotes}},
  year = "2019",
  link = "\url{https://henrikwarne.com/2016/04/17/more-good-programming-quotes}"
}
```

———-

— **axiom.bib** —

```
@misc{Warn19,
  author = "Warne, Henrik",
  title = {{More Good Programming Quotes, Part 3}},
  year = "2019",
  link = "\url{https://henrikwarne.com/2019/04/03/more-good-programming-quotes-part-3}"
}
```

———-

— **axiom.bib** —

```
@misc{Warr77,
  author = "Warren, David H.D. and Pereira, Luis M.",
  title = {{Prolog -- The Language and its Implementation Compared
            with Lisp}},
  year = "1977",
  link =
    "\url{http://www.public.imtbs-tsp.eu/~gibson/Teaching/Teaching-ReadingMaterial/WarrenPereiraPereira77.pdf}",
  abstract =
```

```
    "Prolog is a simple but powerful programming language founded on
    symbolic logic. The basic computational mechanism is a pattern
    matching process (``unification'') operating on general record
    structures (``terms of logic''). We priefly review the language
    and compare it especially with pure Lisp. The remainder of the
    paper discusses techniques for implementing Prolog efficiently; in
    particular we describe how to compile the patterns involved in the
    matching process. These techniques are as incorporated in our
    DECsystem-10 Prolog compiler (written in Prolog). The code it
    generates is comparable in speed with that produced by existing
    DEC10 Lisp compilers. We argue that pattern matching is a better
    method for expressing operations on structured data than
    conventional selectors and constructors -- both for the user and
    for the implementor.",
  paper = "Warr77.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Warr83,
  author = "Warren, David H.D.",
  title = {{An Abstract Prolog Instruction Set}},
  type = "technical note",
  number = "309",
  institution = "SRI International",
  year = "1983",
  paper = "Warr83.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Watt09,
  author = "Watt, Stephen M.",
  title = {{Algorithms for the Functional Decomposition of Laurent
            Polynomials}},
  journal = "LNCS",
  volume = "5625",
  year = "2009",
  abstract =
    "Recent work has detailed the conditions under which univariate
    Laurent polynomials have functional decompositions. This paper
    presents algorithms to compute such univariate Laurent polynomial
    decompositions efficiently and gives their multivariate
    generalizations.

    One application of functiona decomposition of Laurent polynomials
    is the functional decomposition of so-called ``symbolic
```

```
    polynomials''. These are polynomial-like objects whose exponents
    are themselves integer-valued polynomials rather than
    integers. The algebraic independence of $X$, $X^n$, $X^{n^2/2}$,
    etc., and some elementary results on integer-valued polynomials
    allow problems with symbolic polynomials to be reduced to problems
    with multivariate Laurent polynomials. Hence we are interested in
    the functional decomposition of these objects.",
  paper = "Watt09.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Watt09a,
  author = "Watt, Stephen M.",
  title = {{On the Future of Computer Algebra Systems at the threshold
            of 2010}},
  booktitle = "Proc. ASCM-MACIS",
  publisher = "unknown",
  pages = "422-430",
  abstract =
    "This paper discusses ways in which software systems for computer
    algebra could be improved if designed from scratch today rather
    than evolving designs from the 1980s",
  paper = "Watt09a.pdf",
  keywords = "axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Weir18,
  author = "Weirich, Stephanie",
  title = {{Dependent Types in Haskell}},
  year = "2018",
  link = "\url{https://www.youtube.com/watch?v=wNa3MMbhwS4}",
  abstract =
    "What has dependent type theory done for Haskell? Over the past ten
    years, the Glasgow Haskell compiler (GHC) has adopted many type system
    features inspired by dependent type theory. In this talk, I will
    discuss the influence of dependent types on the design of GHC and on
    the practice of Haskell programmers. In particular, I will walk
    through an extended example and use it to analyze what it means to
    program with with dependent types in Haskell. Throughout, I will will
    discuss what we have learned from this experiment in language design:
    what works now, what doesn't work yet, and what surprised us along
    the way."
}
```

———-

— **axiom.bib** —

```
@misc{Weir19,
  author = "Weirich, Stephanie and Choudhury, Pritam and Voizard,
            Antoine and Eisenberg, Richard A.",
  title = {{A Role for Dependent Types in Haskell (Extended Version)}},
  link = "\url{https://arxiv.org/pdf/1905.13706.pdf}",
  year = "2019",
  abstract =
    "Modern Haskell supports zero-cost coercions, a mechanism where
    types that share the same run-time representation may be freely
    converted between. To make sure such conversions are safe and
    desirable, this feature relies on a mechanism of roles to prohibit
    invalid coercions. In this work, we show how to integrate roles
    with dependent type systems and prove, using the Coq proof
    assistant, that the resulting system is sound. We have designed
    this work as a foundation for the addition of dependent types to
    the Glasgow Haskell Compiler, but we also expect that it will be
    of use to designers of other dependently-typed languages who might
    want to adopt Haskell's safe coercion feature.",
  paper = "Weir19.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Weit16,
  author = "Weitz, Edmund",
  title = {{Common Lisp Recipes}},
  publisher = "Apress",
  year = "2016",
  isbn = "978-1-4842-1177-9",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inbook{Wern97,
  author = "Werner, Benjamin",
  title = {{Sets in Types, Types in Sets}},
  booktitle = "Theoretical Aspects of Computer Software",
  publisher = "Springer",
  chapter = "unknown",
  pages = "530-546",
  year = "1997"
}
```

———-

— **axiom.bib** —

```
@misc{Wern19,
  author = "Wernhard, Christoph",
  title = {{PIE -- Proving, Interpolating and Eliminating on the Basis
           of First-Order Logic}},
  year = "2019",
  link = "\url{https://arxiv.org/pdf/1908.11137.pdf}",
  abstract =
    "PIE is a Prolog-embedded environment for automated reasoning on
    the basis of first-order logic. It includes a versatile formula
    macro system and supports the creation of documents that
    intersperse macro definitions, reasoner invocations and LaTeX
    formatted natural language text. Invocation of various reasoners
    is supported. External provers as well as sub-systems of PIE,
    which include preprocessors, a Prolog-based first-order prover,
    methods for Craig interpolation and methods for second-order
    quantifier elimination.",
  paper = "Wern19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Wied03b,
  author = "Wiedijk, Freek",
  title = {{Comparing Mathematical Provers}},
  journal = "LNCS",
  volume = "2594",
  year = "2003",
  abstract =
    "We compare fifteen systems for the formalizatioin of mathematics
    with a computer. We present several tables that list various
    properties of these programs. The three main dimensions on which
    we compare these systems are: the size of their library, the
    strength of their logic and their level of automation.",
  paper = "Wied03b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Wien61,
  author = "Wiener, Norbert",
  title = {{Cybernetics}},
  publisher = "MIT Press",
  year = "1961",
  keywords = "shelf"
```

```
}
```

———-

— **axiom.bib** —

```
@book{Wilf75,
  author = "Wilf, William and Johnsson, Richard K. and
            Weinstock, Charles B. and Hobbs, Steven O.
            and Geschke, Charles M.",
  title = {{The Design of an Optimizing Compiler}},
  publisher = "Elsevier",
  year = "1975",
  isbn = "0-444-00158-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Wilk85a,
  author = "Wilkes, Maurice",
  title = {{Memoirs of a Computer Pioneer}},
  publisher = "MIT Press",
  year = "1985"
}
```

———-

— **axiom.bib** —

```
@article{Wils12,
  author = "Wilson, David J. and Bradford, Russell J. and
            Davenport, James H.",
  title = {{Speeding Up Cylindrical Algebraic Decomposition by
            Groebner Bases}},
  journal = "LNCS",
  volume = "7362",
  year = "2012",
  abstract =
    "Groebner Bases and Cylindrical Algebraic Decomposition are
    generally thought of as two, rather different, methods of looking
    at systems of equations and, in the case of Cylindrical Algebraic
    Decomposition, inequalities. However, even for a mixed system of
    equalities and inequalities, it is possible to apply Groebner
    bases to the (conjoined) equalities before invoking CAD. We see
    that this is, quite often but not always, a beneficial
    preconditioning of the CAD problem.

    It is also possible to precondition the (conjoined) inequalities
```

```
    with respect to the equalities, and this can also be useful in
    many cases.",
  paper = "Wils12.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Wink84,
  author = "Winkler, Franz",
  title = {{The Church Rosser Property in Computer Algebra and Special
            Theorem Proving}},
  publisher = "Wien",
  year = "1984",
  isbn = "3-85369-584-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Wirt80,
  author = "Wirth, N.",
  title = {{Modula-2}},
  year = "1980",
  link = "\url{http://www.ada-auth.org/standards/12rm/RM-Final.pdf}",
  abstract =
    "Modula-2 is a general purpose programming language primarily
    designed for system implementation. This report constitutes its
    definition in a concise, although informal style. It also
    describes the use of an implementation for the PDP-11 computer.",
  paper = "Wirt80.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Wirt83,
  author = "Wirth, N.",
  title = {{Programming in Modula-2}},
  publisher = "Springer-Verlag",
  year = "1983",
  isbn = "978-3-642-96878-5"
}
```

———-

— **axiom.bib** —

```
@article{Wirt88,
  author = "Wirth, N.",
  title = {{Type Extensions}},
  journal = "TOPLAS",
  volume = "10",
  number = "2",
  year = "1988",
  pages = "203-214",
  abstract =
    "Software systems represent a hierarchy of modules. Client modules
    contain sets of procedures that extend the capabilities of imported
    modules. This concept of extension is here applied to data
    types. Extended types are related to their ancestor in terms of a
    hierarchy. Variables of an extended type are compatible with variables
    of the ancestor type. This scheme is expressed by three language
    constructs only: the declaration of extended record types, the type
    test, and the type guard. The facility of extended types, which
    closely resembles the class concept, is defined in rigorous and
    concise terms, and an efficient implementation is presented.",
  paper = "Wirt88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wirt95,
  author = "Wirth, Niklaus",
  title = {{A Plea for Lean Software}},
  publisher = "IEEE",
  journal = "Computer",
  year = "1995",
  pages = "64-68",
  paper = "Wirt95.pdf",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Wirt15,
  author = "Wirth, Niklaus",
  title = {{The Design of a RISC Architecture and its Implementation
            with an FPGA}},
  link =
    "\url{https://www.inf.ethz.ch/personal/wirth/FPGA-relatedWork/RISC.pdf}",
  year = "2015",
  paper = "Wirt15.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Wolf18,
  author = "Wolfram, Stephen",
  title = {{Mathematica Website}},
  year = "2018",
  link = "\url{http://www.wolfram.com}"
}
```

———-

— **axiom.bib** —

```
@book{Wood20,
  author = "Woodcock, Jim and Davies, Jim",
  title = {{Using Z: Specification, Refinement, and Proof}},
  year = "2020",
  link = "\url{http://www.cs.cmu.edu/~15819/zedbook.pdf}",
  publisher = "CMU PDF",
  paper = "Wood20.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wrig95,
  author = "Wright, Andrew K.",
  title = {{Simple Imperative Polymorphism}},
  booktitle = "LISP and Symbolic Computation",
  publisher = "Kluwer Academic",
  pages = "242-256",
  year = "1995",
  abstract =
    "This paper describes a simple extension of the Hindley-Milner
    polymorphic type discipline to call-by-value languages that
    incorporate imperative features like references, exceptions, and
    continuations. This extension sacrifices the ability to type every
    purely functional expression that is typable in the Hindley-Milner
    system. In return, it assigns the same type to functional and
    imperative implementations of the same abstraction. Hence with a
    module system that separates specifications from implementations,
    imperative features can be freely used to implement polymorphic
    specifications. A study of a number of ML programs shows that the
    inability to type all Hindley-Milner typable expressions seldom
    impacts realistic programs. Furthermore, most programs that are
    rendered untypable by the new system can be easily repaired.",
  paper = "Wrig95.pdf",
```

```
  keywords = "printed"
}
```

———-

1.16.24 X

— **axiom.bib** —

```
@book{Xixx19,
  author = "Xi, Hongwei",
  title = {{Introduction to Programming in ATS}},
  publisher = "ATS Trustful Software, Inc",
  year = "2019",
  abstract =
    "As a programming language, ATS is both syntax-rich and
    feature-rich. This book introduces the reader to some core
    features of ATS, including basic functional programming, simple
    types, (recursively defined) datatypes, polymorphic types,
    dependent types, linear types, theorem proving, programming with
    theorem proving (PwTP), and template-based programming. Although
    the reader is not assumed to be familiar with programming in
    genera, the book is likely to be rather dense for someone without
    considerable programming experience",
  paper = "X1xx19.pdf"
}
```

———-

1.16.25 Y

— **axiom.bib** —

```
@inproceedings{Yall19,
  author = "Yallop, Jeremy and White, Leo",
  title = {{Lambda: The Ultimate Sublanguage (Experience Report}},
  booktitle = "Inter. Conf. on Functional Programming",
  publisher = "ACM",
  year = "2019",
  abstract =
    "We describe our experience teaching an advanced typed functional
    programming course based around the use of Sysmte $F_\omega$ as a
    programming language.",
  paper = "Yall19.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Yedi16,
  author = "Yedidia, Adam and Aaronson, Scott",
  title = {{A Relatively Small Turing Machine Whose Behavior Is
            Independent of Set Theory}},
  journal = "Complex Systems",
  volume = "25",
  number = "4",
  pages = "297-327",
  link = "\url{http://www.complex-systems.com/pdf/25-5-5.pdf}",
  year = "2016",
  abstract =
    "Since the definition of the Busy Beaver function in Rado in 1962,
    an interesting open question has been what the smallest value of
    $n$ for which $BB(n)$ is independent of ZFC set theory. Is this
    $n$ approximately 10, or closer to 1,000,000, or is it even
    larger? In this paper, we show that it is at most 7,918 by
    presenting an explicit description of a 7,918-state Turing machine
    $Z$ with 1 tape and a 2-symbol alphabet that cannot be proved to
    run forever in ZFC (even though it presumably does), assuming ZFC
    is consistent. The machine is based on work of Harvey Friedman on
    independent statements involving order-invariant graphs. In doing
    so, we give the first known upper bound on the highest provable
    Busy Beaver number in ZFC. We also present a 4,888-state Turing
    machine $G$ that halts if and only if there is a counterexample of
    Goldbach's conjecure, an at 5,372-state Turing machine $R$ that
    halts if and only if the Riemann hypothesis is false. To create
    $G$, $R$, and $Z$, we develop and use a higher-level language,
    Laconic, which is much more convenient than direct state
    manipulation.",
  paper = "Yedi16.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Yosh97,
  author = "Yoshida, Masaaki",
  title = {{Hypergeometric Functions, My Love}},
  publisher = "Springer",
  year = "1997",
  isbn = "978-3-322-90168-2"
}
```

———-

— **axiom.bib** —

```
@misc{Yous04,
  author = "Youssef, Saul",
  title = {{Prospects for Category Theory in Aldor}},
```

```
  year = "2004",
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/Youssef-ProspectsForCategoryTheoryInAldor.pdf}",
  abstract =
    "Ways of encorporating category theory constructions and results into
    the Aldor language are discussed. The main features of Aldor which
    make this possible are identified, examples of categorical
    constructions are provided and a suggestion is made for a foundation
    for rigorous results.",
  paper = "Yous04.pdf"
}
```

———-

1.16.26 Z

— **axiom.bib** —

```
@misc{Zavi18,
  author = "Zavialov, Vladislav",
  title = {{Why Dependent Haskell is the Future of Software Development}},
  link = "\url{https://serokell.io/blog/why-dependent-haskell}",
  year = "2018"
}
```

———-

— **axiom.bib** —

```
@misc{Zeil02,
  author = "Zeilberger, Doron",
  title = {{Topology: The Slum of Combinatorics}},
  year = "2002",
  link = "\url{http://sites.math.rutgers.edu/~zeilberg/Opinion1.html}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Zhao19,
  author = "Zhao, Jinxu and Oliveira, Bruno C.D.S and
            Schrijvers, Tom",
  title = {{A Mechanical Formalization of Higher-Ranked Polymorphic
            Type Interence}},
  booktitle = "Inter. Conf. on Functional Programming",
  publisher = "ACM",
  year = "2019",
  abstract =
    "Modern functional programming languages, such as Haskell or
    OCaml, use sophisticated forms of type inference. While an
```

```
    important topic in the Programming Languages research, there is
    little work on the mechanization of the metatheory of type
    inference in theorem provers. In particular we are unaware of any
    complete formalization of the type inference algorithms that are
    the backbone of modern functional languages.

    This paper presents the first full mechanical formalization of the
    metatheory for higher-ranked polymorphic type inference. The
    system that we formalize is the bidirectional type system by
    Dunfield and Krishnaswami (DK). The DK type system has two
    variants (a declarative and an algorithmic one) that have been
    manually proven sound, complete and decidable. We present a
    mechanical formalization in the Abella theorem provers of DK's
    declarative type system with a novel algorithmic system. We have a
    few reasons to use a new algorithm. Firstly, our new algorithm
    employs worklist judgments, which precisely capture the scope of
    variables and simplify the formalization of scoping in a theorem
    prover. Secondly, while DKs original formalizations comes with
    very well-written manual proofs, there are several details missing
    and some incorrect proofs, which complicate the task of writing a
    mechanized proof. Despite the use of a different algorithm we
    prove the same results as DK, although with significantly
    different proofs and proof techniques. Since such type inference
    algorithms are quite subtle and have a complex metatheory,
    mechanical formalizations are an important advance in type
    inference research.",
  paper = "Zhao19.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Zuck19,
  author = "Zucker, Philip",
  title = {{Grobner Bases and Optics}},
  link = "\url{http://www.philipzucker.com/grobner-bases-and-optics}",
  year = "2019"
}
```

———-

1.17 Proving Axiom Correct – Spring 2018

1.17.1 A

— **axiom.bib** —

```
@inproceedings{Abad90,
```

```
  author = "Abadi, Martin and Cardelli, Luca and Curien, Pierre-Louis
            and Levy, Jean-Jacques",
  title = {{Explicit Substitutions}},
  booktitle = "Symp. of Principles of Programming Languages",
  publisher = "ACM",
  year = "1990",
  link = "\url{http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-54.pdf}",
  abstract =
    "The $lambda\sigma$-calculus is a refinement of the
    $lambda$-calculus where substitutions are manipulated
    explicitly. The $\lambda\sigma$-calculus provides a setting for
    studying the theory of substitutions, with pleasant mathematical
    properties. It is also a useful bridge between the classical
    $\lambda$-calculus and concrete implementations.",
  paper = "Abad90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Abda73,
  author = "Abdali, S. Kamal",
  title = {{A Simple Lambda-Calculus Model of Programming Languages}},
  isbn = "978-1332196029",
  publisher = "Forgotten Books",
  year = "1973",
  abstract =
    "We present a simple correspondence between a large subset of
    ALGOL 60 language and lambda-calculus. With the aid of this
    correspondence, a program can be translated into a single
    lambda-expression. In general, the representation of a program is
    specified by means of a system of simultaneous conversion
    relations among the representations of the program
    constituents. High-level programming language features are treated
    directly, not in terms of the representations of machine-level
    operations. The model includes input-output in such a way that
    when the representation of a (convergent) program is applied to
    the input item representations, the resulting combination reduces
    to a tuple of the representations of the output items. This model
    does not introduce any imperative notions into the calculus; the
    descriptive programming constructs, such as expressions, and the
    imperative ones, such as assignment and jumps, are translated
    alike into pure lambda expressions. The applicability of the model
    to the problems of proving program equivalence and correctness is
    illustrated by means of simple examples.",
  paper = "Abda73.pdf",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Abda74,
  author = "Abdali, S. Kamal",
  title = {{A Lambda-Calculus Model of Programming Languages -
           I. Simple Constructs}},
  journal = "J. of Computer Languages",
  volume = "1",
  pages = "287-301",
  year = "1974",
  abstract =
    "A simple correspondence is presented between the elementary
    constructs of programming languages and the lambda-calculus. The
    correspondence is obtained by using intuitive, functional
    interpretation of programming language constructs, completely
    avoiding the notions of machine memory and address. In particular,
    the concepts of program variable and assignments are accounted for
    in terms of the concepts of mathematical variable and
    substitution, respectively. Lambda-expression representations are
    given of assignments, conditional and compound statements,
    input-output, and blocks. Algol 60 is used as the illustrative
    language.",
  paper = "Abda74.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Abda74a,
  author = "Abdali, S. Kamal",
  title = {{A Lambda-Calculus Model of Programming Languages -
           II. Jumps and Procedures}},
  journal = "J. of Computer Languages",
  volume = "1",
  pages = "303-320",
  year = "1974",
  abstract =
    "The correspondence between programming languages and the
    lambda-calculus presented in Part I of the paper is extended there
    to include iteration statements, jumps, and procedures. Programs
    containing loops are represented by lambda-expressions whose
    components are specified recursively by means of systems of
    simultaneous conversion relations. The representation of
    call-by-name and side-effects in a program is accomplished without
    any resort to the concepts of memory andaddress by making use of a
    number of variables in addition to those declared by the programs;
    with the aid of these additional variables, the parameter linkage
    operations can be simulated by pure substitution. The
    applicability of the model to the problems of proving program
    correctness and equivalence is demonstrated by means of examples.",
  paper = "Abda74a.pdf",
```

```
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Abda74b,
  author = "Abdali, S. Kamal",
  title = {{A Combinatory Logic Model of Programming Languages}},
  school = "University of Wisconsin",
  year = "1974",
  abstract =
    "A simple correspondence is presented between a large subset of the
    ALGOL 60 language and the combinatory logic.  With the aid of this
    correspondence, a program can be translated into a single combinatory
    object.  The combinatory object representing a program is specified,
    in general, by means of a system of reduction relations among the
    representations of the program constituents.  This object denotes, in
    terms of the combinatory logic, the function that the program is
    intended to compute.

    The model has been derived by using intuitive, functional
    interpretations of the constructs of programming languages, completely
    avoiding the notions of machine command and address.  In particular,
    the concepts of program variable, assignment, and procedure have been
    accounted for in terms of the concepts of mathematical variable,
    substitution, and function, respectively.

    High-level programming language features are represented in the
    combinatory logic directly, not in terms of the representations of
    machine-level operations.  Input-output is treated in such a
    manner that when the representation of a program is applied to the
    representations of the input items, the resulting combination reduces
    to a tuple of the representations of the output items.

    The applicability of the model to the problems of
    proving program equivalence and correctness is illustrated
    by means of examples.",
  paper = "Abda74b.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Abda81,
  author = "Abdali, S. Kamal and Winkler, Franz",
  title = {{A Lambda-Calculus Model for Generating Verification Conditions}},
  type = "technical report",
  institution = "Rensselaer Polytechnic Institute",
  number = "CS-8104",
```

```
  year = "1981",
  abstract =
    "A lambda-calculus-based method is developed for the automatic
    generation of verification conditions. A programming language is
    specified in which inductive assertions associated with a program
    are incorporated within the body of the program by means of assert
    and maintain-while statements. This programming language includes
    the following features: assignments, conditionals, loops,
    compounds, ALGOL-type block structure. A model is developed which
    consists of rules to translate each statement in this programming
    language into the lamba-calculus. The model is such that the
    lambda-expression translation of any program reduces to a list
    (tuple) of lambda-expression representations of all verification
    conditions of the program. A proof of this property is given.",
  paper = "Abda81.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Abra00,
  author = "Abramov, S.A.",
  title = {{A Note on the Number of Division Steps in the Euclidean
            Algorithm}},
  journal = "ACM SIGSAM",
  volume = "34",
  number = "4",
  year = "2000",
  paper = "Abra00.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Abra95,
  author = "Abrams, Marshall D. and Zelkowitz, Marvin V.",
  title = {{Striving for Correctness}},
  journal = "Computers and Security",
  volume = "14",
  year = "1995",
  pages = "719-738",
  paper = "Abra95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Abra92,
  author = "Abramsky, S. and Gabbay, Dov M. and Maibaum, T.S.E",
  title = {{Handbook of Logic in Computer Science (Volume 2)}},
  publisher = "Oxford Science Publications",
  year = "1992",
  isbn = "0-19-853761-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Akba08a,
  author = "Akbarpour, Behzad and Paulson, Lawrence C.",
  title = {{MetiTarski: An Automatic Prover for the Elementary Functions}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
  abstract =
    "Many inequalities involving the functions ln, exp, sin, cos, etc.,
    can be proved automatically by MetiTarski: a resolution theorem prover
    (Metis) modified to call a decision procedure (QEPCAD) for the theory
    of real closed fields. The decision procedure simplifies clauses by
    deleting literals that are inconsistent with other algebraic facts,
    while deleting as redundant clauses that follow algebraically from
    other clauses.  MetiTarski includes special code to simplify
    arithmetic expressions.",
  paper = "Akba08a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Alte04,
  author = "Altenkirch, Thorsten",
  title = {{$\lambda$-calculus and types}},
  comment = "Lecture Notes; APPSEM Spring School 2004",
  link = "\url{http://www.cs.nott.ac.uk/~psztxa/publ/mgs04.pdf}",
  year = "2004",
  paper = "Alte04.pdf",
  keywords = "printed",
}
```

———-

— **axiom.bib** —

```
@misc{Alte05,
```

```
  author = "Altenkirch, Thorsten and McBride, Conor",
  title = {{Why Dependent Types Matter}},
  link = "\url{http://www.cs.nott.ac.uk/~psztxa/publ/ydtm.pdf}",
  year = "2005",
  abstract =
    "We exhibit the rationale behind the design of Epigram, a dependently
    typed programming language and interactive program development system,
    using refinements of a well known program -- merge sort -- as a running
    example. We discuss its relationship with other proposals to introduce
    aspects of dependent types into functional programming languages and
    sketch some topics for further work in this area.",
  paper = "Alte05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Ande78,
  author = "Anderson, E.R. and Belz, F.C. and Blum, E.K.",
  title = {{Extending an Implementation Language to a Specification Language}},
  journal = "LNCS",
  volume = "75",
  year = "1978",
  paper = "Ande78.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Anon18,
  author = "Anonymous",
  title = {{A Compiler From Scratch}},
  link = "\url{https://www.destroyallsoftware.com/screencasts/catalog/a-compiler-from-scratch}",
  year = "2018"
}
```

———-

— **axiom.bib** —

```
@article{Appe87,
  author = "Appel, Andrew W. and MacQueen, David B.",
  title = {{A Standard ML Compiler}},
  journal = "LNCS",
  volume = "274",
  pages = "301-324",
  year = "1987",
```

```
  abstract =
    "Standard ML is a major revision of earlier dialects of the functional
    language ML. We describe the first compiler written for Standard ML in
    Standard ML.  The compiler incorporates a number of novel features and
    techniques, and is probably the largest system written to date in
    Standard ML.

    Great attention was paid to modularity in the construction of the
    compiler, leading to a successful large-scale test of the modular
    capabilities of Standard ML.  The front end is useful for purposes
    other than compilation, and the back end is easily retargetable (we
    have code generators for the VAX and MC68020). The module facilities
    of Standard ML were taken into account early in the design of the
    compiler, and they particularly influenced the environment management
    component of the front end. For example, the symbol table structure is
    designed for fast access to opened structures.

    The front end of the compiler is a single phase that integrates
    parsing, environment management, and type checking. The middle end
    uses a sophisticated decision tree scheme to produce efficient
    pattern matching code for functions and case expressions. The abstract
    syntax produced by the front end is translated into a simple
    lambda-calculus-based intermediate representation that lends itself to
    easy case analysis and optimization in the code generator. Special
    care was taken in designing the mntime data structures for fast
    allocation and garbage collection.

    We describe the overall organization of the compiler and present some
    of the data representations and algorithms used in its various
    phases. We conclude with some lessons learned about the ML language
    itself and about compilers for modem functional languages.",
  paper = "Appe87.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Appe88,
  author = "Appel, Andrew W. and Jim, Trevor",
  title = {{Continuation-Passing, Closure-Passing Style}},
  institution = "Princston University",
  year = "1988",
  number = "CS-TR-183-88",
  abstract =
    "We implemented a continuation-passing style (CPS) code generator for
    ML. Our CPS language is represented as an ML datatype in which all
    functions are named and most kinds of ill-formed expressions are
    impossible. We separate the code generation into phases that rewrite
    this representation into ever-simpler forms. Closures are represented
    explicitly as records, so that closure strategies can be communicated
    from one phase to another. No stack is used. Our benchmark data shows
```

```
    that the new method is an improvement over our previous,
    abstract-machine based code generator.",
  paper = "Appe88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Appe89,
  author = "Appel, Andrew W.",
  title = {{Run-time Tags Aren't Necessary}},
  journal = "Lisp and Symbolic Computation",
  volume = "2",
  pages = "153-162",
  year = "1989",
  publisher = "Kluwer",
  abstract =
    "Many modern programming environments use tag bits at runtime to
    distinguish objects of different types. This is particularly
    common in systems with garbage collection, since the garbage
    collector must be able to distinguish pointers from non-pointers,
    and to learn the length of records pointed to.

    The use of tag bits leads to inefficiency. In addition to the
    obvious space overhead (tag bits and record descriptors occupy
    memory space), there is a time overhead: tag bits must be stripped
    off of data before arithmetic operations are performed, and
    re-attached to the data when it is stored into memory. This takes
    either extra instructions at runtime, or special tag-handling
    hardware, or both.

    This paper shows how the use of tag bits, record descriptor words,
    explicit type parameters, and the like can be avoided in languages
    (like ML) with static polymorphic typechecking. Through a form of
    tag will still be required for user-defined variant records, all
    other type information can be encoded once -- in the program --
    rather than replicated many times in the data. This can lead to
    savings in both space and time.",
  paper = "Appe89.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Appe91,
  author = "Appel, Andrew W. and MacQueen, David B.",
  title = {{Standard ML of New Jersey}},
  booktitle = "Int. Symp. on Prog. Lang. Implementations and Logic
               Programming",
```

```
  publisher = "Springer-Verlag",
  pages = "1-13",
  year = "1991",
  abstract =
    "The Standard ML of New Jersey compiler has been under development for
    five years now.  We have developed a robust and complete environment
    for Standard ML that supports the implementation of large software
    systems and generates efficient code.  The compiler has also served as
    a laboratory for developing novel implementation techniques for a
    sophisticated type and module system, continuation based code
    generation, efficient pattern matching, and concurrent programming
    features.",
  paper = "Appe91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Appe92,
  author = "Appel, Andrew W.",
  title = {{Compiling with Continuations}},
  year = "1992",
  publisher = "Cambridge University Press",
  isbn = "0-521-03311-X"
}
```

———-

— **axiom.bib** —

```
@article{Appe93,
  author = "Appel, Andrew W.",
  title = {{A Critique of Standard ML}},
  journal = "J. Functional Programming",
  volume = "3",
  number = "4",
  pages = "391-429",
  year = "1993",
  abstract =
    "Standard ML is an excellent language for many kinds of
    programming. It is safe, efficient, suitably abstract, and
    concise. There are many aspects of the language that work
    well. However, nothing is perfect: Standard ML has a few
    shortcomings. In some cases there are obvious solutions, and in
    other cases further research is required.",
  paper = "Appe93.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Appe98,
  author = "Appel, Andrew W.",
  title = {{Modern Compiler Implementation in ML}},
  year = "1998",
  publisher = "Cambridge University Press",
  isbn = "0-521-60764-7",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inbook{Aptx99,
  author = "Apt, Krysztof R. and Bezem, Marc",
  title = {{Formulas as Programs}},
  booktitle = "The Logic Programming Paradigm",
  year = "1999",
  publisher = "Springer Berlin Heidelberg",
  pages = "75-107",
  isbn = "978-3-642-60085-2",
  abstract =
    "We provide here a computational interpretation of first-order
    logic based on a constructive interpretation of satisfiability
    w.r.t. a fixed but arbitrary interpretation. In this approach the
    formulas themselves are programs. This contrasts with the
    so-called formulas as types approach in which the proofs of the
    formulas are typed terms that can be taken as programs. This view
    of computing is inspired by logic programming and constraint logic
    programming but differs from them in a number of crucial aspects.

    Formulas as programs is argued to yield a realistic approach to
    programming that has been realized in the implemented programming
    language Alma-0 Apt, Brunekreef, Partington and Schaerf (1998)
    that combines the advantages of imperative and logic
    programming. The work here reported can also be used to reason
    about the correctness of non-recursive Alma-0 programs that do not
    include destructive assignment.",
  paper = "Aptx99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@incollection{Aspi05,
  author = "Aspinall, David and Hofmann, Martin",
  title = {{Dependent Types}},
  booktitle = "Advanced Topics in Types and Programming Languages",
```

```
  year = "2005",
  publisher = "MIT Press",
  isbn = "0-262-16228-8",
  pages = "45-46",
  paper = "Aspi05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Atke18a,
  author = "Atkey, Robert",
  title = {{Parameterised Notions of Computation}},
  link = "\url{https://bentnib.org/paramnotions-jfp.pdf}",
  year = "2018",
  abstract =
    "Moggis Computational Monads and Power et al s equivalent notion of
    Freyd category have captured a large range of computational effects
    present in programming languages. Examples include non-termination,
    non-determinism, exceptions, continuations, side-effects and
    input/output. We present generalisations of both computational monads
    and Freyd categories, which we call parameterised monads and
    parameterised Freyd categories, that also capture computational
    effects with parameters. Examples of such are composable
    continuations, side-effects where the type of the state varies and
    input/output where the range of inputs and outputs varies. By also
    considering structured parameterisation, we extend the range of
    effects to cover separated side-effects and multiple independent
    streams of I/O. We also present two typed $\lambda$-calculi that
    soundly and completely model our categorical definitions  with and
    without symmetric monoidal parameterisation  and act as prototypical
    languages with parameterised effects.",
  paper = "Atke18a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Avig18a,
  author = "Avigad, Jeremy",
  title = {{The Mechanization of Mathematics}},
  journal = "Notices of the AMS",
  volume = "65",
  number = "6",
  year = "2018",
  pages = "681-690",
  paper = "Avig18a.pdf",
  keywords = "printed, DONE"
```

```
}
```

———-

1.17.2 B

— **axiom.bib** —

```
@article{Back81,
  author = "Back, R.J.R",
  title = {{On Correct Refinement of Programs}},
  journal = "J. Computer and System Sciences",
  volume = "23",
  number = "1",
  pages = "49-68",
  year = "1981",
  abstract =
    "The stepwise refinement technique is studied from a mathematical
    point of view. A relation of correct refinement between programs is
    defined, based on the principle that refinement steps should be
    correctness preserving. Refinement between programs will therefore
    depend on the criterion of program correctness used. The application
    of the refinement relation in showing the soundness of different
    techniques for refining programs is discussed. Special attention is
    given to the use of abstraction in program construction. Refinement
    with respect to partial and total correctness will be studied in more
    detail, both for deterministic and nondeterministic programs. The
    relationship between these refinement relations and the approximation
    relation of fixpoint semantics will be studied, as well as the
    connection with the predicate transformers used in program
    verification.",
  paper = "Back81.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Bail11,
  author = "Bailey, David H. and Borwein, Jonathan M.",
  title = {{High-precision Numerical Integration: Progress and Challenges}},
  journal = "J. Symbolic Computation",
  volume = "46",
  number = "7",
  pages = "741-754",
  year = "2011",
  abstract =
    "One of the most fruitful advances in the field of experimental
    mathematics has been the development of practical methods for very
    high-precision numerical integration, a quest initiated by Keith
```

```
    Geddes and other researchers in the 1980s and 1990s.These
    techniques, when coupled with equally powerful integer relation
    detection methods, have resulted in the analytic evaluation of many
    integrals that previously were beyond the realm of symbolic
    techniques.This paper presents a survey of the current state-of-the-art
    in this area (including results by the present authors and others),
    mentions some new results, and then sketches what challenges lie
    ahead.",
  paper = "Bail11.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bail14,
  author = "Bailey, David H. and Borwein, Jonathan M. and
            Kaiser, Alexander D.",
  title = {{Automated Simplification of Large Symbolic Expressions}},
  journal = "J. Symbolic Computation",
  volume = "60",
  pages = "120-136",
  year = "2014",
  abstract =
    "We present a set of algorithms for automated simplification of
    symbolic constants of the form
    $\sum_i\alpha_i x_i$ with $\alpha_i$ rational and $x_i$
    complex. The included algorithms, called SimplifySum and
    implemented in Mathematica , remove redundant terms, attempt to make
    terms and the full expression real, and remove terms using repeated
    application of the multipair PSLQ integer relation detection
    algorithm. Also included are facilities for making substitutions
    according to user-specified identities. We illustrate this toolset by
    giving some real-world examples of its usage, including one, for
    instance, where the tool reduced a symbolic expression of
    approximately 100 000 characters in size enough to enable manual
    manipulation to one with just four simple terms.",
  paper = "Bail14.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bals91,
  author = "Balsters, Herman and Fokkinga, Maarten M.",
  title = {{Subtyping can have a simple semantics}},
  journal = "Theoretical Computer Science",
  volume = "87",
  pages = "81-96",
  year = "1991",
  abstract =
```

```
    "Consider a first order typed language, with semantics
     $\llbracket~\rrbracket$ for expressions and types. Adding
     subtyping means that a partial order $\le$ on types is defined
     and that the typing rules are extended to the effect that
     expression $e$ has type $\tau$ whenever $e$ has type $\sigma$ and
     $\sigma \le \tau$. We show how to adapt the semantics
     $\llbracket~\rrbracket$ in a {\sl simple set theoretic way},
     obtaining a semantics $\llbracket~\rrbracket$ that satisfies, in
     addition to some obvious requirements, also the property
     $\llbracket\sigma\rrbracket \subseteq \llbracket\tau\rrbracket$,
     whenever $\sigma \le \tau$.",
  paper = "Bals91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@incollection{Bare92,
  author = "Barendregt, Henk",
  title = {{Lambda Calculi with Types}},
  booktitle = "Handbook of Logic in Computer Science (vol 2)",
  publisher = "Oxford University Press",
  year = "1992",
  paper = "Bare92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bare02,
  author = "Barendregt, Henk and Barendsen, Erik",
  title = {{Autorkic Computations in Formal Proofs}},
  journal = "J. Automated Reasoning",
  volume = "28",
  pages = "321-336",
  year = "2002",
  abstract =
    "Formal proofs in mathematics and computer science are being studied
    because these objects can be verified by a very simple computer
    program. An important open problem is whether these formal proofs can
    be generated with an effort not much greater than writing a
    mathematical paper in, say, LATEX. Modern systems for proof
    development make the formalization of reasoning relatively
    easy. However, formalizing computations in such a manner that the
    results can be used in formal proofs is not immediate. In this paper
    we show how to obtain formal proofs of statements such as Prime(61) in
    the context of Peano arithmetic or $(x + 1)(x + 1) = x 2 + 2x + 1$ in
    the context of rings. We hope that the method will help bridge the gap
    between the efficient systems of computer algebra and the reliable
```

```
    systems of proof development.",
  paper = "Bare02.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Bare05,
  author = "Barendregt, Henk and Wiedijk, Freek",
  title = {{The Challenge of Computer Mathematics}},
  journal = "Phil. Trans. Royal Society",
  volume = "363",
  pages = "2351-2375",
  year = "2005",
  abstract =
    "Progress in the foundations of mathematics has made it possible to
    formulate all thinkable mathematical concepts, algorithms and proofs
    in one language and in an impeccable way. This is not in spite of, but
    partially based on the famous results of Godel and Turing. In this way
    statements are about mathematical objects and algorithms, proofs show
    the correctness of statements and computations, and computations are
    dealing with objects and proofs. Interactive computer systems for a
    full integration of defining, computing and proving are based on
    this. The human defines concepts, constructs algorithms and provides
    proofs, while the machine checks that the definitions are well formed
    and the proofs and computations are correct. Results formalized so far
    demonstrate the feasibility of this computer mathematics. Also there
    are very good applications. The challenge is to make the systems more
    mathematician-friendly, by building libraries and tools. The eventual
    goal is to help humans to learn, develop, communicate, referee and
    apply mathematics.",
  paper = "Bare05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Bare13,
  author = "Barendregt, Henk and Dekkers, Wil and Statman, Richard",
  title = {{Lambda Calculus with Types}},
  publisher = "Cambridge University Press",
  year = "2013",
  isbn = "978-0-521-76614-2",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Barn08,
  author = "Barnett, Michael P.",
  title = {{Reasoning in Symbolic Computation}},
  journal = "ACM Communications in Symbolic Algebra",
  volume = "42",
  number = "1",
  year = "2008",
  abstract =
    "I discuss notations for some styles of mathematical reasoning that
    include analogy. These notations extend the conventions of the
    mathematica package mathscape that I reported recently in the
    Journal of Symbolic Computation. The paper introduces the reasoning
    objects that I call 'precursors' and 'consequences lists'.",
  paper = "Barn08.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Barr96a,
  author = "Barras, Bruno",
  title = {{Coq en Coq}},
  institution = "INRIA",
  year = "1996",
  type = "Research Report",
  number = "inria-00073667",
  comment = "French",
  abstract =
    "The essential step of the formal verification of a proof-checker such
    as Coq is the verification of its kernel: a type-checker for the
    {\sl Calculus of Inductive Constructions} (CIC) whihc is its underlying
    formalism. The present work is a first small-scale attempt on a
    significative fragment of CIC: the Calculus of Constructions (CC). We
    formalize the definition and the metatheory of (CC) in Coq. In
    particular, we prove strong normalization and decidability of type
    inference. From the latter proof, we extract a certified {\sl Caml
    Light} program, which performs type inference (or type-checking) for
    an arbitrary typing judgement in CC. Integrating this program in a
    larger system, including a parser and pretty-printer, we obtain a
    stand-alone proof-checker, called {\sl CoC}, the {\sl Calculus of
    Constructions}. As an example, the formal proof of Newman's lemma,
    build with Coq, can be re-verified by {\sl CoC} with reasonable
    performance.",
  paper = "Barr96a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Barr18,
  author = "Barras, Bruno and Werner, Benjamin",
  title = {{Coq in Coq}},
  link =
"\url{http://www.lix.polytechnique.fr/Labo/Bruno.Barras/publi/coqincoq.pdf}",
  comment = "https://github.com/coq-contribs/coq-in-coq",
  abstract =
    "We formalize the definition and the metatheory of the Calculus of
    Constructions (CC) using the proof assistant Coq. In particular,
    we prove strong normalization and decidability of type
    inference. From the latter proof, we extract a certified Objective
    Caml program which performs type inference in CC and use this code
    to build a small-scale certified proof-checker.",
  paper = "Barr18.pdf",
  keywords = "printed, reviewed"
}
```

———-

— **axiom.bib** —

```
@book{Basu10,
  author = "Basu, Saugata and Pollack, Richard and Roy, Marie-Francoise",
  title = {{Algorithms in Real Algebraic Geometry}},
  publisher = "Springer-Verlag",
  year = "2010",
  isbn = "978-3-642-06964-2"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bent07,
  author = "Benton, Nick and Zarfaty, Uri",
  title = {{Formalizing and Verifying Semantic Type Soundness of a
            Simple Compiler}},
  booktitle = "Principles and Practice of Declarative Programming",
  publisher = "ACM",
  pages = "1-12",
  year = "2007",
  abstract =
    "We describe a semantic type soundness result, formalized in the Coq
    proof assistant, for a compiler from a simple imperative language with
    heap-allocated data into an idealized assembly language. Types in the
    high-level language are interpreted as binary relations, built using
    both second-order quantification and a form of separation structure,
    over stores and code pointers in the low-level machine.",
  paper = "Bent07.pdf",
  keywords = "printed"
```

```
}
```

———-

— **axiom.bib** —

```
@book{Bees80,
  author = "Beeson, Michael J.",
  title = {{Foundations of Constructive Mathematics}},
  publisher = "Springer-Verlag",
  year = "1980",
  isbn = "3-540-12173-0",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Bees80a,
  author = "Beeson, Michael",
  title = {{Formalizing Constructive Mathematics: Why and How?}},
  journal = "Lecture Notes in Mathematics",
  volume = "873",
  pages = "146-190",
  year = "1980",
  paper = "Bees80a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Beni95,
  author = "Benini, Marco",
  title = {{Barendregt's $\lambda$-Cube in Isabelle}},
  link = "\url{}",
  year = "1995",
  abstract =
    "We present an implementation of Barendregts $\lambda$-Cube in the
    Isabelle proof system.  Isabelle provides many facilities for
    developing a useful specification and proving environment from the
    basic formulation of formal systems.  We used those facilities to
    provide an environment where the user can describe problems and derive
    proofs interactively.",
  paper = "Beni95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bonn18,
  author = "Bonnaire-Sergeant, Ambrose",
  title = {{Are unsound type systems wrong?}},
  link =
"\url{http://frenchy64.github.io/2018/04/07/unsoundness-in-untyped-types.html}",
}
```

———-

— **axiom.bib** —

```
@article{Boye75,
  author = "Boyer, Robert S. and Moore, J Strother",
  title = {{Proving Theorems About LISP Functions}},
  journal = "J. ACM",
  volume = "22",
  number = "1",
  pages = "129-144",
  year = "1975",
  abstract =
    "Program verification is the idea that properties of programs can be
    precisely stated and proved in the mathematical sense. In this paper,
    some simple heuristics combining evaluation and mathematical induction
    are describe, which the authors have implemented in a program that
    automatically proves a wide variety of theorems about recursive LISP
    functions. The method the program uses a generate induction formulas
    is described at length. The theorems proved by the program include
    that REVERSE is its own inverse and that a particular SORT program is
    correct. A list of theorems proved by the program is given.",
  paper = "Boye75.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Boye84,
  author = "Boyer, Robert S. and Moore, J Strother",
  title = {{Proof-Checking, Theorem-Proving, and Program Verification}},
  journal = "Contemporary Mathematics",
  volume = "29",
  year = "1984",
  abstract =
    "This article consists or three parts: a tutorial introduction to a
    computer program that proves theorems by induction; a brief
    description or recent applications or that theorem-prover; and a
    discussion or several nontechnical aspects of the problem or building
    automatic theorem-provers. The theorem-prover described has proved
    theorems such as the uniqueness of prime factorizations, Fermat's
```

```
    theorem, and the recursive unsolvability or the halting problem.

    The article is addressed to those who know nothing about automatic
    theorem-proving but would like a glimpse or one such system. This
    article definitely does not provide a balanced view or all automatic
    theorem-proving, the literature of which is already rather large and
    technica1. Nor do we describe the details or our theorem-proving
    system, but they can be round in the books, articles, and technical
    reports that we reference.

    In our opinion, progress in automatic theorem-proving is largely a
    function or the mathematical ability or those attempting to build such
    systems. We encourage good mathematicians to work in the field.",
  paper = "Boye84.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Boye84a,
  author = "Boyer, Robert S. and Moore, J Strother",
  title = {{A Mechanical Proof of the Turing Completeness of Pure Lisp}},
  journal = "Contemporary Mathematics",
  volume = "29",
  year = "1984",
  abstract =
    "We describe a proof by a computer prosram of theTuring completeness
    of a computational paradigm akin to Pure LISP. That is, we defme
    formally the notions of a Turing machine and of a version of Pure
    LISP and prove that anything that can be computed by a Turing machine
    can be computed by LISP. While this result is straightforward, we
    believe this is the rarst instance of a machine proving the Turing
    completeness of another computational paradigm.",
  paper = "Boye84a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Brod94,
  author = "Broda, Krysia and Eisenbach, Susan and Khoshnevisan, Hessam
            and Vickers, Steve",
  title = {{Reasoned Programming}},
  publisher = "Imperial College",
  year = "1994",
  paper = "Brod94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Broo87a,
  author = "Brooks, Rodney A.",
  title = {{Intelligence Without Representation}},
  journal = "Artificial Intelligence",
  volume = "47",
  year = "1991",
  pages = "139-159",
  abstract =
    "Artificial intelligence research has foundered on the issue of
    representation. When intelligence is approached in an incremental
    manner, with strict reliance on interfacing to the real world through
    perception and action, reliance on representation disappears.  In this
    paper we outline our approach to incrementally building complete
    intelligent Creatures.  The fundamental decomposition of the
    intelligent system is not into independent information processing
    units which must interface with each other via representations.
    Instead, the intelligent system is decomposed into independent and
    parallel activity producers which all interface directly to the world
    through perception and action, rather than interface to each other
    particularly much.  The notions of central and peripheral systems
    evaporateeverything is both central and peripheral. Based on these
    principles we have built a very successful series of mobile robots
    which operate without supervision as Creatures in standard office
    environments.",
  paper = "Broo87a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Buch02,
  author = "Buchberger, Bruno",
  title = {{Computer Algebra: The End of Mathematics?}},
  journal = "ACM SIGSAM",
  volume = "36",
  number = "1",
  year = "2002",
  abstract =
    "Mathematical software systems, such as Mathematica, Maple, Derive,
    and so on, are substantially based on enormous advances in the area of
    mathematics known as Computer Algebra or Symbolic Mathematics. In
    fact, everything taught in high school and in the first semesters of a
    university mathematical education, is available in these systems 'at
    the touch of the button'. Will mathematics become unnecessary because
    of this? In the three sections of this essay, I answer this question
    for non-mathematicians, for mathematicians and for (future) students
    of mathematics.",
  paper = "Buch02.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Byte79,
  author = "Byte Publications",
  title = {{LISP}},
  link = "\url{https://ia802603.us.archive.org/30/items/byte-magazine-1979-08/1979_08_BYTE_04-08_LISP.pdf}",
  publisher = "McGraw-Hill",
  year = "1979",
  paper = "Byte79.pdf"
}
```

———-

1.17.3 C

— **axiom.bib** —

```
@inproceedings{Cald97,
  author = "Caldwell, James L.",
  title = {{Moving proofs-as-programs into practice}},
  booktitle = "Automated Software Engineering",
  publisher = "IEEE",
  year = "1997",
  abstract =
    "Proofs in the Nuprl system, an implementation of a constructive
    type theory, yield correct-by-construction programs.  In this
    paper a new methodology is presented for extracting efficient and
    readable programs from inductive proofs. The resulting extracted
    programs are in a form suitable for use in hierarchical
    verifications in that they are amenable to clean partial evaluation
    via extensions to the Nuprl rewrite system. The method is based on two
    elements: specifications written with careful use of the Nuprl
    set-type to restrict the extracts to strictly computational content;
    and on proofs that use induction tactics that generate extracts
    using familiar fixed-point combinators of the untyped lambda
    calculus. In this paper the methodology is described and its
    application is illustrated by example.",
  paper = "Cald97.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Camp02,
```

```
  author = "Campbell, J.A.",
  title = {{Indefinite Integration as a Testbed for Developments in
           Multi-agent Systems}},
  journal = "LNCS",
  volume = "2385",
  year = "2002",
  abstract =
    "Coordination of multiple autonomous agents to solve problems that
    require each of them to contribute their limited expertise in the
    construction of a solution is often ensured by the use of numerical
    methods such as vote-counting, payoff functions, game theory and
    economic criteria. In areas where there are no obvious numerical methods
    for agents to use in assessing other agents contributions, many
    questions still remain open for research. The paper reports a study of
    one such area: heuristic indefinite integration in terms of agents with
    different single heuristic abilities which must cooperate in finding
    indefinite integrals. It examines the reasons for successes and lack
    of success in performance, and draws some general conclusions about
    the usefulness of indefinite integration as a field for realistic
    tests of methods for multi-agent systems where the usefulness of
    'economic' criteria is limited. In this connection, the role of
    numerical taxonomy is emphasised.",
  paper = "Camp02.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Card84,
  author = "Cardelli, Luca",
  title = {{A Semantics of Multiple Inheritance}},
  journal = "LNCS",
  volume = "173",
  year = "1984",
  paper = "Card84.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Care07,
  author = "Carette, Jacques and Farmer, William M. and Sorge, Volker",
  title = {{A Rational Reconstruction of a System for Experimental
           Mathematics}},
  booktitle = "14th Workshop on Automated Reasoning",
  publisher = "unknown",
  year = "2007",
  abstract =
```

```
    "In previous papers we described the implementation of a system
    which combines mathematical object generation, transformation and
    filtering, conjecture generation, proving and disproving for
    mathematical discovery in non-associative algebra. While the system
    has generated novel, fully verified theorems, their construction
    involved a lot of ad hoc communication between disparate systems. In
    this paper we carefully reconstruct a specification of a sub-process
    of the original system in a framework for trustable communication
    between mathematics systems put forth by us. It employs the concept
    of biform theories that enables the combined formalisation of the
    axiomatic and algorithmic theories behind the generation
    process. This allows us to gain a much better understanding of the
    original system, and exposes clear generalisation opportunities.",
  paper = "Care07.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Cart76,
  author = "Cartwright, Robert",
  title = {{A Practical Formal Semantic Definition and Verification
            System for Typed LISP}},
  school = "Stanford Artificial Intelligence Labs",
  year = "1976",
  abstract =
    "Despite the fact that computer scientists have developed a variety of
    formal methods for proving computer programs correct, the formal
    verification of a non-trivial program is still a formidable
    task. Moreover, the notion of proof is so imprecise in most existing
    verification systems, that the validity of the proofs generated is
    open to question. With an aim toward rectifying these problems, the
    research discussed in this dissertation attempts to accomplish the
    following objectives:
    \begin{enumerate}
    \item To develop a programming language which is sufficiently powerful
    to express many interesting algorithms clearly and succintly, yet
    simple enough to have a tractable formal semantic definition
    \item To completely specify both proof theoretic and model theoretic
    formal semantics for this language using the simplest possible
    abstractions
    \item To develop an interactive program verification system for the
    language which automatically performs as many of the straightforward
    steps in a verification as possible
    \end{enumerate}",
  paper = "Cart76.pdf",
}
```

———-

— **axiom.bib** —

```
@article{Cart78,
  author = "Cartwright, Robert and McCarthy, John",
  title = {{Recursive Programs as Functions in a First Order Theory}},
  journal = "LNCS",
  volume = "75",
  pages = "576-629",
  year = "1978",
  abstract =
    "Pure Lisp style recursive function programs are represented in a new
    way by sentences and schemata of first order logic. This permits easy
    and natural proofs of extensional properties of such programs by
    methods that generalize structural induction. It also systematizes
    known methods such as recursion induction, subgoal induction,
    inductive assertions by interpreting them as first order axiom
    schemata. We discuss the metatheorems justifying the representation
    and techniques for proving facts about specific programs. We also give
    a simpler version of the GiSdeI-Kleene way of representing computable
    functions by first order sentences.",
  paper = "Cart78.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Cart84,
  author = "Cartwright, Robert",
  title = {{Recursive Programs as Definitions in First-Order Logic}},
  journal = "SIAM J. Computing",
  volume = "13",
  number = "2",
  pages = "374-408",
  year = "1984",
  abstract =
    "Despite the reputed limitations of first order logic, it is easy to
    state and prove almost all interesting properties of recursive
    programs within a simple first order theory, by using an approach we
    call 'first order programming logic'. Unlike higher order logics based
    on fixed-point induction, first order programming logic is founded on
    deductive principles that are familiar to most programmers. Informal
    structural induction arguments (such as termination proofs for LISP
    append, McCarthys 91-function, and Ackermanns function) have direct
    formalizations within the system.

    The essential elements of first order programming logic are:
    \begin{enumerate}
    \item The data domain $D$ must be a finitely generated set that
    explicitly includes the 'undefined' object $\bot$
    (representing nontermination) as well as ordinary data objects.
    \item Recursive programs over $D$ are treated as logical definitions
    augmenting a first order theory of the data domain.
```

```
    \item The interpretation of a recursive program is the least
    fixed-point of the functional corresponding to the program.
    \end{enumerate}

    Since the data domain $D$ is a finitely generated set, the first order
    axiomatization of $D$ includes a structural induction axiom scheme. This
    axiom scheme serves as the fundamental 'proof rule' of first order
    programming logic.

    The major limitation of first order programming logic is that every
    fixed-point of the functional corresponding to a recursive program is
    an acceptable interpretation for the program. The logic fails to
    capture the notion of least fixed-point. To overcome this limitation,
    we present a simple, effective procedure for transforming an arbitrary
    recursive program into an equivalent recursive program that has a
    unique fixed-point, yet retains the logical structure of the
    original. Given this transformation technique, it is our experience
    that first order programming logic is sufficiently powerful to prove
    almost any property of practical interest about the functions computed
    by recursive programs.",
  paper = "Cart84.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Char08,
  author = "Charnley, John and Colton, Simon",
  title = {{A Global Workspace Framework for Combining Reasoning Systems}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
  paper = "Char08.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Chen08a,
  author = "Chen, William Y.C. and Paule, Peter and Saad, Husam L.",
  title = {{Converging to Gosper's Algorithm}},
  journal = "Advances in Applied Mathematics",
  volume = "41",
  number = "3",
  pages = "351-364",
  year = "2008",
  abstract =
    "Given two polynomials, we find a convergence property of the GCD of
    the rising factorial and the falling factorial. Based on this
```

```
    property, we present a unified approach to computing the universal
    denominators as given by Gosper's algorithm and Abramov's algorithm
    for finding rational solutions to linear difference equations with
    polynomial coefficients. Our approach easily extends to the
    q-analogues.",
  paper = "Chen08a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Chli07,
  author = "Chlipala, Adam J.",
  title = {{A Certified Type-Preserving Compiler from the Lambda Calculus
           to Assembly Language}},
  journal = "ACM SIGPLAN PLDI'07",
  volume = "42",
  number = "6",
  year = "2007",
  pages = "54-65",
  abstract =
    "We present a certified compiler from the simply-typed lambda calculus
    to assembly language. The compiler is certified in the sense that it
    comes with a machine-checked proof of semantics preservation,
    performed with the Coq proof assistant. The compiler and the terms of
    its several intermediate languages are given dependent types that
    guarantee that only well-typed programs are representable. Thus, type
    preservation for each compiler pass follows without any significant
    'proofs' of the usual kind. Semantics preservation is proved based on
    denotational semantics assigned to the intermediate languages. We
    demonstrate how working with a type-preserving compiler enables
    type-directed proof search to discharge large parts of our proof
    obligations automatically.",
  paper = "Chli07.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Clin72,
  author = "Clint, M. and Hoare, C.A.R",
  title = {{Program Proving: Jumps and Functions}},
  journal = "Acta Informatica",
  volume = "1",
  pages = "214-224",
  year = "1972",
  abstract =
    "Proof methods adequate for a wide range of computer programs have
    been expounded in [1] and [2]. This paper develops a method suitable
```

```
    for programs containing functions, and a certain kind of jump. The
    method is illustrated by the proof of a useful and efficient program
    for table lookup by logarithmic search.",
  paper = "Clin72.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Cohn79,
  author = "Cohn, Avra Jean",
  title = {{Machine Assisted Proofs of Recursion Implementation}},
  school = "University of Edinburgh",
  year = "1979",
  link = "\url{https://www.era.lib.ed.ac.uk/handle/1842/6643}",
  abstract =
    "Three studies in the machine assisted proof of recursion
    implementation are described. The verification system used is
    Edinburgh LCF (Logic for Computable Functions). Proofs are generated,
    in LCF, in a goal-oriented fashion by the application of strategies
    reflecting informal proof plans. LCF is introduced in Chapter 1. We
    present three case studies in which proof strategies are developed and
    (except in the third) tested in LCF. Chapter 2 contains an account of
    the machine generated proofs of three program transformations (from
    recursive to iterative function schemata). Two of the examples are
    taken from Manna and Waldinger. In each case, the recursion is
    implemented by the introduction of a new data type, e.g., a stack or
    counter. Some progress is made towards the development of a general
    strategy for producing the equivalence proofs of recursive and
    iterative function schemata by machine. Chapter 3 is concerned with
    the machine generated proof of the correctness of a compiling
    algorithm. The formulation, borrowed from Russell, includes a simple
    imperative language with a while and conditional construct, and a low
    level language of labelled statements, including jumps. We have, in
    LCF, formalised his denotational descriptions of the two languages and
    performed a proof of the preservation of the semantics under
    compilation. In Chapter 4, we express and informally prove the
    correctness of a compiling algorithm for a language containing
    declarations and calls of recursive procedures. We present a low level
    language whose semantics model a standard activation stack
    implementation. Certain theoretical difficulties (connected with
    recursively defined relations) are discussed, and a proposed proof in
    LCF is outlined. The emphasis in this work is less on proving original
    theorems, or even automatically finding proofs of known theorems, than
    on (i) exhibiting and analysing the underlying structure of proofs,
    and of machine proof attempts, and (ii) investigating the nature of
    the interaction (between a user and a computer system) required to
    generate proofs mechanically; that is, the transition from informal
    proof plans to behaviours which cause formal proofs to be performed.",
  paper = "Cohn79.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Como91a,
  author = "Comon, Hubert",
  title = {{Disunification: a Survey}},
  booktitle = "Computational Logic",
  publisher = "MIT Press",
  year = "1991",
  abstract =
    "Solving an equation in an algebra of terms is known as
    unification. Solving more complex formulas combining equations and
    involving in particular negation is called {\sl
    disunification}. With such a broad definition, many works fall
    into the scope of disunification. The goal of this paper is to
    survey these works and bring them together in a same framework.",
  paper = "Como91a.ps"
}
```

———-

— **axiom.bib** —

```
@incollection{Como01,
  author = "Comon, Hubert",
  title = {{Inductionless Induction}},
  booktitle = "Handbook of Automated Reasoning",
  comment = "Chapter 14",
  publisher = "Elsevier",
  year = "2001",
  paper = "Como01.ps"
}
```

———-

— **axiom.bib** —

```
@article{Cons92,
  author = "Constable, Robert L.",
  title = {{Formal Theories and Software Systems: Fundamental
            Connections between Computer Science and Logic}},
  journal = "LNCS",
  volume = "653",
  pages = "105-127",
  year = "1992",
  abstract =
    "A formal Theory of Logics is sketched using concepts from a
    modern proof development sysmte (like Nuprl, Coq or other such
    software systems). The Theory can be applied to understanding
    these software systems, and the application suggests a design
    principle called the {\sl theories-as-systems notion}.
```

```
    Applications of the Theory to automated reasoning have led to
    an empirical study of the notion of {\sl obvious inference}.
    Experimental results are cited to explain key constants of a
    scientific theor of {\sl obvious inference}. The constants appear
    in what is called here the {\sl deBruijn equation}.",
  paper = "Cons92.pdf",
  keywords = "printed",
}
```

———-

— **axiom.bib** —

```
@incollection{Cons03,
  author = "Constable, Robert L.",
  title = {{Naive Computational Type Theory}},
  booktitle = "Proof and System Reliability",
  publisher = "unknown",
  link = "\url{http://www.nuprl.org/documents/Constable/naive.pdf}",
  year = "2003",
  paper = "Cons03.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Coqu88,
  author = "Coquuand, Thierry and Huet, Gerard",
  title = {{The Calculus of Constructions}},
  booktitle = "Information and Computation, Volume 76",
  year = "1988",
  publisher = "Academic Press",
  paper = "Coqu88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Cram14,
  author = "Cramer, Marcos",
  title = {{Modelling the usage of partial functions and undefined
            terms using presupposition theory}},
  year = "2014",
  isbn = "978-1-84890-130-8",
  publisher = "College Publications",
  pages = "71-88",
  abstract =
    "We describe how the linguistic theory of presuppositions can be used
```

```
    to analyse and model the usage of partial functions and undefined
    terms in mathematical texts. We compare our account to other accounts
    of partial functions and undefined terms, showing how our account
    models the actual usage of partial functions and undefined terms more
    faithfully than existing accounts. The model described in this paper
    has been developed for the Naproche system, a computer system for
    proof-checking mathematical texts written in controlled natural
    language, and has largely been implemented in this system.",
  paper = "Cram14.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Crar99,
  author = "Crary, Karl and Harper, Robert and Puri, Sidd",
  title = {{What is a Recursive Module}},
  booktitle = "Conf. on Programming Language Design and Implementation",
  year = "1999",
  link = "\url{https://www.cs.cmu.edu/~crary/papers/1999/recmod/recmod.dvi}",
  abstract =
    "A hierarchical module system is an effective tool for structuring
    large programs. Strictly hierarchical module systems impose an
    acyclic ordering on import dependencies among program units. This
    can impede modular programming by forcing mutually-dependent
    components to be consolidated into a single module. Recently there
    have been several proposals for module systems that admit cyclic
    dependencies, but it is not clear how these proposals relate to
    one another, nor how one mught integrate them into an expressive
    module system such as that of ML.

    To address this question we provide a type-theoretic analysis of
    the notion of a recursive module in the context of a
    ``phase-distinction'' formalism for higher-order module
    systems. We extend this calculus with a recursive module mechanism
    and a new form of signature, called a {\sl recurslively dependent
    signature}, to support the definition of recursive modules. These
    extensions are justified by an interpretation in terms of more
    primitive language constructs. This interpretation may also serve
    as a guide for implementation.",
  paper = "Crar99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Crar02,
  author = "Crary, Karl",
```

```
  title = {{Toward a Foundational Typed Assembly Language}},
  institution = "Carnegie Mellon University",
  number = "CMU-CS-02-196",
  year = "2002",
  link = "\url{https://www.cs.cmu.edu/~crary/papers/2003/talt/talt-tr.pdf}",
  abstract =
    "We present the design of a typed assembly language called TALT that
    supports heterogeneous tuples, disjoint sums, arrays, and a general
    account of addressing modes. TALT also implements the von Neumann
    model in which programs are stored in memory, and supports relative
    addressing.  Type safety for execution and for garbage collection are
    shown by machine-checkable proofs. TALT is the first formalized typed
    assembly language to provide any of these features.",
  paper = "Crar02.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Crar08,
  author = "Crary, Karl and Sarkar, Susmit",
  title = {{Foundational Certified Code in a Metalogical Framework}},
  link = "\url{http://repository.cmu.edu/compsci/470/}",
  year = "2008",
  abstract =
    "Foundational certified code systems seek to prove untrusted programs
    to be safe relative to safety policies given in terms of actual
    machine architectures, thereby improving the systems' flexibility and
    extensibility. Using the Twelf metalogical framework, we have
    constructed a safety policy for the IA-32 architecture with a trusted
    runtime library. The safety policy is based on a formalized
    operational semantics. We have also developed a complete, foundational
    proof that a fully expressive typed assembly language satisfies that
    safety policy",
  paper = "Crar08.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Chur28,
  author = "Church, Alonzo",
  title = {{On the Law of the Excluded Middle}},
  journal = "Bull. of the American Mathematical Society",
  volume = "34",
  number = "1",
  pages = "75-78",
  year = "1928",
  paper = "Chur28.pdf",
  keywords = "printed, DONE"
```

```
}
```

———-

1.17.4 D

— **axiom.bib** —

```
@incollection{Dahl72,
  author = "Dahl, Ole-Johan and Hoare, C.A.R",
  title = {{Hierachical Program Structure}},
  booktitle = "Structured Programming",
  publisher = "Academic Press",
  year = "1972",
  pages = "175-220"
}
```

———-

— **axiom.bib** —

```
@misc{Daly06a,
  author = "Daly, Timothy",
  title = {{Literate Programming}},
  link = "\url{http://lambda-the-ultimate.org/node/1336}",
  year = "2006",
  keywords = "axiomref, TPDref"
}
```

———-

— **axiom.bib** —

```
@misc{Daly18,
  author = "Daly, Timothy",
  title = {{Proving Axiom Sane: Survey of CAS and Proof Cooperation}},
  year = "2018",
  comment = "In Preparation"
}
```

———-

— **axiom.bib** —

```
@article{Darg07,
  author = "Dargaye, Zaynah and Leroy, Xavier",
  title = {{Mechanized Verification of CPS Transformations}},
  journal = "LNCS",
  volume = "4790",
  pages = "211-225",
```

```
  year = "2007",
  abstract =
    "Transformation to continuation-passing style (CPS) is often performed
    by optimizing compilers for functional programming languages.  As part
    of the development and proof of correctness of a compiler for the
    mini-ML functional language, we have mechanically verified the
    correctness of two CPS transformations for a call-by-value
    $\lambda$-calculus with n-ary functions, recursive functions, data
    types and pattern-matching.  The transformations generalize Plotkins
    original call-by-value transformation and Danvy and Nielsens
    optimized transformation, respectively. We used the Coq proof
    assistant to formalize the transformations and conduct and check the
    proofs. Originalities of this work include the use of big-step
    operational semantics to avoid difficulties with administrative
    redexes, and of two-sorted de Bruijn indices to avoid difficulties
    with $\alpha$-conversion.",
  paper = "Darg07.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Davi01,
  author = "Davies, Rowan and Pfenning, Frank",
  title = {{A Modal Analysis of Staged Computation}},
  journal = "J. ACM",
  volume = "48",
  number = "3",
  pages = "555-604",
  year = "2001",
  abstract =
    "We show that a type system based on the intuitionistic modal logic S4
    provides an expressive framework for specifying and analyzing
    computation stages in the context of typed $\lambda$-calculi and
    functional languages. We directly demonstrate the sense in which our
    $\lambda_e^{\rightarrow\Box}$-calculus captures staging, and also give
    a conservative embedding of Nielson and Nielson's two-level functional
    language in our functional language Mini-ML${}^\Box$, thus proving that
    binding-time correctness is equivalent to modal correctness on this
    fragment. In addition, Mini-ML${}^\Box$ can also express immediate
    evaluation and sharing of code across multiple stages, thus supporting
    run-time code generation as well as partial evaluation.",
  paper = "Davi01.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Davi09,
```

```
  author = "Davis, Jared Curran",
  title = {{A Self-Verifying Theorem Prover}},
  school = "University of Texas at Austin",
  year = "2009",
  abstract =
    "Programs have precise semantics, so we can use mathematical proof to
    establish their properties. These proofs are often too large to
    validate with the usual social process of mathematics, so instead we
    create and check them with theorem-proving software. This software
    must be advanced enough to make the proof process tractable, but this
    very sophistication casts doubt upon the whole enterprise: who
    verifies the verifier?

    We begin with a simple proof checker, Level 1, that only accepts
    proofs composed of the most primitive steps, like Instantiation and
    Cut. This program is so straightforward the ordinary, social process
    can establish its soundness and the consistency of the logical
    theory it implements (so we know theorems are always true).

    Next, we develop a series of increasingly capable proof checkers,
    Level 2, Level 3, etc.  Each new proof checker accepts new kinds of
    proof steps which were not accepted in the previous levels. By taking
    advantage of these new proof steps, higher-level proofs can be
    written more concisely than lower-level proofs, and can take less time
    to construct and check. Our highest-level proof checker, Level 11, can
    be thought of as a simplified version of the ACL2 or NQTHM theorem
    provers. One contribution of this work is to show how such systems can
    be verified.

    To establish that the Level 11 proof checker can be trusted, we first
    use it, without trusting it, to prove the fidelity of every Level n to
    Level 1: whenever Level n accepts a proof of some $\phi$, there exists a
    Level 1 proof of $\phi$. We then mechanically translate the Level 11 proof
    for each Level n into a Level n  1 proofthat is, we create a Level 1
    proof of Level 2s fidelity, a Level 2 proof of Level 3s fidelity,
    and so on. This layering shows that each level can be trusted, and
    allows us to manage the sizes of these proofs.

    In this way, our system proves its own fidelity, and trusting Level 11
    only requires us to trust Level 1.",
  paper = "Davi09.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@article{Davi15,
  author = "Davis, Jared and Myreen, Magnus O.",
  title = {{The Reflective Milawa Theorem Prover is Sound}},
  journal = "J. Automated Reasoning",
  volume = "55",
```

```
  number = "2",
  pages = "117-183",
  year = "2015",
  link = "\url{https://www.cl.cam.ac.uk/~mom22/jitawa/}",
  abstract =
    "This paper presents, we believe, the most comprehensive evidence of a
    theorem provers soundness to date. Our subject is the Milawa theorem
    prover. We present evidence of its soundness down to the machine
    code. Milawa is a theorem prover styled after NQTHM and ACL2. It is
    based on an idealised version of ACL2s computational logic and
    provides the user with high-level tactics similar to ACL2s. In
    contrast to NQTHM and ACL2, Milawa has a small kernel that is somewhat
    like an LCF-style system. We explain how the Milawa theorem prover is
    constructed as a sequence of reflective extensions from its
    kernel. The kernel establishes the soundness of these extensions
    during Milawas boot-strapping process. Going deeper, we explain how
    we have shown that the Milawa kernel is sound using the HOL4 theorem
    prover. In HOL4, we have formalized its logic, proved the logic sound,
    and proved that the source code for the Milawa kernel (1,700 lines of
    Lisp) faithfully implements this logic. Going even further, we have
    combined these results with the x86 machine-code level verification of
    the Lisp runtime Jitawa. Our top-level theorem states that Milawa can
    never claim to prove anything that is false when it is run on this
    Lisp runtime.",
  paper = "Davi15.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@inproceedings{Deha16,
  author = "Dehaye, Paul-Olivier and Iancu, Mihnea and Kohlhase, Michael
            and Konovalov, Alexander and Lelievre, Samuel and
            Muller, Dennis and Pfeiffer, Markus and Rabe, Florian and
            Thiery, Nicolas M. and Wiesling, Tom",
  title = {{Interoperability in the OpenDreamKit project: The
           Math-in-the-Middle Approach}},
  booktitle = "Intelligent Computer Mathematics",
  pages = "117-131",
  year = "2016",
  isbn = "9783319425467",
  publisher = "Springer",
  abstract =
    "OpenDreamKit - 'Open Digital Research Environment Toolkit for the
    Advancement of Mathematics' - is an H2020 EU Research Infrastructure
    project that aims at supporting, over the period 2015-2019, the
    ecosystem of open-source mathematical software systems. OpenDreamKit
    will deliver a flexible toolkit enabling research groups to set up
    Virtual Research Environments, customised to meet the varied needs of
    research projects in pure mathematics and applications.
```

```
    An important step in the OpenDreamKit endeavor is to foster the
    interoperability between a variety of systems, ranging from
    computer algebra systems over mathematical databases to
    front-ends. This is the mission of the integration work
    package. We report on experiments and future plans with the
    Math-in-the-Middle approach. This architecture consists of a
    central mathematical ontology that documents the domain and fixes a
    joint vocabulary, or even a language, going beyond existing
    systems such as OpenMath, combined with specifications of the
    functionalities of the various systems. Interaction between
    systems can then be enriched by pivoting around this architecture.",
  paper = "Deha16.pdf",
  keywords = "printed, axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Ders87,
  author = "Dershowitz, Nachum",
  title = {{Termination of Rewriting}},
  journal = "J. Symbolic Computation",
  volume = "3",
  year = "1987",
  pages = "69-116",
  abstract =
    "This survey describes methods for proving that systems of rewrite
    rules are terminating programs.  We illustrate the use in termination
    proofs of various kinds of orderings on terms, including polynomial
    interpretations and path orderings. The effect of restrictions, such
    as linearity, on the form of rules is also considered.  In general,
    however, termination is an undeeidable property of rewrite systems.",
  paper = "Ders87.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Dick88,
  author = "Dick, A.J.J.",
  title = {{Automated Equational Reasoning and the Knuth-Bendix Algorithm:
            An Informal Introduction}},
  year = "1988",
  type = "technical report",
  institution = "Rutherford Appelton Laboratory",
  number = "RAL-88-043",
  paper = "Dick88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Dick91,
  author = "Dick, Jeremy",
  title = {{An Introduction to Knuth-Bendix Completion}},
  journal = "The Computer Journal",
  volume = "34",
  number = "1",
  year = "1991",
  abstract =
    "An informal introduction is given to the underlying concepts of
    term rewriting. Topics covered are Knuth-Bendix completion,
    completion modulo equations, unfailing completion and theorem
    proving by completion.",
  paper = "Dick91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Dijk71,
  author = "Dijkstra, E.W.",
  title = {{A Short Introduction to the Art of Programming}},
  comment = "EWD316",
  year = "1971",
  paper = "Dijk71.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Dijk72a,
  author = "Dijkstra, E.W.",
  title = {{Notes on Structured Programming}},
  booktitle = "Structured Programming",
  publisher = "Academic Press",
  year = "1972",
  pages = "1-82"
}
```

———-

— **axiom.bib** —

```
@article{Dijk75,
```

```
  author = "Dijkstra, Edsgar W.",
  title = {{Guarded Commands, Nondeterminancy and Formal Derivation
           of Programs}},
  journal = "Comm. of the ACM",
  volume = "18",
  number = "8",
  year = "1975",
  pages = "453-457",
  abstract =
    "So-called 'guarded commands' are introduced as a building block
    for alternative and repetitive constructs that allow
    nondeterministic program components for which at least the
    activity evoked, but possibly even the final state, is not
    necessarily uniquly determined by the initial state. For the
    formal derivation of programs expressed in terms of these
    constructs, a calculus will be shown.",
  paper = "Dijk75.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Dola16,
  author = "Dolan, Stephen",
  title = {{Algebraic Subtyping}},
  school = "University of Cambridge",
  year = "2016",
  link = "\url{https://www.cl.cam.ac.uk/~sd601/thesis.pdf}",
  abstract =
    "Type inference gives programmers the benefit of static,
    compile-time type checking without the cost of manually specifying
    types, and has long been a standard feature of functional programming
    languages. However, it has proven difficult to integrate type
    inference with subtyping, since the unification engine at the core of
    classical type inference accepts only equations, not subtyping
    constraints.

    This thesis presents a type system combining ML-style parametric
    polymorphism and subtyping, with type inference, principal types,
    and decidable type subsumption. Type inference is based on {\sl
    biunification} , an analogue of unification that works with
    subtyping constraints.

    Making this possible are several contributions, beginning with the
    notion of an 'extensible' type system, in which an open world of
    types is assumed, so that no typeable program becomes untypeable
    by the addition of new types to the language. While previous
    formulations of subtyping fail to be extensible, this thesis
    shows that adopting a more algebraic approach can remedy this.
    Using such an approach, this thesis develops the theory of
    biunification, shows how it is used to infer types, and shows how
```

```
    it can be efficiently implemented, exploiting deep connections
    between the algebra of regular languages and polymorphic subtyping.",
  paper = "Dola16.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Dola17,
  author = "Dolan, Stephen and Mycroft, Alan",
  title = {{Polymorphism, Subtyping, and Type Inference in MLsub}},
  journal = "ACM SIGPLAN Notices",
  volume = "52",
  number = "1",
  year = "2017",
  pages = "60-72",
  link = "\url{https://www.cl.cam.ac.uk/~sd601/papers/mlsub-preprint.pdf}",
  abstract =
    "We present a type system combining subtyping and ML-style parametric
    polymorphism. Unlike previous work, our system supports type inference
    and has compact principal types. We demonstrate this system in the
    minimal language MLsub, which types a strict superset of core ML
    programs.

    This is made possible by keeping a strict separation between the types
    used to describe inputs and those used to describe outputs, and
    extending the classical unification algorithm to handle subtyping
    constraints between these input and output types. Principal types are
    kept compact by type simplification, which exploits deep connections
    between subtyping and the algebra of regular languages. An
    implementation is available online.",
  paper = "Dola17.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Domi18,
  author = "Domipheus",
  title = {{Designing a CPU in VHDL}},
  link = "\url{http://labs.domipheus.com/blog/tpu-series-quick-links}",
  year = "2018",
  abstract = "A VHDL CPU"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Down16,
  author = "Downen, Paul and Maurer, Luke and Ariola, Zena M.",
  title = {{Sequent Calculus as a Compiler Intermediate Language}},
  booktitle = "ICFP'16",
  year = "2016",
  pages = "74-88",
  publisher = "ACM",
  isbn = "978-1-4503-4219-3",
  abstract =
    "The $\lambda$-calculus is popular as an intermediate language for
    practical compilers. But in the world of logic it has a
    lesser-known twin, born at the same time, called the {\sl sequent
    calculus}. Perhaps that would make for a good intermediate
    language, too? To explore this question we designed Sequent Core,
    a practically-oriented core calculus based on the sequent
    calculus, and used it to re-implement a substantial chunk of the
    Glasgow Haskell Compiler.",
  paper = "Down16.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Drey02,
  author = "Dreyer, Derek and Crary, Karl and Harper, Robert",
  title = {{A Type System for Higher-Order Modules (Expanded Version)}},
  type = "technical report",
  institution = "Carnegie Mellon University",
  number = "CMU-CS-02-122R",
  year = "2002",
  link = "\url{https://www.cs.cmu.edu/~crary/papers/2003/thoms/thoms-tr.pdf}",
  abstract =
    "We present a type theory for higher-order modules that accounts for
    many central issues in module system design, including translucency,
    applicativity , generativity , and modules as first-class values.
    Our type system harmonizes design elements from previous work,
    resulting in a simple, economical account of modular programming.  The
    main unifying principle is the treatment of abstraction mechanisms
    as computational effects.  Our language is the first to provide a
    complete and practical formalization of all of these critical issues
    in module system design.",
  paper = "Drey02.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Drey03,
```

```
  author = "Dreyer, Derek and Crary, Karl and Harper, Robert",
  title = {{A Type System for Higher-Order Modules}},
  journal = "ACM SIGPLAN Notices",
  volume = "38",
  number = "1",
  year = "2003",
  link = "\url{https://www.cs.cmu.edu/~crary/papers/2003/thoms/thoms.pdf}",
  abstract =
    "We present a type theory for higher-order modules that accounts for
    many central issues in module system design, including translucency,
    applicativity , generativity , and modules as first-class values.
    Our type system harmonizes design elements from previous work,
    resulting in a simple, economical account of modular programming.  The
    main unifying principle is the treatment of abstraction mechanisms
    as computational effects.  Our language is the first to provide a
    complete and practical formalization of all of these critical issues
    in module system design.",
  paper = "Drey03.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Drey07,
  author = "Dreyer, Derek and Harper, Robert and Chakravarty, Manuel M.T.
            and Keller, Gabriele",
  title = {{Modlular Type Classes}},
  booktitle = "Proc. POPL'07",
  pages = "63-70",
  year = "2007",
  abstract =
    "ML modules and Haskell type classes have proven to be highly
    effective tools for program structuring.  Modules emphasize explicit
    configuration of program components and the use of data abstraction.
    Type classes emphasize implicit program construction and ad hoc
    polymorphism.  In this paper , we show how the implicitly-typed
    style of type class programming may be supported within the framework
    of an explicitly-typed module language by viewing type classes as a
    particular mode of use of modules.  This view offers a harmonious
    integration of modules and type classes, where type class features,
    such as class hierarchies and associated types, arise naturally as
    uses of existing module-language constructs, such as module
    hierarchies and type components.  In addition, programmers have
    explicit control over which type class instances are available for
    use by type inference in a given scope.  We formalize our approach as
    a Harper-Stone-style elaboration relation, and provide a sound type
    inference algorithm as a guide to implementation.",
  paper = "Drey07.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Dura14,
  author = "Duran, Antonio J. and Perez, Mario and Varona, Juan L.",
  title = {{The Misfortunes of a Trio of Mathematicians Using Computer
           Algebra Systems. Can We Trust in Them?}},
  journal = "Notices of the AMS",
  volume = "61",
  number = "10",
  year = "2014",
  link = "\url{www.ams.org/notices/201410/rnoti-p1249.pdf}",
  pages = "1249-1252",
  paper = "Dura14.pdf",
  keywords = "printed, DONE"
}
```

———-

1.17.5 E

— **axiom.bib** —

```
@article{Emde76,
  author = "van Emden, M.H. and Kowalski, R.A.",
  title = {{The Semantics of Predicate Logic as a Programming Language}},
  journal = "J. Association for Computing Machinery",
  volume = "23",
  number = "4",
  year = "1976",
  pages = "733-742",
  abstract =
    "Sentences in first-order predicate logic can be usefully interpreted
    as programs In this paper the operational and fixpomt semantics of
    predicate logic programs are defined, and the connections with the
    proof theory and model theory of logic are investigated It is
    concluded that operational semantics is a part of proof theory and
    that fixpolnt semantics is a special case of model-theoret:c
    semantics.",
  paper = "Emde76.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Engl14c,
  author = "England, M. and Cheb-Terrab, E. and Bradford, R. and
            Davenport, J.H. and Wilson, D.",
```

```
  title = {{Branch Cuts in MAPLE 17}},
  journal = "ACM Comm. in Computer Algebra",
  volume = "48",
  number = "1",
  year = "2014",
  abstract =
    "Accurate and comprehensible knowledge about the position of branch
    cuts is essential for correctly working with multi-valued functions,
    such as the square root and logarithm. We discuss the new tools in
    Maple 17 for calculating and visualising the branch cuts of such
    functions, and others built up from them. The cuts are described in an
    intuitive and accurate form, offering substantial improvement on the
    descriptions previously available.",
  paper = "Engl14c.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Espa17,
  author = "Esparza, Javier",
  title = {{Automata Theory: An Algorithmic Approach}},
  publisher = "Esparza, Javier",
  year = "2017",
  link = "\url{https://www7.in.tum.de/~esparza/autoskript.pdf}",
  paper = "Espa17.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Esse00,
  author = "Essex, Christopher and Davison, Matt and Schulzky, Christian",
  title = {{Numerical Monsters}},
  journal = "ACM SIGSAM",
  volume = "34",
  number = "4",
  year = "2000",
  abstract =
    "When the results of certain computer calculations are shown to be not
    simply incorrect but dramatically incorrect, we have a powerful reason
    to be cautious about all computer-based calculations. In this paper we
    present a 'Rogue's Gallery' of simple calculations whose correct
    solutions are obvious to humans but whose numerical solutions are
    incorrect in pathological ways. We call these calculations, which can
    be guaranteed to wreak numerical mayhem across both software
    packages and hardware platforms, 'Numerical Monsters'. Our monsters
    can be used to provide deep insights into how computer calculations
    fail, and we use t h e m to engender appreciation for the subject of
```

```
    numerical analysis in our students. Although these monsters are based
    on well-understood numerical pathologies, even experienced numerical
    analysts will find surprises in their behaviour and ,can use the
    lessons they bring to become even better masters of their tools.",
  paper = "Esse00.pdf"
}
```

———-

1.17.6 F

— **axiom.bib** —

```
@article{Farm95b,
  author = "Farmer, William M.",
  title = {{Reasoning about Partial Functions with the Aid of a
            Computer}},
  journal = "Erkenntnis",
  volume = "43",
  number = "3",
  pages = "279-294",
  year = "1995",
  abstract =
    "Partial functions are ubiquitous in both mathematics and computer
    science. Therefore, it is imperative that the underlying logical
    formalism for a general-purpose mechanized mathematics system
    provide strong support for reasoning about partial
    functions. Unfortunately, the common logical formalisms --
    first-order logic, type theory, and set theory -- are usually only
    adequate for reasoning about partial functions {\sl in
    theory}. However, the approach to partial functions traditionally
    employed by mathematicians is quite adequate {\sl in
    practice}. This paper shows how the traditional approach to
    partial functions can be formalized in a range of formalisms that
    includes first-order logic, simple type theory, and Von-Neuman,
    Bernays, Godel set theory. It argues that these new formalisms
    allow one to directly reason about partial functions; are based on
    natural, well-understood, familiar principles; and can be
    effectively implemented in mechanized mathematics systems.",
  paper = "Farm95b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Farm00,
  author = "Farmer, William M.",
  title = {{A Proposal for the Development of an Interactive
            Mathematics Laboratory for Mathematics Eductions}},
```

```
  booktitle = "Workshop on Deductions Systems for Mathematics Eduation",
  pages = "20-25",
  year = "2000",
  paper = "Farm00.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Farm00a,
  author = "Farmer, William M. and Mohrenschildt, Martin v.",
  title = {{Transformers for Symbolic Computation and Formal Deduction}},
  booktitle = "CADE-17",
  pages = "36-45",
  year = "2000",
  abstract =
    "A transformer is a function that maps expressions to expressions.
    Many transformational operators - such as expression evaluators and
    simplifiers, rewrite rules, rules of inference, and decision
    procedures - can be represented by transformers. Computations and
    deductions can be formed by applying sound transformers in
    sequence. This paper introduces machinery for defining sound
    transformers in the context of an axiomatic theory in a formal
    logic. The paper is intended to be a first step in a development of an
    integrated framework for symbolic computation and formal deduction.",
  paper = "Farm00a.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Farm04,
  author = "Farmer, William M.",
  title = {{MKM A New Interdisciplinary Field of Research}},
  journal = "SIGSAM",
  volume = "38",
  pages = "47-52",
  year = "2004",
  abstract =
    "Mathematical Knowledge Management (MKM) is a new interdisciplinary
    field of research in the intersection of mathematics, computer
    science, library science, and scientific publishing. Its objective is
    to develop new and better ways of managing mathematical knowledge
    using sophisticated software tools.  Its grand challenge is to create
    a universal digital mathematics library (UDML), accessible via the
    World-Wide Web, that contains essentially all mathematical knowledge
    (intended for the public). The challenges facing MKM are daunting,
    but a UDML, even just partially constructed, would transform how
```

```
    mathematics is learned and practiced.",
  paper = "Farm04.pdf",
  keywords = "printed, axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Farm07,
  author = "Farmer, William M.",
  title = {{Biform Theories in Chiron}},
  journal = "LNCS",
  volume = "4573",
  pages = "66-79",
  year = "2007",
  abstract =
    "An axiomatic theory represents mathematical knowledge declaratively
    as a set of axioms. An algorithmic theory represents mathematical
    knowledge procedurally as a set of algorithms. A biform theory is
    simultaneously an axiomatic theory and an algorithmic theory. It
    represents mathematical knowledge both declaratively and procedurally.
    Since the algorithms of algorithmic theories manipulate th e syntax of
    expressions, biform theories - as well as algorithmic theories - are
    difficult to formalize in a traditional logic without the means to
    reason about syntax. Chiron is a derivative of
    von-Neumann-Bernays-Godel ( NBG ) set theory that is intended to be a
    practical, general-purpose logic for mechanizing mathematics. It
    includes elements of type theory, a scheme for handling undefinedness,
    and a facility for reasoning about the syntax of expressions. It is an
    exceptionally well-suited logic for formalizing biform theories. This
    paper defines the notion of a biform theory, gives an overview of
    Chiron, and illustrates how biform theories can be formalized in Chiron.",
  paper = "Farm07.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Farm07a,
  author = "Farmer, William M.",
  title = {{Chiron: A Multi-Paradigm Logic}},
  journal = "Studies in Logic, Grammar and Rhetoric",
  volume = "10",
  number = "23",
  year = "2007",
  abstract =
    "Chiron is a derivative of von-Neumann-Bernays-Godel ( NBG ) set
    theory that is intended to be a practical, general-purpose logic for
    mechanizing mathematics. It supports several reasoning paradigms by
```

```
    integrating NBG set theory with elements of type theory, a scheme for
    handling undefinedness, and a facility for reasoning about the syntax
    of expressions.  This paper gives a quick, informal presentation of
    the syntax and semantics of Chiron and then discusses some of the
    benefits Chiron provides as a multi-paradigm logic.",
  paper = "Farm07a.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Farm13,
  author = "Farmer, William M.",
  title = {{Chiron: A Set Theory with Types, Undefinedness, Quotation,
            and Evaluation}},
  type = "technical report",
  institution = "McMaster University",
  number = "SQRL Report No. 38",
  year = "2013",
  link = "\url{https://arxiv.org/pdf/1305.6206.pdf}",
  abstract =
    "Chiron is a derivative of von-Neumann-Bernays-Godel (NBG) set
    theory that is intended to be a practical, general-purpose logic for
    mechanizing mathematics. Unlike traditional set theories such as
    Zermelo-Fraenkel (ZF) and NBG, Chiron is equipped with a type system,
    lambda notation, and definite and indefinite description. The type
    system includes a universal type, dependent types, dependent function
    types, subtypes, and possibly empty types. Unlike traditional logics
    such as first-order logic and simple type theory, Chiron admits
    undefined terms that result, for example, from a function applied to
    an argument outside its domain or from an improper definite or
    indefinite description. The most noteworthy part of Chiron is its
    facility for reasoning about the syntax of expressions. Quotation is
    used to refer to a set called a construction that represents the
    syntactic structure of an expression, and evaluation is used to refer
    to the value of the expression that a construction represents. Using
    quotation and evaluation, syntactic side conditions, schemas,
    syntactic transformations used in deduction and computation rules, and
    other such things can be directly expressed in Chiron. This paper
    presents the syntax and semantics of Chiron, some definitions and
    simple examples illustrating its use, a proof system for Chiron, and a
    notion of an interpretation of one theory of Chiron in another.",
  paper = "Farm13.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Farm14,
```

```
  author = "Farmer, William M. and Larjani, Pouya",
  title = {{Frameworks for Reasoning about Syntax that Utilize
           Quotation and Evaluation}},
  links = "\url{http://imps.mcmaster.ca/doc/syntax.pdf}",
  year = "2014",
  abstract =
    "It is often useful, if not necessary, to reason about the syntactic
    structure of an expression in an interpreted language (i.e., a
    language with a semantics).  This paper introduces a mathematical
    structure called a syntax framework that is intended to be an abstract
    model of a system for reasoning about the syntax of an interpreted
    language.  Like many concrete systems for reasoning about syntax, a
    syntax framework contains a mapping of expressions in the
    interpreted language to syntactic values that represent the syntactic
    structures of the expressions; a language for reasoning about the
    syntactic values; a mechanism called quotation to refer to the
    syntactic value of an expression; and a mechanism called evaluation to
    refer to the value of the expression represented by a syntactic value.
    A syntax framework provides a basis for integrating reasoning about
    the syntax of the expressions with reasoning about what the
    expressions mean.  The notion of a syntax framework is used to discuss
    how quotation and evaluation can be built into a language and to
    define what quasiquotation is.  Several examples of syntax frameworks
    are presented.",
  paper = "Farm14.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@article{Fate14a,
  author = "Fateman, Richard",
  title = {{Algorithm Differentiation in Lisp: ADIL}},
  journal = "ACM Comm. in Computer Algebra",
  volume = "48",
  number = "3",
  year = "2014",
  abstract =
    "Algorithm differentiation (AD) is a technique used to transform a
    program $F$ computing a numerical function of one argument $F(x)$ into
    another program $G(p)$ that returns a pair,
    $\langle F (p), F^\prime(p)\rangle$ where by $F^\prime(p)$
    we mean the derivative of $F$ with respect to its argument $x$,
    evaluated at $x = p$. That is, we have a program AD that takes as input
    a program, and returns another: $G := AD(F)$. Over the years AD
    programs have been developed to allow $F$ to be expressed in some
    specialized variant of a popular programming language L (FORTRAN, C,
    Matlab, Python) and where $G$ is delivered in that language $L$ or some
    other. Alternatively, executing $F(p)$ is some environment will deliver
    $\langle F^\prime(p), F(p)\rangle$ directly. AD tools have also
    been incorporated in computer algebra systems (CAS) such as
```

```
    Maple. A CAS is hardly necessary for the task of writing the AD
    program, since the main requirement is a set of tools for
    manipulation of an internal (typically tree) form of a program.
    In Lisp, a normal program is already in this form and so the AD
    program in Lisp (ADIL), the target $F$ and the product $G$ can
    all be expressed compactly in Lisp. In spite of the brevity and
    extensibility of the ADIL program, we can provide
    features which are unsupported in other AD programs. In particular,
    recursive functions are easily accommodated. Our perspective here is
    to point out that for scientists who write programs in Lisp or any
    language that can be converted to Lisp, AD is easily at hand.",
  paper = "Fate14a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fefe95,
  author = "Feferman, Solomon",
  title = {{Definedness}},
  year = "1995",
  link = "\url{https://math.stanford.edu/~feferman/papers/definedness.pdf}",
  abstract =
    "Questions of definedness are ubiquitous in mathematics.  Informally,
    these involve reasoning about expressions which may or may not have a
    value.  This paper surveys work on logics in which such reasoning can
    be carried out directly, especially in computational contexts.  It
    begins with a general logic of 'partial terms', continues with partial
    combinatory and lambda calculi, and concludes with an expressively
    rich theory of partial functions and polymorphic types, where
    termination of functional programs can be established in a natural way.",
  paper = "Fefe95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Felt90,
  author = "Felty, Amy and Miller, Dale",
  title = {{Encoding a Dependent-Type Lambda Calculus in a Logic
            Programming Language}},
  type = "technical report",
  number = "MS-CIS-90-18",
  institution = "University of Pennsylvania",
  year = "1990",
  abstract =
    "Various forms of typed $\lambda$-calculi have been proposed as
    specification languages for representing wide varieties of object
    logics. The logical framework, LF, is an example of such a
```

```
    dependent-type $\lambda$-calculus.  A small subset of intuitionistic
    logic with quantification over simply typed $\lambda$-calculus has
    also been proposed as a framework for specifying general logics. The
    logic of {\sl hereditary Harrop formulas} with quantification at all
    non-predicate types, denoted here as $hh^\omega$ , is such a
    meta-logic that has been implemented in both the Isabelle theorem
    prover and the $\lambda$Prolog logic programming language. Both
    frameworks provide for specifications of logics in which details
    involved with free and bound variable occurrences, substitutions,
    eigenvariables, and the scope of assumptions within object logics are
    handled correctly and elegantly at the 'meta' level. In this paper, we
    show how LF ca n be encoded into $hh^\omega$ in a direct and natural
    way by mapping the typing judgments in LF in to propositions in the
    logic of $hh^\omega$. This translation establishes a very strong
    connection between these two languages: the order of quantification in
    an LF signature is exactly the order of a set of $hh^\omega$ clauses,
    and the proofs in one system correspond directly to proofs in the
    other system. Relating these two languages makes it possible to
    provide implementations of proof checkers and theorem provers for
    logics specified in LF by using standard logic programming techniques
    which can be used to implement $hh^\omega$.",
  paper = "Felt90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Fill03,
  author = "Filliatre, Jean-Christophe",
  title = {{Verification of Non-Functional Programs using Interpretations
           in Type Theory}},
  journal = "J. Functional Programming",
  volume = "13",
  number = "4",
  pages = "709-745",
  year = "2003",
  abstract =
    "We study the problem of certifying programs combining imperative
    and functional features within the general framework of type
    theory.

    Type theory is a powerful specification language, which is
    naturally suited for the proof of purely functional programs. To
    deal with imperative programs, we propose a logical interpretation
    of an annotated program as a partial proof of its
    specification. The construction of the corresponding partial proof
    term is based on a static analysis of the effects of the program
    which excludes aliases. The missing subterms in the partial proof
    term are seen as proof obligations, whose actual proofs are left
    to the user. We show that the validity of those proof obligations
    implies the total correctness of the program.
```

```
    This work has been implemented in the Coq proof assistant. It
    appears as a tactic taking an annotated program as argument and
    generating a set of proof obligations. Several nontrivial
    algorithms have been certified using this tactic.",
  paper = "Fill03.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Fill13,
  author = "Filliatre, Jean-Christophe and Paskevich, Andrei",
  title = {{Why3 -- Where Programs Meet Provers}},
  journal = "LNCS",
  volume = "7792",
  year = "2013",
  link = "\url{https://hal.inria.fr/hal-00789533/document}",
  abstract =
    "We present Why3, a tool for deductive program verification, and
    WhyML, its programming and specification language. WhyML is a
    first-order language with polymorphic types, pattern matching, and
    inductive predicates. Programs can make use of record types with
    mutable fields, type invariants, and ghost code. Verification
    conditions are discharged by Why3 with the help of various existing
    automated and interactive theorem provers. To keep verification
    conditions tractable and comprehensible, WhyML imposes a static
    control of aliases that obviates the use of a memory model. A user
    can write WhyML programs directly and get correct-by-construction
    OCaml programs via an automated extraction mechanism. WhyML is also
    used as an intermediate language for the verification of C, Java, or
    Ada programs. We demonstrate the benefits of Why3 and WhyML on
    non-trivial examples of program verification.",
  paper = "Fill13.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Finn96,
  author = "Finney, Kate",
  title = {{Mathematical Notation in Formal Specifications: Too
            Difficult for the Masses?}},
  journal = "IEEE Trans. on Software Engineering",
  volume = "22",
  number = "2",
  year = "1996",
  pages = "158-159",
```

```
  abstract =
    "The phrase 'not much mathematics required' can imply a
    variety of skill levels. When this phrase is applied to computer
    scientists, software engineers, and clients in the area of formal
    specification, the word 'much' can be widely misinterpreted with
    disastrous consequences. A small experiment in reading
    specifications revealed that students already trained in discrete
    mathematics and the specification notation performed very poorly;
    much worse than could reasonably be expected if formal methods
    proponents are to be believed.",
  paper = "Finn96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Flan03,
  author = "Flanagan, Cormac and Sabry, Amr and Duba, Bruce F. and
            Felleisen, Matthias",
  title = {{The Essence of Compiling with Continuations}},
  link =
  "\url{https://www.cs.rice.edu/~javaplt/411/17-spring/Readings/essence-retro.pdf}",
  paper = "Flan03.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Floy64,
  author = "Floyd, Robert W.",
  title = {{Algorithm 245: Treesort}},
  journal = "CACM",
  volume = "7",
  number = "12",
  year = "1964",
  pages = "701",
  paper = "Floy64.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Floy67,
  author = "Floyd, Robert W.",
  title = {{Assigning Meanings to Programs}},
  booktitle = "Proc. Symp. in Applied Mathematics",
```

```
  year = "1967",
  pages = "19-32",
  publisher = "American Mathematical Society",
  paper = "Floy67.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Fong18,
  author = "Fong, Brendan and Spivak, David I.",
  title = {{Seven Sketches in Compositionality: An Invitation to
            Applied Category Theory}},
  link = "\url{http://math.mit.edu/~dspivak/teaching/sp18/7Sketches.pdf}",
  year = "2018",
  paper = "Fong18.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Free91,
  author = "Freeman, Tim and Pfenning, Frank",
  title = {{Refinement Types for ML}},
  booktitle = "ACM SIGPLAN PLDI'91",
  year = "1991",
  link = "\url{http://www.cs.cmu.edu/~fp/papers/pldi91.pdf}",
  abstract =
    "We describe a refinement of ML's type system allowing the
    specification of recursively defined subtypes of user-defined
    datatypes. The resulting system of {\sl refinement types}
    preserves desirable properties of ML such as decidability of type
    inference, while at the same time allowing more errors to be
    detected at compile-time. The type system combines abstract
    interpretation with ideas from the intersection type discipline,
    but remains closely tied to ML in that refinement types are given
    only to programs which are already well-typed in ML.",
  paper = "Free91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Freu08,
  author = "Freundt, Sebastian and Horn, Peter and Konovalov, Alexander
            and Linton, Steve and Roozemond, Dan",
```

```
  title = {{Symbolic Computation Software Composability}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
  abstract =
    "We present three examples of the composition of Computer Algebra
    Systems to illustrate the progress on a composability infrastructure
    as part of the SCIEnce (Symbolic Computation Infrastructure for
    Europe) project 1 . One of the major results of the project so far is
    an OpenMath based protocol called SCSCP (Symbolic Computation Software
    Composability Protocol). SCSCP enables the various software packages
    for example to exchange mathematical objects, request calcula- tions,
    and store and retrieve remote objects, either locally or accross the
    internet. The three examples show the current state of the GAP, KANT,
    and MuPAD software packages, and give a demonstration of exposing
    Macaulay using a newly developed framework.",
  paper = "Freu08.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Foxx03,
  author = "Fox, Anthony",
  title = {{Formal Specification and Verification of ARM6}},
  journal = "LNCS",
  volume = "2758",
  pages = "25-40",
  year = "2003",
  abstract =
    "This paper gives an overview of progress made on the formal
    specification and verification of the ARM6 micro-architecture using
    the HOL proof system. The ARM6 is a commercial processor design preva-
    lent in mobile and embedded systems  it features a 3-stage pipeline
    with a multi-cycle execute stage, six operating modes and a rich
    32-bit RISC instruction set. This paper describes some of the
    difficulties encountered when working with a full blown instruction
    set architecture that has not been designed with verification in mind.",
  paper = "Foxx03.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Fran92,
  author = "Francez, Nissim",
  title = {{Program Verification}},
  year = "1992",
  publisher = "Addison-Wesley",
```

```
  isbn = "0-201-41608-5",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@techreport{Frie76,
  author = "Friedman, Daniel P. and Wise, David S.",
  title = {{CONS should not Evaluate its Arguments}},
  institution = "Indiana University",
  number = "TR44",
  year = "1976",
  abstract =
    "The constructor function which allocates and fills records in
    recursive, side-effect-free procedural languages is redefined to be a
    non-strict (Vuillemin 1974) elementary operation. Instead of
    evaluating its arguments, it builds suspensions of them which are not
    coerced until the suspensions is accessed by strict elementary
    function. The resulting evalutation procedures are strictly more
    powerful than existing schemes for languages such as LISP. The main
    results are that Landin's streams are subsumed into McCarthy's LISP
    merely by the redefinition of elementar functions, that invocations of
    LISP's evaluator can be minimized by redefining the elemntary
    functions without redefining the interpreter, and as a strong
    conjecture, that redefining the elementary functions yields the least
    fixed-point semantics for McCarthy's evalution scheme. This new
    insight into the role of construction functions will do much to ease
    the interface between recursive programmers and iterative programmers,
    as well as the interface between programmers and data structure
    designers.",
  paper = "Frie16.pdf",
  keywords = "printed"
}
```

———-

1.17.7 G

— **axiom.bib** —

```
@misc{Giar95,
  author = "Girard, Jean-Yves",
  title = {{Linear Logic: Its Syntax and Semantics}},
  year = "1995",
  paper = "Gira95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Glas78,
  author = "Glasner, Ingrid and Loeckx, Jacquest",
  title = {{A Calculus for Proving Properties of While-Programs}},
  journal = "LNCS",
  volume = "75",
  pages = "252-281",
  year = "1978",
  paper = "Glas78.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Glym15,
  author = "Glymour, Clark and Serafin, Luke",
  title = {{Mathematical Metaphysics}},
  link = "\url{http://shelf1.library.cmu.edu/HSS/2015/a1626190.pdf}",
  year = "2015",
  paper = "Glym15.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Gont09a,
  author = "Gonthier, Georges",
  title = {{Software Engineering for Mathematics}},
  journal = "LNCS",
  volume = "5625",
  year = "2009",
  pages = "27-27",
  abstract =
    "Despite its mathematical origins, progress in computer assisted
    reasoning has mostly been driven by applications in computer science,
    like hardware or protocol security verification. Paradoxically, it has
    yet to gain widespread acceptance in its original domain of
    application, mathematics; this is commonly attributed to a 'lack of
    libraries': attempts to formalize advanced mathematics get bogged down
    into the formalization of an unwieldly large set of basic resuts.

    This problem is actually a symptom of a deeper issue: the main
    function of computer proof systems, checking proofs down to their
    finest details, is at odds with mathematical practice, which ignores
    or defers details in order to apply and combine abstractions in
    creative and elegant ways. Mathematical texts commonly leave logically
    important parts of proofs as 'exercises to the reader', and are rife
```

```
    with 'abuses of notation that make mathematics tractable' (according
    to Bourbaki). This (essential) flexibility cannot be readily
    accomodated by the narrow concept of 'proof library' used by most
    proof assistants and based on 19th century first-order logic: a
    collection of constants, definitions, and lemmas.

    This mismatch is familiar to software engineers, who have been
    struggling for the past 50 years to reconcile the flexibility needed
    to produce sensible user requirements with the precision needed to
    implement them correctly with computer code. Over the last 20 years
    object and components have replaced traditional data and procedure
    libraries, partly bridging this gap and making it possible to build
    significantly larger computer systems.

    These techniques can be implemented in compuer proof systems by
    exploiting advances in mathematical logic. Higher-order logics allow
    the direct manipulation of functions; this can be used to assign
    behaviour, such as simplification rules, to symbols, similarly to
    objects. Advanced type systems can assign a secondary, contextual
    meaning to expressions, using mechanisms such as type classes,
    similarly to the metadata in software components. The two can be combined
    to perform reflection, where an entire statement gets quoted as
    metadata and then proved algorithmically by some decision procedure.

    We propose to use a more modest, small-scale form of reflection, to
    implement mathematical components. We use the type-derived metadata to
    indicate how symbols, definitions and lemmas should be used in other
    theories, and functions to implement this usage  roughly, formalizing
    some of the exercize section of a textbook. We have applied
    successfully this more engineered approch to computer proofs in our
    past work on the Four Color Theorem, the Cayley-Hamilton Theorem, and
    our ongoing long-term effort on the Odd Order Theorem, which is the
    starting point of the proof of the Classification of Finite Simple
    Groups (the famous 'monster theorem' whose proof spans 10,000 pages in
    400 articles).",
  paper = "Gont09a.pdf",
  keywords = "printed, DONE"
}
```

———-

— axiom.bib —

```
@article{Good80,
  author = "Goodman, Nicolas D.",
  title = {{Reflections on Bishop's Philosophy of Mathematics}},
  journal = "Lecture Notes in Mathematics",
  volume = "873",
  pages = "135-145",
  year = "1980",
  paper = "Good80.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Gord79,
  author = "Gordon, Michael J. and Milner, Arthur j. and
            Wadsworth, Christopher P.",
  title = {{Edinburgh LCF, A Mechanised Logic of Computation:
            Introduction}},
  publisher = "Springer-Verlag",
  journal = "LNCS",
  volume = "78",
  year = "1979",
  paper = "Gord79.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gord89,
  author = "Gordon, Michael J.C.",
  title = {{Mechanizing Programming Logics in Higher Order Logic}},
  booktitle = "Current Trends in Hardware Verification and Automated
               Theorem Proving",
  publisher = "Springer",
  year = "1989",
  abstract =
    "Formal reasoning about computer programs can be based directly on the
    seman tics of the programming language, or done in a special purpose
    logic like Hoare logic.  The advantage of the first approach is that
    it guarantees that the formal reasoning applies to the language being
    used (it is well known, for example, that Hoare's assignment axiom
    fails to hold for most programming languages).  The advantage of the
    second approach is that the proofs can be more direct and natural.

    In this paper, an attempt to get the advantages of both approaches
    is described.  The rules of Hoare logic are mechanically derived
    from the semantics of a simple imperative programming language
    (using the HOL system).  These rules form the basis for a simple
    program verifier in which verification conditions are generated by LCF
    -style tactics whose validations use the derived Hoare rules.
    Because Hoare logic is derived, rather than postulated, it is straight
    tforw ard to mix seman tic and axiomatic rea- soning.  It is also
    forward to combine the constructs of Hoare logic with other
    application-specific notations.  This is briefly illustrated for
    various logical constructs, including termination statements, VDM-style
    'relational' correctness specifications, weakest precondition
    statements and dynamic logic formulae.

    The theory underlying the work presented here is well known. Our
```

```
    contribution is to propose a way of mechanizing this theory in a
    way that makes certain practical details work out smoothly .",
  paper = "Gord89.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Gord06,
  author = "Gordon, Mike and Iyoda, Juliano and Owens, Scott and
            Slind, Konrad",
  title = {{Automatic Formal Synthesis of Hardware from Higher Order Logic}},
  journal = "Electronic Notes in Theoretical Computer Science",
  volume = "145",
  pages = "27-43",
  year = "2006",
  abstract =
    "A compiler that automatically translates recursive function
    definitions in higher order logic to clocked synchronous hardware is
    described. Compilation is by mechanised proof in the HOL4 system, and
    generates a correctness theorem for each function that is
    compiled. Logic formulas representing circuits are synthesised in a
    form suitable for direct translation to Verilog HDL for simulation and
    input to standard design automation tools. The compilation scripts are
    open and can be safely modified: synthesised circuits are
    correct-by-construction. The synthesisable subset of higher order
    logic can be extended using additional proof-based tools that
    transform definitions into the subset.",
  paper = "Gord06.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Grah93,
  author = "Graham, Paul",
  title = {{On Lisp}},
  publisher = "Prentice Hall",
  year = "1993",
  link = "\url{http://ep.yimg.com/ty/cdn/paulgraham/onlisp.pdf}",
  comment = "code in papers/onlisp.lisp",
  isbn = "0130305529",
  abstract =
    "On Lisp is a comprehensive study of advanced Lisp techniques, with
    bottom-up programming as the unifying theme. It gives the first
    complete description of macros and macro applications. The book also
    covers important subjects related to bottom-up programming, including
    functional programming, rapid prototyping, interactive development,
    and embedded languages. The final chapter takes a deeper look at
```

```
    object-oriented programming than previous Lisp books, showing the
    step-by-step construction of a working model of the Common Lisp Object
    System (CLOS).",
  paper = "Grah93.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Grai07,
  author = "Graillat, Stef and Menissier-Morain, Valerie",
  title = {{Error-free Transformations in Real and Complex
            Floating-point Arithmetic}},
  booktitle = "Int. Symp. on Nonlinear Theory and Applications",
  year = "2007",
  abstract =
    "Error-free transformation is a concept that makes it possible to
    compute accurate results within a floating point arithmetic.  Up to
    now, it has only be studied for real floating point arithmetic.  In
    this short note, we recall the known error-free transformations for
    real arithmetic and we propose some new error-free transformations for
    complex floating point arithmetic. This will make it possible to
    design some new accurate algorithms for summation, dot product and
    polynomial evaluation with complex entries.",
  paper = "Grai07.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Grev08,
  author = "Greve, David A. and Kaufmann, Matt and Manolios, Panagiotis
            and Moore, J Strother and Ray, Sandip and Ruiz-Reina, Jose Luis
            and Sumners, Rob and Vroon, Daron and Wilding, Matthew",
  title = {{Efficient Execution in an Automated Reasoning Environment}},
  journal = "Journal of Functional Programming",
  volume = "18",
  number = "1",
  pages = "15-46",
  year = "2008",
  abstract =
    "We describe a method that permits the user of a mechanized
    mathematical logic to write elegant logical definitions while allowing
    sound and efficient execution. In particular, the features supporting
    this method allow the user to install, in a logically sound way,
    alternative executable counterparts for logically defined
    functions. These alternatives are often much more efficient than the
    logically equivalent terms they replace.  These features have been
    implemented in the ACL2 theorem prover, and we discuss several
    applications of the features in ACL2.",
```

```
  paper = "Grev08.pdf",
  keywords = "printed"

}
```

———-

— **axiom.bib** —

```
@misc{Gros15,
  author = "Gross, Jason and Chlipala, Adam",
  title = {{Parsing Parses}},
  link = "\url{https://people.csail.mit.edu/jgross/personal-website/papers/2015-parsing-parse-trees.pdf}",
  year = "2015",
  abstract =
    "We present a functional parser for arbitrary context-free grammars,
    together with soundness and completeness proofs, all inside Coq. By
    exposing the parser in the right way with parametric polymorphism and
    dependent types, we are able to use the parser to prove its own
    soundness, and, with a little help from relational parametricity,
    prove its own completeness, too. Of particular interest is one strange
    instantiation of the type and value parameters: by parsing parse trees
    instead of strings, we convince the parser to generate its own
    completeness proof. We conclude with highlights of our experiences
    iterating through several versions of the Coq development, and some
    general lessons about dependently typed programming.",
  paper = "Gros15.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Gual14,
  author = "Guallart, Nino",
  title = {{An Overview of Type Theories}},
  link = "\url{https://arxiv.org/pdf/1411.1029.pdf}",
  year = "2014",
  abstract =
    "Pure type systems arise as a generalisation of simply typed lambda
    calculus. The contemporary development of mathematics has renewed
    the interest in type theories, as they are not just the object of
    mere historical research, but have an active role in the development
    of computational science and core mathematics.  It is worth exploring
    some of them in depth, particularly predicative Martin-Lfs
    intuitionistic type theory and impredicative Coquands calculus of
    constructions. The logical and philosophical differences and
    similarities between them will be studied, showing the relationship
    between these type theories and other fields of logic.",
  paper = "Gual14.pdf",
  keywords = "printed, DONE"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Guinxx,
  author = "Guindon, Dick",
  title = {{Writing is nature's way of letting you know how sloppy
            your thinking is}},
  link = "\url{http://www.guindoncartoons.com/userfiles/1231656389.jpg?rand=876778554}",
  comment = "Guindon's Michigan",
  year = "unknown"
}
```

———-

— **axiom.bib** —

```
@techreport{Gunt89,
  author = "Gunter, Elsa L.",
  title = {{Doing Algebra in Simple Type Theory}},
  type = "technical report",
  institution = "University of Pennsylvania",
  year = "1989",
  number = "MS-CIS-89-38",
  abstract =
    "To fully utilize the power of higher-order logic in interactive
    theorem proving, it is desirable to be able to develop abstract areas
    of Mathematics such as algebra and topology in an automated
    setting. Theorems provers capable of higher order reasoning have
    generally had some form of type theory as theory object
    language. But mathematicians have tended to use the language of set
    theory to give definitions and prove theorems in algebra and
    topology. In this paper,we give an incremental description of how to
    express various basic algebraic concepts in terms of simple type
    theory. We present a method for representing algebras, subalgebras,
    quotient algebras, homorphisms and isomorphisms simple type theory,
    using group theory as an example in each case. Following this we
    discuss how to automatically apply such an abstract theory to concrete
    examples. Finally, we conclude with some observations about a
    potential inconvenience associated with this method of representation,
    and discuss a difficulty inherent in any attempt to remove this
    inconvenience.",
  paper = "Gunt89.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gunt89a,
  author = "Gunter, Elsa L.",
  title = {{Extensions to Logic Programming Motiviated by the
            Construction of a Generic Theorem Prover}},
  booktitle = "Int. Workshop on Extensions of Logic Programming",
  publisher = "Springer",
  year = "1989",
  pages = "223-244",
  abstract =
    "In this article, we discuss several possible extensions to
    traditional logic programming languages. The specific extensions
    proposed here fall into two categories: logical extensions and the
    addition of constructs to allow for increased control.  There is a
    unifying theme to the proposed logical extensions, and that is the
    scoped introduction of extensions to a programming context. More
    specifically, these extensions are the ability to introduce variables
    whose scope is limited to the term in which they occur (i.e. A-bound
    variables within A-terms), the ability to intro~iuce into a goal a
    fresh constant whose scope is limited to the derivation of that goal,
    and the ability to introduce into a goal a program clause whose scope
    is limited to the derivation of that goal. The purpose of the
    additions for increased control is to facilitate the raising and
    handling of failures.

    To motivate these various extensions, we have repeatedly appealed to
    examples related to the construction of a generic theorem prover. It
    is our thesis that this problem domain is specific enough to lend
    focus when one is considering various language constructs, and yet
    complex enough to encompass many of the general difficulties found in
    other areas of symbolic computation.",
  paper = "Gunt89a.pdf"
}
```

———-

— axiom.bib —

```
@article{Gutt95,
  author = "Guttman, Joshua D. and Ramsdell, John D. and Wand, Mitchell",
  title = {{VLISP: A Verified Implementation of Scheme}},
  journal = "Lisp and Symbolic Computation",
  volume = "8",
  pages = "5-32",
  year = "1995",
  abstract =
    "The VL!SP project showed how to produce a comprehensively verified
    implementation for a programming language, namely Scheme.  This paper
    introduces two more detailed studies on VLISP [13, 21).  It summarizes
    the basic techniques that were used repeatedly throughout the effort.
    It presents scientific conclusions about the applicability of the
    these techniques as well as engineering conclusions about the crucial
    choices that allowed the verification to succeed.",
  paper = "Gutt95.pdf"
```

```
}
```

———-

1.17.8 H

— **axiom.bib** —

```
@article{Hama14,
  author = "Hamada, Tatsuyoshi",
  title = {{MathLibre: Personalizable Computer Environment for
            Mathematical Research}},
  journal = "ACM Communications in Computer Algebra",
  volume = "48",
  number = "3",
  year = "2014",
  abstract =
    "MathLibre is a project to archive free mathematical software and free
    mathematical documents and offer them on Live Linux system. MathLibre
    Project is the direct descendant of KNOPPIX/Math Project. It provides
    a desktop for mathematicians that can be set up easily and quickly.",
  paper = "Hama14.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Hamm95,
  author = "Hamming, Richard",
  title = {{Hamming, 'You and Your Research'}},
  link = "\url{https://www.youtube.com/watch?v=a1zDuOPkMSw}",
  year = "1995"
}
```

———-

— **axiom.bib** —

```
@article{Hanu90,
  author = "Hanus, Michael",
  title = {{Compiling Logic Programs with Equality}},
  journal = "LNCS",
  volume = "456",
  year = "1990",
  pages = "387-401",
  abstract =
    "Horn clause logic with equality is an amalgamation of functional and
    logic programming languages. A sound and complete operational
    semantics for logic programs with equality is based on resolution to
```

```
    solve literals, and rewriting and narrowing to evaluate functional
    expressions. This paper proposes a technique for compiling programs
    with these inference rules into programs of a low-level abstract
    machine which can be efficiently executed on conventional
    architectures. The presented approach is based on an extension of the
    Warren abstract machine (WAM). In our approach pure logic programs
    without function definitions are compiled in the same way as in the
    WAM-approach, and for logic programs with function definitions
    particular instructions are generated for occurrences of functions
    inside clause bodies. In order to obtain an efficient implementation
    of functional computations, a stack of occurrences of function symbols
    in goals is managed by the abstract machine. The compiler generates
    the necessary instructions for the efficient manipulation of the
    occurrence stack from the given equational logic programs.",
  paper = "Hanu90.pdf"
}
```

———-

— axiom.bib —

```
@inproceedings{Harp93b,
  author = "Harper, Robert and Lillibridge, Mark",
  title = {{Explicit Polymorphism and CPS Conversion}},
  booktitle = "Symp. of Principles of Programming Languages",
  publisher = "ACM Press",
  year = "1993",
  pages = "206=219",
  abstract =
    "We study the typing properties of CPS conversion for an extension
    of F$\omega$  with control operators.  Two classes of evaluation
    strategies are considered, each with call-by-name and call-by-value
    variants.  Under the 'standard' strategies, constructor abstractions
    are values, and constructor applications can lead to non-trivial
    control effects.  In contrast, the 'ML-like' strategies evaluate
    beneath constructor abstractions, reflecting the usual interpretation
    of programs in languages based on implicit polymorphism.  Three
    continuation passing style sub-languages are considered, one on which
    the standard strategies coincide, one on which the ML-like
    strategies coincide, and one on which all the strategies coincide.
    Compositional, type-preserving CPS transformation algorithms are
    given for the standard strategies, resulting in terms on which all
    evaluation strategies coincide.  This has as a corollary the
    soundness and termination of well-typed programs under the standard
    evaluation strategies.  A similar result is obtained for the ML-like
    call-by-name strategy .  In contrast, such results are obtained for
    the call-by-value ML-like strategy only for a restricted
    sub-language in which constructor abstractions are limited to
    values.",
  paper = "Harp93b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Harr09,
  author = "Harrison, John",
  title = {{Handbook of Practical Logic and Automated Reasoning}},
  publisher = "Cambridge University Press",
  year = "2009",
  isbn = "978-0-521-89957-4",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Hatt97,
  author = "Hatton, Les",
  title = {{Software Failure: Follies and Fallacies}},
  journal = "IEEE Review",
  volume = "43",
  number = "2",
  year = "1997",
}
```

———-

— **axiom.bib** —

```
@article{Hell17,
  author = "Hellman, Martin E.",
  title = {{Cybersecurity, Nuclear Security, Alan Turing, and
            Illogical Logic}},
  journal = "J. ACM",
  volume = "60",
  number = "12",
  pages = "52-59",
  year = "2017",
  link = "\url{https://cacm.acm.org/magazines/2017/12/223042-cybersecurity-nuclear-security-alan-turing-and-illo
}
```

———-

— **axiom.bib** —

```
@incollection{Hoar72,
  author = "Hoare, C.A.R",
  title = {{Notes on Data Structuring}},
  booktitle = "Structured Programming",
  publisher = "Academic Press",
```

```
  year = "1972",
  pages = "83-174"
}
```

———-

— **axiom.bib** —

```
@misc{Huda89,
  author = "Hudak, Paul",
  title = {{The Conception, Evolution, and Application of Functional
           Programming Languages}},
  link = "\url{http://haskell.cs.yale.edu/wp-content/uploads/2011/01/cs.pdf}",
  year = "1989",
  abstract =
    "The foundations of functional programming languages are examined from
    both historical and technical perspectives.  Their evolution is traced
    through several critical periods: early work on lambda calculus and
    combinatory calculus, Lisp, Iswim, FP, ML, and modern functional
    languages such as Miranda 1 and Haskell.  The fundamental premises on
    which the functional programming methodology stands are critically
    analyzed with respect to philosophical, theoretical, and pragmatic
    concerns. Particular attention is paid to the main features that
    characterize modern functional languages: higher-order functions, lazy
    evaluation, equations and pattern-matching, strong static typing and
    type inference, and data abstraction.  In addition, current research
    areassuch as parallelism, non-determinism, input/output, and
    state-oriented computationsare examined with the goal of predicting
    the future development and application of functional languages.",
  paper = "Huda89.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Huet75,
  author = "Huet, Gerard P.",
  title = {{A Unification Algorithm for typed $\lambda$-Calculus}},
  journal = "Theoretical Computer Science",
  volume = "1",
  number = "1",
  pages = "25-57",
  year = "1975",
  abstract =
    "A semi-decision algorithm is presented, to search for unification of
    formulas in typed $\omega$-order $\lambda$-calculus, and its
    correctness is proved.

    It is shown that the search space is significantly smaller than the
    one for finding the most general unifiers. In particluar, our search
```

```
    is not redundant. This allows our algorithm to have good
    directionality and convergence properties.",
  paper = "Huet75.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Huet78,
  author = "Huet, Gerard P. and Lang, Bernard",
  title = {{Proving and Applying Program Transformations Expressed
            with Second-Order Patterns}},
  journal = "Acta Informatica",
  volume = "11",
  number = "1",
  pages = "31-55",
  year = "1978",
  abstract =
    "We propose a program transformation method based on rewriting-rules
    composed of second-order schemas. A complete second-order matching
    algorithm is presented that allows effective use of these rules. We
    show how to formally prove the correctness of the rules using a
    denotational semantics for the programming language. We establish the
    correctness of the transformation method itself, and give techniques
    pertaining to its actual implementation. The paper is illustrated with
    recursion removal examples.",
  paper = "Huet78.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hugh90,
  author = "Hughes, John",
  title = {{Why Functional Programming Matters}},
  booktitle = "Research Topics in Functional Programming",
  publisher = "Addison-Wesley",
  year = "1990",
  pages = "17-42",
  abstract =
    "As software becomes more and more complex, it is more and more
    important to structure it well.  Well-structured software is easy to
    write and to debug, and provides a collection of modules that can be
    reused to reduce future programming costs.  In this paper we show that
    two fea- tures of functional languages in particular, higher-order
    functions and lazy evaluation, can contribute significantly to
    modularity.  As examples, we manipulate lists and trees, program
    several numerical algorithms, and implement the alpha-beta heuristic
    (an algorithm from Artificial Intelligence used in game-playing
    programs). We conclude that since modularity is the key to successful
```

```
    programming, functional programming offers important advantages for
    software development.",
  paper = "Hugh90.pdf",
  keywords = "printed"
}
```

———-

1.17.9 I

1.17.10 J

— **axiom.bib** —

```
@article{Jenk11,
  author = "Jenks, Richard D.",
  title = {{The 2011 Richard D. Jenks Memorial Prize}},
  journal = "ACM Communications in Computer Algebra",
  volume = "45",
  number = "4",
  year = "2011",
  abstract =
    "The 2011 Richard D. Jenks Memorial Prize for Excellence in
    Software Engineering Applied to Computer Algebra was presented by
    members of the Prize Selection Committee and its chair, Erich
    Kaltofen, at ISSAC and SNC in San Jose, CA, on June 9, 2011 to the
    Maple Project at the University of Waterloo.",
  paper = "Jenk11.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Jenk13,
  author = "Jenks, Richard D.",
  title = {{The 2013 Richard D. Jenks Memorial Prize}},
  journal = "ACM Communications in Computer Algebra",
  volume = "47",
  number = "4",
  year = "2013",
  abstract =
    "The 2013 Richard D. Jenks Memorial Prize for Excellence in
    Software Engineering Applied to Computer Algebra was announced by
    members of the Prize Selection Committee, Mark Giesbrecht
    representing its chair Erich Kaltofen, at ISSAC in Boston, MA, on
    June 28, 2013 to have been awared to Professor William Arthur
    Stein of the Sage Project at the University of Washington.",
  paper = "Jenk13.pdf",
  keywords = "axiomref"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Jenk15,
  author = "Jenks, Richard D.",
  title = {{The 2015 Richard D. Jenks Memorial Prize}},
  journal = "ACM Communications in Computer Algebra",
  volume = "49",
  number = "4",
  year = "2015",
  abstract =
    "The 2015 Richard D. Jenks Memorial Prize was awarded on October
    30, 2015 at the Fields Institute in Toronto during the Major
    Thematic Program on Computer Algebra to Professor Victor Shoup for
    NTL: A Library for doing Number Theory.",
  paper = "Jenk15.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Jenk17,
  author = "Jenks, Richard D.",
  title = {{The 2017 Richard D. Jenks Memorial Prize}},
  journal = "ACM Communications in Computer Algebra",
  volume = "51",
  number = "4",
  year = "2017",
  abstract =
    "The 2017 Richard D. Jenks Memorial Prize for Excellence in
    Software Engineering Applied to Computer Algebra has been awarded
    to Stephen Wolfram for Wolfram-Alpha and Mathematica.",
  paper = "Jenk17.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{John96,
  author = "Johnson, C.W.",
  title = {{Literate Specifications}},
  journal = "Software Engineering Journal",
  volume = "11",
  number = "4",
```

```
  year = "1996",
  pages = "225-237",
  paper = "John96.pdf",
  abstract =
    "The increasing scale and complexity of software is imposing serious
    burdens on many industries. Formal notations, such as Z, VDM and
    temporal logic, have been developed to address these problems. There
    are, however, a number of limitations that restrict the use of
    mathematical specifications far large-scale software development. Many
    projects take months or years to complete. This creates difficulties
    because abstract mathematical requirements cannot easily be used by
    new members of a development team to understand past design
    decisions. Formal specifications describe what software must do, they
    do not explain why it must do it. In order to avoid these limitations,
    a literate approach to software engineering is proposed. This
    technique integrates a formal specification language and a semi-formal
    design rationale.  The Z schema calculus is usecl to represent what a
    system must do. In contrast, the Questions, Options and Criteria
    notation is used to represent the justifications that lie behind
    development decisions. Empirical evidence is presented that suggests
    the integration of these techniques provides significant benefits over
    previous approaches to mathematical analysis and design techniques. A
    range of tools is described that have been developed to support our
    literate approach to the specification of large-scale sohare systems.",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Jone96,
  author = "Jones, Simon > Peyton",
  title = {{Compiling Haskell by Program Transformation: A Report from
            the Trenches}},
  booktitle = "Proc. European Symposium on Programming",
  year = "1996",
  publisher = "Eurographics",
  abstract =
    "Many compilers do some of their work by means of
    correctness-preseving, and hopefully performance-improving, program
    transformations. The Glasgow Haskell Compiler (GHC) takes this idea
    of 'compilation by transformation' as its war-cry, trying to express
    as much as possible of the compilation process in the form of
    program transformations.

    This paper reports on our practical experience of the
    transformational approach to compilation, in the context of a
    substantial compiler.",
  paper = "Jone96.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Jone97,
  author = "Jones, Simon Peyton and Meijer, Erik",
  title = {{Henk: A Typed Intermediate Language}},
  year = "1997",
  link =
  "\url{https://www.microsoft.com/en-us/research/wp-content/uploads/1997/01/henk.pdf}",
  abstract =
    "There is a growing interest in the use of richly-typed
    intermediate languages in sophisticated compilers for
    higher-order, typed source languages. These intermediate languages
    are typically stratified, involving terms, types, and kinds. As
    the sophistication of the type system increases, there three
    levels begin to look more and more similar, so an attraictive
    approach is to use a single syntax, and a single data type in the
    compiler, to represent all three.

    The theory of so-called {\sl pure type systems} amkes precisely
    such an identification. This paper describes Henk, a new typed
    intermediate langugage based closely on a particuarl pure type
    system, {\sl the lambda cube}. On the way we give a tutorial
    introduction to the lambda cube.",
  paper = "Jone97.pdf",
  keywords = "printed"
}
```

———-

1.17.11 K

— **axiom.bib** —

```
@article{Kaes88,
  author = "Kaes, Stefan",
  title = {{Parametric Overloading in Polymorphic Programming Languages}},
  journal = "LNCS",
  volume = "300",
  pages = "131-144",
  year = "1988",
  abstract =
    "The introduction of unrestricted overloading in languagues with type
    systems based on implicit parametric potymorphism generally destroys
    the principal type property: namely that the type of every expression
    can uniformly be represented by a single type expression over some set
    of type variables.  As a consequence, type inference in the presence
    of unrestricted overloading can become a NP-complete problem.  In
    this paper we define the concept of parametric overloading as a
    restricted form of overloading which is easily combined with
    parametric polymorphism.  Parametric overloading preserves the
    principal type property, thereby allowing the design of efficient type
    inference algorithms.  We present sound type deduction systems, both
```

```
    for predefined and programmer defined overloading.  Finally we state
    that parametric overloading can be resolved either statically, at
    compile time, or dynamically, during program execution.",
  paper = "Kaes88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Kaes92,
  author = "Kaes, Stefan",
  title = {{Type Inference inthe Presence of Overloading, Subtyping and
            Recursive Types}},
  journal = "ACM Lisp Pointers",
  volume = "5",
  number = "1",
  year = "1992",
  pages = "193-204",
  abstract =
    "We present a unified approach to type inference in the presence of
    overloading and coercions based on the concept of {\sl constrained
    types}. We define a generic inference system, show that subtyping and
    overloading can be treated as a special instance of this system and
    develop a simple algorithm to compute principal types. We prove the
    decidability of type inference for hte class of {\sl decomposable
    predicates} and deelop a canonical representation for principal types
    based on {\sl most accurate simplifications} of constraint
    sets. Finally, we investigate the extension of our techniques to
    {\sl recursive types}.",
  paper = "Kaes92.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Kant98,
  author = "Kant, Immanuel",
  title = {{The Critique of Pure Reason}},
  publisher = "Cambridge University Press",
  year = "1998",
  isbn = "0-521-35402-1",
  link = "\url{http://strangebeautiful.com/other-texts/kant-first-critique-cambridge.pdf}",
  paper = "Kant98.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Keim09,
  author = "Keimel, Klaus and Plotkin, Gordon D.",
  title = {{Predicate Transformers for Extended Probability and
            Non-Determinism}},
  journal = "Math. Struct. in Comp. Science",
  volume = "19",
  pages = "501-539",
  year = "2009",
  abstract =
    "We investigate laws for predicate transformers for the combination of
    non-deterministic choice and (extended) probabilistic choice, where
    predicates are taken to be functions to the extended non-negative
    reals, or to closed intervals of such reals. These predicate
    transformers correspond to state transformers, which are functions to
    conical powerdomains, which are the appropriate powerdomains for the
    combined forms of non-determinism. As with standard powerdomains for
    non-deterministic choice, these come in three flavours  lower, upper
    and (order-)convex  so there are also three kinds of predicate
    transformers. In order to make the connection, the powerdomains are
    first characterised in terms of relevant classes of functionals.

    Much of the development is carried out at an abstract level, a kind of
    domain-theoretic functional analysis: one considers d-cones, which are
    dcpos equipped with a module structure over the non-negative extended
    reals, in place of topological vector spaces. Such a development still
    needs to be carried out for probabilistic choice per se ; it would
    presumably be necessary to work with a notion of convex space rather
    than a cone.",
  paper = "Keim09.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Kell13,
  author = "Keller, C.",
  title = {{A Matter of Trust: Skeptical Commuication Between Coq and
            External Provers}},
  school = "Ecole Polytechnique",
  year = "2013",
  link =
"\url{https://www.lri.fr/~keller/Documents-recherche/Publications/thesis13.pdf}",
  abstract =
    "This thesis studies the cooperation between the Coq proof assistant
    and external provers through proof witnesses.  We concentrate on two
    different kinds of provers that can return certicates: first, answers
    coming from SAT and SMT solvers can be checked in Coq to increase both
    the confidence in these solvers and Coq 's automation; second,
    theorems established in interactive provers based on Higher-Order
    Logic can be exported to Coq and checked again, in order to offer the
    possibility to produce formal developments which mix these two
```

```
    different logical paradigms.  It ended up in two software : SMTCoq, a
    bi-directional cooperation between Coq and SAT/SMT solvers, and
    HOLLIGHTCOQ, a tool importing HOL Light theorems into Coq.

    For both tools, we took great care to define a modular and efficient
    architecture, based on three clearly separated ingredients: an
    embedding of the formalism of the external tool inside Coq which is
    carefully translated into Coq terms, a certified checker to establish
    the proofs using the certicates and a Ocaml preprocessor to transform
    proof witnesses coming from different provers into a generic
    certificate. This division allows that a change in the format of proof
    witnesses only affects the preprocessor, but no proved Coq code.
    Another fundamental component for efficiency and modularity is
    computational reflection, which exploits the computational power of
    Coq to establish generic and small proofs based on the certicates.",
  paper = "Kell13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Klae84,
  author = "Klaeren, Herbert A.",
  title = {{A Constructive Method for Abstract Algebraic Software
            Specification}},
  journal = "Theoretical Computer Science",
  vollume = "30",
  year = "1984",
  pages = "139-204",
  abstract =
    "A constructive method for abstract algebraic software
    specification is presented, where the operations are not
    implicitly specified by equations but by an explicit recursion on
    the generating operations of an algebra characterizing the
    underlying data structure. The underlying algebra itself may be
    equationally specified since we cannot assume that all data
    structures will correspond to free algebras. This implies that we
    distinguish between generating and defined operations and that the
    underlying algebra has a mechanism of well-founded decomposition
    w.r.t. the generating operations. We show that the explicit
    specification of operations using so-called structural recursive
    schemata offers advantages over purely equational specifications,
    especially concerning the safeness of enrichments, the ease of
    semantics description and the separation between the underlying
    data structure and the operations defined on it.",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Klee52,
  author = "Kleene, Stephen Cole",
  title = {{Introduction to MetaMathematics}},
  year = "1952",
  publisher = "Ishi Press International",
  isbn = "978-0923891572",
  paper = "Klee52.pdf",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@incollection{Knut70,
  author = "Knuth, Donald E. and Bendix, Peter B.",
  title = {{Simple Word Problems in Unversal Algebras}},
  booktitle = "Computational Problems in Abstract Algebra",
  editor = "Leech, John",
  publisher = "Pergamon Press",
  isbn = "08-012975-7",
  year = "1970",
  pages = "263-298",
  paper = "Knut70.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kohl17,
  author = "Kohlhase, Michael and De Feo, Luca and Muller, Dennis and
            Pfeiffer, Markus and Rabe, Florian and Thiery, Nicolas M.
            and Vasilyev, Victor and Wesing, Tom",
  title = {{Knowledge-Based Interoperability for Mathematical Software
            Systems}},
  booktitle = "7th Int. Conf. on Mathematical Aspects of Computer and
               Information Sciences",
  publisher = "Springer",
  year = "2017",
  pages = "195-210",
  isbn = "9783319724539",
  abstract =
    "There is a large ecosystem of mathematical software systems.
    Individually, these are optimized for particular domains and
    functionalities, and together they cover many needs of practical and
    theoretical mathematics. However, each system specializes on one area,
    and it remains very difficult to solve problems that need to involve
    multiple systems. Some integrations exist, but the are ad-hoc and have
    scalability and maintainability issues. In particular, there is not
    yet an interoperability layer that combines the various systems into a
```

```
    virtual research environment (VRE) for mathematics.

    The OpenDreamKit project aims at building a toolkit for such VREs. It
    suggests using a central system-agnostic formalization of mathematics
    (Math-in-the-Middle, MitM) as the needed interoperability layer. In
    this paper, we conduct the first major case study that instantiates
    the MitM paradigm for a concrete domain as well as a concrete set of
    systems. Specifically, we integrate GAP, Sage, and Singular to perform
    computation in group and ring theory.

    Our work involves massive practical efforts, including a novel
    formalization of computational group theory, improvements to the
    involved software systems, and a novel mediating system that sits at
    the center of a star-shaped integration layout between mathematical
    software systems.",
  paper = "Kohl17.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Koze93,
  author = "Kozen, Dexter and Palsberg, Jens and Schwartzbach, Michael I.",
  title = {{Efficient Recursive Subtyping}},
  journal = "Mathematical Structures in Computer Science",
  volume = "5",
  number = "1",
  pages = "113-125",
  year = "1995",
  abstract =
    "Subtyping in the presence of recursive types for the
    $\lambda$-calculus was studied by Amadio and Cardelli in 1991. In that
    paper they showed that the problem of deciding whether one recursive
    type is a subtype of another is decidable in exponential time.  In
    this paper we give an $O(n^2)$ algorithm.  Our algorithm is based on a
    simplification of the definition of the subtype relation, which allows
    us to reduce the problem to the emptiness problem for a certain finite
    automaton with quadratically many states.  It is known that equality
    of recursive types and the covariant Bohm order can be decided
    efficiently by means of finite automata, since they are just language
    equality and language inclusion, respectively.  Our results extend the
    automata-theoretic approach to handle orderings based on
    contravariance.",
  paper = "Koze93.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Kralxx,
  author = "Krall, Andreas",
  title = {{Implementation Techniques for Prolog}},
  link = "\url{https://pdfs.semanticscholar.org/fdbf/aa46bf6ab2148595f638fe9afe97033583ee.pdf}",
  year = "unknown",
  abstract =
    "This paper is a short survey about currently used implementation
    techniques for Prolog. It gives an introduction to unification and
    resolution in Prolog and presents the memory model and a basic
    execution model. These models are expanded to the Vienna Abstract
    Machine (VAM) with its two versions, the VAM${}_{2p}$ and the
    VAM${}_{1p}$, and the most famous abstract machine, the Warren
    Abstract Machine (WAM). The continuation passing style model of
    Prolog, binary Prolog, leands to the BinWAM. Abstract
    interpretation can be applied to gather information about a
    program. This information is used in the generation of very
    specialized machine code and in optimizations like clause indexing
    and instruction scheduling on each kind of abstract machine.",
  paper = "Kralxx.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kran86,
  author = "Kranz, David and Kelsey, Richard and Rees, Jonathan and
            Hudak, Paul and Philbin, James and Adams, Norman",
  title = {{ORBIT: An Optimizing Compiler for Scheme}},
  booktitle = "SIGLAN '86",
  publisher = "ACM",
  pages = "219-233",
  year = "1986",
  abstract =
    "In this paper we describe an optimizing compiler for Scheme
    called {\sl Orbit} that incorporates our experience with an
    earlier Scheme compiler called TC, together with some ideas from
    Steele's Rabbit compiler. The three main design goals have been
    correctness, generating very efficient compiled code, and
    portability.",
  paper = "Kran86.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Krip75,
  author = "Kripke, Saul",
  title = {{Outline of a Theory of Truth}},
```

```
  journal = "Journal of Philosophy",
  volume = "72",
  number = "19",
  year = "1975",
  pages = "690-716",
  paper = "Krip75.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Kroh93,
  author = "Krohnfeldt, Jed and Steury, Craig",
  title = {{Frolic: Logic Programming wiht Frobs}},
  type = "technical report",
  number = "86-08",
  institution = "University of Utah",
  comment = "Utah PASS Project OpNote",
  year = "1993",
  code = "PAS/frolic.tgz",
  abstract =
    "This paper describes Frolic, a logic programming system written
    in Common Lisp. It provides logic programming capabilities for
    Lisp programmers. In addition to being a Prolog in Lisp, Frolic
    has capabilities for reasoning about {\sl frobs}. A frob is a data
    structure that gives the power and flexibility of both frame and
    object data types. This paper gives a brief introduction to the
    use and implementation of Frolic.",
  paper = "Kroh93.pdf",
  keywords = "printed, DONE"
}
```

———-

1.17.12 L

— **axiom.bib** —

```
@article{Lamp78,
  author = "Lamport, Leslie",
  title = {{The Specification and Proof of Correctness of Interactive
            Programs}},
  journal = "LNCS",
  volume = "75",
  year = "1978",
  abstract =
    "method production assertional rules specified, is proved is modified
    correctly to accept and interactive is described, method that a
    program A program of specifying and to permit typed its implementation
```

```
    correct.  by and the Floyd-Hoare implements format programs one to
    prove its specification.  input is formally with a TECO program.",
  paper = "Lamp78.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Lamp81,
  author = "Lamport, Leslie and Owicki, Susan",
  title = {{Program Logics and Program Verification}},
  journal = "LNCS",
  volume = "131",
  pages = "197-199",
  year = "1981",
  paper = "Lamp81.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Lamp03,
  author = "Lamport, Leslie",
  title = {{The Future of Computing: Logic or Biology}},
  link =
    "\url{https://lamport.azurewebsites.net/pubs/future-of-computing.pdf}",
  year = "2003",
  paper = "Lamp03.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Laza01,
  author = "Lazard, Daniel",
  title = {{Solving Systems of Algebraic Equations}},
  journal = "ACM SIGSAM Bulletin",
  volume = "35",
  number = "3",
  year = "2001",
  abstract =
    "Let $f_1,\ldots,f_k$ be $k$ multivariate polynomials which have
    a finite number of common zeros in the algebraic closure of the ground
    field, counting the common zeros at infinity. An algorithm is
    given and proved which reduces the computations of these zeros to
```

```
    the resolution of a single univariate equation whose degree is the
    number of common zeros. This algorithm gives the whole algebraic
    and geometric structure of the set of zeros (multiplicities,
    conjugate zeros,...). When all the polynomials have the same degree,
    the complexity of this algorithm is polynomial relative to the generic
    number of solutions.",
  paper = "Laza01.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Leec70,
  author = "Leech, John",
  title = {{Computational Problems in Abstract Algebra}},
  publisher = "Pergamon Press",
  year = "1970",
  isbn = "08-012975-7",
  paper = "Leec70.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Legu76,
  author = "LeGuin, Ursula K.",
  title = {{The Left Hand of Darkness}},
  publisher = "Penguin Random House",
  year = "1976",
  isbn = "978-1-101-66539-8"
}
```

———-

— **axiom.bib** —

```
@techreport{Lero90,
  author = "Leroy, Xavier",
  title = {{The ZINC Experiment: An Economical Implementation of the
            ML Language.}},
  type = "technical report",
  institution = "Institut National de Recherche en Informatique et en
                 Automatique",
  number = "117",
  year = "1990",
  abstract =
    "This report details the design and implementation of the ZINC
    system. This is an implementation of the ML language, inended to
    serve as a test field for various extensions of the language, and
```

```
    for new implementation techniques as well. This system is strongly
    oriented toward separate compilation and the production of small,
    standalone programs; type safety is ensured by a Modula-2-like
    module system. ZINC uses simple, portable techniques, such as
    bytecode interpretation; a sophisticated execution model helps
    counterbalance the interpretation overhead. Other highlights
    include an efficient implementation of records with inclusion
    (subtyping).",
  paper = "Lero90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Lero98,
  author = "Leroy, Xavier",
  title = {{An Overview of Types in Compilation}},
  journal = "LNCS",
  volume = "1473",
  year = "1998",
  publisher = "Springer-Verlang",
  pages = "1-8",
  paper = "Lero98.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Lero08,
  author = "Leroy, Xavier",
  title = {{Formal Verification of a Realistic Compiler}},
  year = "2008",
  link = "\url{https://xavierleroy.org/publi/compcert-CACM.pdf}",
  abstract =
    "This paper reports on the development and formal verification
    (proof of semantic preservation) of CompCert, a compiler from
    Clight (a large subset of the C programming language) to PowerPC
    assembly code, using the Coq proof assistant both for programming
    the compiler and for proving its correctness. Such a verificed
    compiler is useful in the context of critical software and its
    formal verification: the verification of the compiler guarantees
    that the safety properties proved on the source code hold for the
    executable compiled code as well.",
  paper = "Lero08.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Lero09,
  author = "Leroy, Xavier",
  title = {{A Formally Verified Compiler Back-end}},
  journal = "Logic in Computer Science",
  volume = "43",
  number = "4",
  pages = "363-446",
  year = "2009",
  abstract =
    "This article describes the development and formal verification
    (proof of semantic preservation) of a compiler back-end from Cminor
    (a simple imperative intermediate language) to PowerPC assembly code,
    using the Coq proof assistant both for programming the compiler and
    for proving its soundness. Such a verified compiler is useful in the
    context of formal methods applied to the certification of critical
    software: the verification of the compiler guarantees that the safety
    properties proved on the source code hold for the executable compiled
    code as well.",
  paper = "Lero09.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Lest89,
  author = "Lester, David R.",
  title = {{Combinator Graph Reduction: A Congruence and its Applications}},
  year = "1989",
  school = "Oxford University",
  isbn = "0-902928-55-4",
  abstract =
    "The G-Machine is an efficient implementation of lazy functional
    languages developed by Augustsson and Johnsson. This thesis may be
    read as a formal mathematical proof that the G-machine is correct with
    respect to a denotational semantic specification of a simple
    language. It also has more general implications. A simple lazy
    functional language is defined both denotationally and operationally;
    both are defined to handle erroneous results. The operational
    semantics models combinator graph reduction, and is based on reduction
    to weak head normal form. The two semantic definitions are shown to be
    congruent. Because of error handling the language is not
    confluent. Complete strictness is shown to be a necessary and
    sufficient condition for changing lazy function calls to strict
    ones. As strictness analyses are usually used with confluent
    languages, methods are discussed to restore this property. The
    operational semantic model uses indirection nodes to implement
    sharing. An alternative, which is without indirection nodes, is shown
    to be operationally equivalent for terminating programs. The G-machine
    is shown to be a representation of the combinator graph reduction
    operational model. It may be represented by the composition of a small
```

```
    set of combinators which correspond to an abstract machine instruction
    set. Using a modified form of graph isomorphism, alternative sequences
    of instructions are shown to be isomorphic, and hence may be used
    interchangeably.",
  paper = "Lest89.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Leto04,
  author = "Letouzey, Pierre",
  title = {{A New Extraction for Coq}},
  booktitle = "Types for Proofs and Programs. TYPES 2002",
  publisher = "Springer",
  pages = "617-635",
  year = "2004",
  abstract =
    "We present here a new extraction mechanism for the Coq proof
    assistant. By extraction, we mean automatic generation of
    functional code from Coq proofs, in order to produce certified
    programs. In former versions of Coq, the extraction mechanism
    suffered several limitations and in particular worked only with a
    subset of the language. We first discuss difficulties encountered
    and solutions proposed to remove these limitations. Then we give a
    proof of correctness for a theoretical model of the new
    extraction. Finally we describe the actual implementation.",
  paper = "Leto04.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Lohx18,
  author = "Loh, Andres and McBride, Conor and Swierstra, Wouter",
  title = {{Simply Easy! An Implementation of a Dependently Typed
            Lambda Calculus}},
  link = "\url{http://strictlypositive.org/Easy.pdf}",
  year = "2018",
  abstract =
    "We present an implementation in Haskell of a dependently-typed
    lambda calculus that can be used as the core of a programming
    language. We show that a dependently-typed lambda calculus is no
    more difficult to implement than other typed lambda calculi. In fact,
    our implementation is almost as easy as an implementation of the
    simply typed lambda calculus, which we emphasize by discussing
    the modifications necessary to go from one to the other. We explain
    how to add data types and write simple programs in the core
    language, and discuss the steps necessary to build a full-fledged
```

```
    programming language on top of our simple core.",
  paper = "Lohx18.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Luxx16,
  author = "Lu, Eric",
  title = {{Barendregt's Cube and Programming with Dependent Types}},
  link = "\url{https://www.seas.harvard.edu/courses/cs252/2016fa/15.pdf}",
  paper = "Luxx16.pdf",
  keywords = "printed"
}
```

———-

1.17.13 M

— **axiom.bib** —

```
@article{Mann72,
  author = "Manna, Zohar and Vuillemin, Jean",
  title = {{Fixpoint Approach to the Theory of Computation}},
  journal = "Communications of the ACM",
  volume = "15",
  number = "7",
  year = "1972",
  pages = "828-836",
  abstract =
    "Following the fixpoint theory of Scott, the semantics of computer
    programs are defined in terms of the least fixpoints of recursive
    programs. This allows not only the justification of all existing
    verification techniques, but also their extension to the handling, in
    a uniform manner of various properties of computer programs, including
    correctness, termination, and equivalence.",
  paper = "Mann72.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Mann85,
  author = "Manna, Zohar and Waldinger, Richard",
  title = {{The Logical Basis for Computer Programming (Vol 1)}},
  publisher = "Addison-Wesley",
```

```
  year = "1985",
  isbn = "0-201-15260-2",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Mano03,
  author = "Manolios, Panagiotis and Moore, J Strother",
  title = {{Partial Functions in ACL2}},
  journal = "J. of Automated Reasoning",
  volume = "31",
  pages = "107-127",
  year = "2003",
  abstract =
    "We describe a method for introducing 'partial functions' into ACL2,
    that is, functions not defined everywhere. The function 'definitions'
    are actually admitted via the encapsulation principle: the new
    function symbol is constrained to satisfy the appropriate
    equation. This is permitted only when a witness function can be
    exhibited, establishing that the constraint is satisfiable. Of
    particular interest is the observation that every tail recursive
    definition can be witnessed in ACL2. We describe a macro that allows
    the convenient introduction of arbitrary tail recursive functions, and
    we discuss how such functions can be used to prove theorems about
    state machine models without reasoning about clocks or counting the
    number of steps until termination. Our macro for introducing partial
    functions also permits a variety of other recursive schemes, and we
    briefly illustrate some of them.",
  paper = "Mano03.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Mart82,
  author = "Martelli, Alberto and Montanari, Ugo",
  title = {{An Efficient Unification Algorithm}},
  journal = "ACM TOPLAS",
  volume = "4",
  number = "2",
  pages = "258-282",
  year = "1982",
  abstract =
    "The unification problem in first-order predicate calculus is
    described in general terms as the solution of a system of equations,
    and a nondeterministic algorithm is given.  A new unification
    algorithm, characterized by having the acyclicity test efficiently
```

```
    embedded into it, is derived from the nondeterministic one, and a
    PASCAL implementation is given.  A comparison with other well-known
    unification algorithms shows that the algorithm described here
    performs well in all cases.",
  paper = "Mart82.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Mazu07,
  author = "Mazur, Barry",
  title = {{When is One Thing Equal to Some Other Thing?}},
  link = "\url{http://www.math.harvard.edu/~mazur/preprints/when_is_one.pdf}",
  year = "2007",
  paper = "Mazu07.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Mcca60,
  author = "McCarthy, John",
  title = {{Recursive Functions of Symbolic Expressions and Their
            Computation by Machine, Part I}},
  journal = "CACM",
  volume = "3",
  pages = "184-195",
  year = "1960",
  paper = "Mcca60.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@incollection{Mcca63,
  author = "McCarthy, John",
  title = {{A Basis for a Mathematical Theory of Computation}},
  booktitle = "Computer Programming and Formal Systems",
  publisher = "Elsevier",
  year = "1963",
  paper = "Mcca63.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mcca67,
  author = "McCarthy, John and Painter, James",
  title = {{Correctness of a Compiler for Arithmetic Expressions}},
  booktitle = "Proc. Symp. in Applied Mathematics",
  publisher = "American Mathematical Society",
  year = "1967",
  paper = "Mcca67.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Mcca96,
  author = "McCarthy, J.",
  title = {{Towards a Mathematical Science of Computation}},
  link = "\url{http://www-formal.stanford.edu/jmc/towards.pdf}",
  year = "1996",
  paper = "Mcca96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mcdo97,
  author = "McDowell, Raymond and Miller, Dale",
  title = {{A Logic for Reasoning with Higher-Order Abstract Syntax}},
  booktitle = "LICS'97",
  publisher = "IEEE",
  year = "1997",
  link = "\url{http://www.lix.polytechnique.fr/~dale/papers/lics97.pdf}",
  abstract =
    "Logical frameworks based on intuitionistic or linear logics with
    higher-type quantification have been successfully used to give
    high-level, modular, and formal specifications of many important
    judgments in the area of programming languages and inference
    systems. Given such specifications, it is natural to consider proving
    properties about the specified systems in the framework: for example,
    given the specification of evaluation for a functional programming
    language, prove that the language is deterministic or that the
    subject-reduction theorem holds.  One challenge in developing a
    framework for such reasoning is that higher-order abstract syntax
    (HOAS), an elegant and declarative treatment of object-level
    abstraction and substitution, is difficult to treat in proofs
    involving induction.  In this paper, we present a meta-logic that can
    be used to reason about judgments coded using HOAS; this meta-logic is
    an extension of a simple intuitionistic logic that admits higher-order
```

```
    quantification over simply typed $\lambda$-terms (key ingredients for
    HOAS) as well as induction and a notion of definition. The latter
    concept of a definition is a proof-theoretic device that allows
    certain theories to be treated as 'closed' or as defining fixed
    points. The resulting meta-logic can specify various logical
    frameworks and a large range of judgments regarding programming
    languages and inference systems.  We illustrate this point through
    examples, including the admissibility of cut for a simple logic and
    subject reduction, determinacy of evaluation, and the equivalence of
    SOS and natural semantics presentations of evaluation for a simple
    functional programming language.",
  paper = "Mcdo97.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mcdo02,
  author = "McDowell, Raymond and Miller, Dale",
  title = {{Reasoning with Higher-Order Abstract Syntax in a Logical
            Framework}},
  journal = "ACM Trans. Computational Logic",
  volume = "3",
  year = "2002",
  pages = "80-136",
  link = "\url{http://www.lix.polytechnique.fr/~dale/papers/mcdowell01.pdf}",
  abstract =
    "Logical frameworks based on intuitionistic or linear logics with
    higher-type quantification have been successfully used to give
    high-level, modular, and formal specifications of many important
    judgments in the area of programming languages and inference systems.
    Given such specifications, it is natural to consider proving
    properties about the specified systems in the framework: for example,
    given the specification of evaluation for a functional programming
    language, prove that the language is deterministic or that evaluation
    preserves types.  One challenge in developing a framework for such
    reasoning is that higher-order abstract syntax (HOAS), an elegant and
    declarative treatment of object-level abstraction and substitution, is
    difficult to treat in proofs involving induction.  In this paper, we
    present a meta-logic that can be used to reason about judgments coded
    using HOAS; this meta-logic is an extension of a simple intuitionistic
    logic that admits higher-order quantification over simply typed
    $\lambda$-terms (key ingredients for HOAS) as well as induction and a
    notion of definition. The latter concept of definition is a
    proof-theoretic device that allows certain theories to be treated as
    'closed' or as defining fixed points.  We explore the difficulties of
    formal meta-theoretic analysis of HOAS encodings by considering
    encodings of intuitionistic and linear logics, and formally derive the
    admissibility of cut for important subsets of these logics.  We then
    propose an approach to avoid the apparent tradeoff between the
    benefits of higher-order abstract syntax and the ability to analyze
    the resulting encodings.  We illustrate this approach through examples
```

```
    involving the simple functional and imperative programming languages
    $PCF$ and $PCF_{:=}$. We formally derive such properties as unicity of
    typing, subject reduction, determinacy of evaluation, and the
    equivalence of transition semantics and natural semantics
    presentations of evaluation.",
  paper = "Mcdo02.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Mich11,
  author = "Michaelson, Greg",
  title = {{Functional Programming Through Lambda Calculus}},
  year = "2011",
  publisher = "Dover",
  isbn = "978-0-486-47883-8",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Mili09,
  author = "Mili, Ali and Aharon, Shir and Nadkarni, Chaitanya",
  title = {{Mathematics for Reasoning about Loop Functions}},
  journal = "Science of Computer Programming",
  volume = "79",
  year = "2009",
  pages = "989-1020",
  abstract =
    "The criticality of modern software applications, the pervasiveness of
    malicious code concerns, the emergence of third-party software
    development, and the preponderance of program inspection as a quality
    assurance method all place a great premium on the ability to analyze
    programs and derive their function in all circumstances of use and all
    its functional detail. For C-like programming languages, one of the
    most challenging tasks in this endeavor is the derivation of loop
    functions. In this paper, we outline the premises of our approach to
    this problem, present some mathematical results, and discuss how these
    results can be used as a basis for building an automated tool that
    derives the function of while loops under some conditions.",
  paper = "Mili09.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Pren16,
  author = "Prengel, Alex",
  title = {{MIT axiom-math\_v8.14}},
  link = "\url{http://web.mit.edu/axiom-math_v8.14/}",
  year = "2016",
  keywords = "axiomref, TPDref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mitc84,
  author = "Mitchell, John",
  title = {{Type Inference and Type Containment}},
  booktitle = "Proc. Int. Symp. on Semantics of Data Types",
  publisher = "Springer",
  isbn = "3-540-13346-1",
  year = "1984",
  pages = "257-277",
  abstract =
    "Type inference, the process of assigning types to untyped
    expressions, may be motivated by the design of a typed language or
    semantical considerations on the meeaaings of types and expressions.
    A typed language GR with polymorphic functions leads to the GR
    inference rules. With the addition of an 'oracle' rule for equations
    between expressions, the GR rules become complete for a general class
    of semantic models of type inference. These inference models are
    models of untyped lambda calculus with extra structure similar to
    models of the typed language GR. A more specialized set of type
    inference rules, the GRS rules, characterize semantic typing when the
    functional type $a~$ , is interpreted as all elements of the model that
    map a to $~r$ and the polymorphic type $Vt,~0$ is interpreted as the
    intersection of all $o'(r)$. Both inference systems may be reformulated
    using rules for deducing containments between types. The advantage of
    the type inference rules based on containments is thatproofs
    correspond more closely to the structure of terms.",
  paper = "Mitc84.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mell15,
  author = "Mellies, Paul-Andre and Zeilberger, Noam",
  title = {{Functors are Type Refinement Systems}},
  booktitle = "POPL'15",
  publisher = "ACM",
  year = "2015",
  abstract =
    "The standard reading of type theory through the lens of category
```

```
    theory is based on the idea of viewing a type system as a category
    of well-typed terms. We propose a basic revision of this reading;
    rather than interpreting type systems as categories, we describe
    them as {\sl functors} from a category of typing derivations to a
    category of underlying terms. Then, turning this around, we
    explain how in fact {\sl any} functor gives rise to a generalized
    type system, with an abstract notion of type judgment, typing
    derivations and typing rules. This leads to a purely categorical
    reformulation of various natural classes of type systems as
    natural classes of functors.

    The main purpose of this paper is to describe the general
    framework (which can also be seen as providing a categorical
    analysis of {\sl refinement types}, and to present a few
    applications. As a larger case study, we revisit Reynold's paper
    on ``The Meaning of Types'' (2000), showing how the paper's main
    results may be reconstructed along these lines.",
  paper = "Mell15.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Moor96,
  author = "Moore, Andrew P. and Payne Jr., Charles N.",
  title = {{Increasing Assurance with LIterate Programming Techniques}},
  booktitle = "Comf. on Computer Assurance COMPASS'96",
  publisher = "National Institute of Standards and Technology",
  year = "1996",
  pages = "187-198",
  abstract =
    "The assurance argument that a trusted system satisfies its
    information security requirements must be convincing, because the
    argument supports the accreditation decision to allow the computer to
    process classified information in an operational
    environment. Assurance is achieved through understanding, but some
    evidence that supports the assurance argument can be difficult to
    understand. This paper describes a novel applica- tion of a technique,
    called literate programming [ll], that significantly improves the
    readability of the assur- ance argument while maintaining its
    consistency with formal specifications that are input to specification
    and verification systems. We describe an application c,f this
    technique to a simple example and discuss the lessons learned from
    this effort.",
  paper = "Moor96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Morg98,
  author = "Morgan, Carroll",
  title = {{Programming from Specifcations, 2nd Ed.}},
  publisher = "Prentice Hall",
  year = "1998",
  link =
"\url{http://www.cse.unsw.edu.au/~carrollm/ProgrammingFromSpecifications.pdf}",
  paper = "Morg98.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Morr95,
  author = "Morrisett, Greg",
  title = {{Compiling with Types}},
  school = "Carnegie Mellon University",
  year = "1995",
  comment = "CMU-CS-95-226",
  link = "\url{https://www.cs.cmu.edu/~rwh/theses/morrisett.pdf}",
  abstract =
    "Compilers for monomorphic languages, suc h as C and Pascal, take
    advantage of types to determine data representations, alignment,
    calling conventions, and register selection.  However, these languages
    lack important features including polymorphism, abstract datatypes,
    and garbage collection.  In contrast, modern programming languages
    such as Standard ML (SML), provide all of these features, but existing
    implementations fail to take full advantage of types.  The result is
    that performance of SML code is quite bad when compared to C.

    In this thesis, I provide a general framework, called
    {\sl type-directed compilation}, that allows compiler writeres to
    take advantage of types at all stages in compilation. In the
    framework, types are used not only to determine efficient
    representations and calling conventions, but also to prove the
    correctness of the compiler. A key property of type-directed
    compilation is that all but the lowest levels of the compiler use
    {\sl typed intermediate languages}. An advantage of this approach
    is that it provides a means for automatically checking the
    integrity of the resulting code.

    An mportant contribution of this work is the development of a new,
    statically-typed intermediate language, called $\lambda^{ML}_i$.
    This language supports {\sl dynamic type dispatch}, providing a
    means to select operations based on types at run time. I show how
    to use dynamic type dispatch to support polymorphism, ad-hoc
    operators, and garbage collection without having to box or tag
    values. This allows compilers for SML to take advantage of
    techniques used in C compilers, without sacrificing language
    features or separate compilation.
```

```
    TO demonstrate the applicability of my approach, I, along with
    others, have constructed a new compiler for SML called TIL that
    eliminates most restrictions on the representations of values. The
    code produced by TIL is roughly twice as fast as code produced by
    the SML/NJ compiler. This is due to at least partially to the use
    of natural representations, but primarily to the conventional
    optimizer which manipulates typed, $\lambda^{ML}_i$ code. TIL
    demonstrates that combining type-directed compilation with dynamic
    type dispatch yields a superior architecture for compilers of
    modern languages.",
  paper = "Morr95.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Morr99,
  author = "Morrisett, Greg and Walker, David and Crary, Karl and Glew, Neil",
  title = {{From System F to Typed Assembly Language}},
  booktitle = "Trans. on Progamming Languages and Systems TOPLAS",
  volume = "21",
  year = "1999",
  pages = "527-568",
  abstract =
    "We motivate the design of typed assembly language (TAL) and present a
    type-preserving ttranslation from Systemn F to TAL. The typed assembly
    language we pressent is based on a conventional RISC assembly
    language, but its static type sytem provides support for enforcing
    high-level language abstratctions, such as closures, tuples, and
    user-defined abstract data types. The type system ensures that
    well-typed programs cannot violatet these abstractionsl In addition,
    the typing constructs admit many low-level compiler optimiztaions. Our
    translation to TAL is specified as a sequence of type-preserving
    transformations, including CPS and closure conversion phases;
    type-correct source programs are mapped to type-correct assembly
    language. A key contribution is an approach to polymorphic closure
    conversion that is considerably simpler than previous work. The
    compiler and typed assembly lanugage provide a fully automatic way to
    produce certified code, suitable for use in systems where unstrusted
    and potentially malicious code must be checked for safety before
    execution.",
  paper = "Morr99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Moss84,
  author = "Mosses, Peter",
```

```
  title = {{A Basic Abstract Semantics Algebra}},
  journal = "LNCS",
  volume = "173",
  year = "1984",
  abstract =
    "It seems that there are some pragmatic advantages in using Abstract
    Semantic Algebras (ASAs) instead of X-notation in denotational
    semantics. The values of ASAs correspond to 'actions' (or
    'processes'), and the operators correspond to primitive ways of
    combining actions. There are simple ASAs for the various independent
    'facets' of actions: a functional ASA for data-flow, an imperative ASA
    for assignments, a declarative ASA for bindings, etc. The aim is to
    obtain general ASAs by systematic combination of these simple ASAs.

    Here we specify a basic ASA that captures the common features of the
    functional, imperative and declarative ASAs -- and highlights their
    differences. We discuss the correctness of ASA specifications, and
    sketch the proof of the consistency and (limiting) completeness of the
    functional ASA, relative to a simple model.

    Some familiarity with denotational semantics and algebraic
    specifications is assumed.",
  paper = "Moss84.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Murt09,
  author = "Murthy, S.G.K and Sekharam, K. Raja",
  title = {{Software Reliability through Theorem Proving}},
  journal = "Defence Science Journal",
  volume = "59",
  number = "3",
  year = "2009",
  pages = "314-317",
  abstract =
    "Improving software reliability of mission-critical systems is
    widely recognised as one of the major challenges.  Early detection
    of errors in software requirements, designs and implementation,
    need rigorous verification and validation techniques. Several
    techniques comprising static and dynamic testing approaches are
    used to improve reliability of mission critical software; however
    it is hard to balance development time and budget with software
    reliability. Particularly using dynamic testing techniques, it is
    hard to ensure software reliability, as exhaustive testing is not
    possible. On the other hand, formal verification techniques
    utilise mathematical logic to prove correctness of the software
    based on given specifications, which in turn improves the
    reliability of the software.  Theorem proving is a powerful formal
    verification technique that enhances the software reliability for
    mission- critical aerospace applications. This paper discusses the
```

```
    issues related to software reliability and theorem proving used to
    enhance software reliability through formal verification
    technique, based on the experiences with STeP tool, using the
    conventional and internationally accepted methodologies, models,
    theorem proving techniques available in the tool without proposing
    a new model.",
  paper = "Murt09.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Myre14,
  author = "Myreen, Magnus O. and Davis, Jared",
  title = {{The Reflective Milawa Theorem Prover is Sound}},
  journal = "LNAI",
  pages = "421-436",
  year = "2014",
  abstract =
    "Milawa is a theorem prover styled after ACL2 but with a small kernel
    and a powerful reflection mechanism. We have used the HOL4 theorem
    prover to formalize the logic of Milawa, prove the logic sound, and
    prove that the source code for the Milawa kernel (2,000 lines of Lisp)
    is faithful to the logic. Going further, we have combined these
    results with our previous verification of an x86 machine-code
    implementation of a Lisp runtime. Our top-level HOL4 theorem states
    that when Milawa is run on top of our verified Lisp, it will only
    print theorem statements that are semantically true. We believe that
    this top-level theorem is the most comprehensive formal evidence of a
    theorem provers soundness to date.",
  paper = "Myre14.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Myre09,
  author = "Myreen, Magnus O. and Gordon, Michael J.C.",
  title = {{Verified LISP Implementations on ARM, x86 and PowerPC}},
  journal = "LNCS",
  volume = "5674",
  pages = "359-374",
  year = "2009",
  abstract =
    "This paper reports on a case study, which we believe is the first
    to produce a formally verified end-to-end implementation of a
    functional programming language running on commercial
    processors. Interpreters for the core of McCarthys LISP 1.5
```

```
    were implemented in ARM, x86 and PowerPC machine code, and proved
    to correctly parse, evaluate and print LISP s-expressions. The
    proof of evaluation required working on top of verified
    implementations of memory allocation and garbage collection. All
    proofs are mechanised in the HOL4 theorem prover.",
  paper = "Myre09.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Myre09a,
  author = "Myreen, Magnus O. and Slind, Konrad and Gordon, Michael J.C.",
  title = {{Extensible Proof-Producing Compilation}},
  journal = "LNCS",
  volume = "5501",
  pages = "2-16",
  year = "2009",
  abstract =
    "This paper presents a compiler which produces machine code from
    functions defined in the logic of a theorem prover, and at the same
    time proves that the generated code executes the source functions.
    Unlike previously published work on proof-producing compilation from a
    theorem prover, our compiler provides broad support for user-defined
    extensions, targets multiple carefully modelled commercial machine
    languages, and does not require termination proofs for input
    functions. As a case study, the compiler is used to construct verified
    interpreters for a small LISP-like language. The compiler has been
    implemented in the HOL4 theorem prover.",
  paper = "Myre09a.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Myre10,
  author = "Myreen, Magnus O.",
  title = {{Verified Just-In-Time Compiler on x86}},
  journal = "ACM SIGLAN Notices - POPL'10",
  volume = "45",
  number = "1",
  year = "2010",
  pages = "107-118",
  abstract =
    "This paper presents a method for creating formally correct just-in-
    time (JIT) compilers. The tractability of our approach is demonstrated
    through, what we believe is the first, verification of a JIT
    compiler with respect to a realistic semantics of self-modifying x86
```

```
    machine code. Our semantics includes a model of the instruction
    cache. Two versions of the verified JIT compiler are presented: one
    generates all of the machine code at once, the other one is incremental
    i.e. produces code on-demand. All proofs have been performed
    inside the HOL4 theorem prover.",
  paper = "Myre10.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Myre11,
  author = "Myreen, Magnus O. and Davis, Jared",
  title = {{A Verified Runtime for a Verified Theorem Prover}},
  journal = "NCS",
  volume = "6898",
  pages = "265-280",
  year = "2011",
  link = "\url{https://www.cl.cam.ac.uk/~mom22/jitawa/}",
  abstract =
    "Theorem provers, such as ACL2, HOL, Isabelle and Coq, rely on the
    correctness of runtime systems for programming languages like ML,
    OCaml or Common Lisp. These runtime systems are complex and critical
    to the integrity of the theorem provers.

    In this paper, we present a new Lisp runtime which has been formally
    verified and can run the Milawa theorem prover. Our runtime consists
    of 7,500 lines of machine code and is able to complete a 4 gigabyte
    Milawa proof effort. When our runtime is used to carry out Milawa
    proofs, less unverified code must be trusted than with any other
    theorem prover.

    Our runtime includes a just-in-time compiler, a copying garbage collector,
    a parser and a printer, all of which are HOL4-verified down to
    the concrete x86 code. We make heavy use of our previously developed
    tools for machine-code verification. This work demonstrates that our
    approach to machine-code verification scales to non-trivial
    applications.",
  paper = "Myre11.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Myre12,
  author = "Myreen, Magnus O.",
  title = {{Functional Programs: Conversions between Deep and Shallow
            Embeddings}},
```

```
  journal = "LNCS",
  volume = "7406",
  pages = "412-417",
  year = "2012",
  link = "\url{https://www.cl.cam.ac.uk/~mom22/jitawa/}",
  abstract =
    "This paper presents a method which simplifies verification of deeply
    embedded functional programs. We present a technique by which
    proof-certified equations describing the effect of functional
    programs (shallow embeddings) can be automatically extracted from their
    operational semantics. Our method can be used in reverse, i.e. from
    shallow to deep embeddings, and thus for implementing certifying code
    synthesis: we have implemented a tool which maps HOL functions to
    equivalent Lisp functions, for which we have a verified Lisp runtime.
    A key benefit, in both directions, is that the verifier does not need
    to understand the operational semantics that gives meanings to the
    deep embeddings.",
  paper = "Myre12.pdf",
  keywords = "printed, DONE"
}
```

———-

1.17.14 N

— **axiom.bib** —

```
@book{Nede14,
  author = "Nederpelt, Rob and Geuvers, Herman",
  title = {{Type Theory and Formal Proof}},
  year = "2014",
  publisher = "Cambridge University Press",
  isbn = "978-1-107-03650-5",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Niss99,
  author = "Nissanke, Nimal",
  title = {{Formal Specification}},
  publisher = "Springer",
  year = "1999",
  isbn = "978-1-85233-002-6",
  paper = "Niss99.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Nobl16,
  author = "Noble, James and Black, Andrew P. and Bruce, Kim B. and
            Homer, Michael and Miller, Mark S.",
  title = {{The Left Hand of Equals}},
  booktitle = "Int. Symp. of New Ideas, New Paradigms, and Reflections
               on Programming and Software",
  publisher = "ACM",
  pages = "224-237",
  year = "2016",
  abstract =
    "When is one object equal to another object? While object identity is
    fundamental to object-oriented systems, object equality, although
    tightly intertwined with identity, is harder to pin down. The
    distinction between identity and equality is reflected in
    object-oriented languages, almost all of which provide two variants of
    'equality', while some provide many more. Programmers can usually
    override at least one of these forms of equality, and can always
    define their own methods to distinguish their own objects.

    This essay takes a reflexive journey through fifty years of identity
    and equality in object-oriented languages, and ends somewhere we did
    not expect: a 'left-handed' equality relying on trust and grace.",
  paper = "Nobl16.pdf",
  keywords = "printed, DONE"
}
```

———-

1.17.15 O

— **axiom.bib** —

```
@article{Odon81,
  author = "O'Donnell, Michael J.",
  title = {{A Critique of the Foundations of Hoare-style Programming Logics}},
  journal = "LNCS",
  volume = "131",
  pages = "349-374",
  year = "1981",
  abstract =
    "Much recent discussion in computing journals has been devoted to
    arguments about the feasibility and usefulness of formal
    verification methods for increasing confidence in computer
    programs. Too little attention has been given to precise criticism
    of specific proposed systems for reasoning about programs. Whether
    such systems are to be used for formal verification, by hand or
    automatically, or as a rigorous foundation for imformal reasoning,
    it is essential that they be logically sound. Several popular
    rules in the Hoare language are in fact not sound. These rules
    have been accepted because they have not been subjected to
```

```
    sufficiently strong standards of correctness. This paper attempts
    to clarify the different technical definitions of correctness of a
    logic, to show that only the strongest of these definitions is
    acceptable for Hoare logic, and to correct some of the unsound
    rules which have appeared in the literature. The corrected rules
    are given merely to show that it is possible to do so. Convenient
    and elegant rules for reasoning about certain programming
    constructs will probably require a more flexible notation than
    Hoare's.",
  paper = "Odon81.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Okas96,
  author = "Okasaki, Chris",
  title = {{Purely Functional Data Structures}},
  school = "Carnegie Mellon University",
  year = "1996",
  link = "\url{}",
  comment = "CMU-CS-96-177",
  isbn = "978-0521663502",
  abstract =
    "When a C programmer needs an efficient data structure for a
    particular problem, he or she can often simply look one up in any of
    a number of good textbooks or handbooks.  Unfortunately, programmers
    in functional languages such as Standard ML or Haskell do not have
    this luxury.  Although some data structures designed for imperative
    languages such as C can be quite easily adapted to a functional
    setting, most cannot, usually because they depend in crucial ways on
    assignments, which are disallowed, or at least discouraged, in
    functional languages.  To address this imbalance, we describe several
    techniques for designing functional data structures, and numerous
    original data structures based on these techniques, including multiple
    variations of lists, queues, double-ended queues, and heaps, many
    supporting more exotic features such as random access or efficient
    catenation.

    In addition, we expose the fundamental role of lazy evaluation in
    amortized functional data structures.  Traditional methods of
    amortization break down when old versions of a data structure, not
    just the most recent, are available for further processing.  This
    property is known as persistence , and is taken for granted in
    functional languages.  On the surface, persistence and amortization
    appear to be incompatible, but we show how lazy evaluation can be used
    to resolve this conflict, yielding amortized data structures that are
    efficient even when used persistently.  Turning this relationship
    between lazy evaluation and amortization around, the notion of
    amortization also provides the first practical techniques for
    analyzing the time requirements of non-trivial lazy programs.
```

```
    Finally, our data structures offer numerous hints to programming
    language designers, illustrating the utility of combining strict and
    lazy evaluation in a single language, and providing non-trivial
    examples using polymorphic recursion and higher-order, recursive
    modules.",
  paper = "Okas96.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Oles82,
  author = "Oles, Frank Joseph",
  title = {{A Category-Theoretic Approach to the Semantics of
            Programming Languages}},
  school = "Syracuse University",
  year = "1982",
  abstract =
    "Here we create a framework for handling the semantics of fully
    typed programming languages with imperative features, higher-order
    ALGOL-like procedures, block structure, and implicit
    conversions. Our approach involves introduction of a new family of
    programming languages, the {\sl coercive typed $\lambda$-calculi},
    denoted by $L$ in the body of the dissertation. By appropriately
    choosing the linguistic constants (i.e. generators) of $L$, we can
    view phrases of variants of ALGOL as syntactically sugared phrases
    of $L$.

    This dissertation breaks into three parts. In the first part,
    consisting of the first chapter, we supply basic definitions and
    motivate the idea that functor categories arise naturally in the
    explanation of block structure and stack discipline. The second
    part, consisting of the next three chapters, is dedicated to the
    general theory of the semantics of the coercive typed
    $\lambda$-calculus; the interplay between posets, algebras, and
    Cartesian closed categories is particularly intense here. The
    remaining four chapters make up the final part, in which we apply
    the general theory to give both direct and continuation semantics
    for desugared variants of ALGOL. To do so, it is necessary to show
    certain functor categories are Cartesian closed and to describe a
    category $\Sigma$ of store shapes. An interesting novelty in the
    presentation of continuation semantics is the view that commands
    form a procedural, rather than a primitive, phrase type.",
  paper = "Oles82.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Oury08,
  author = "Oury, Nicolas and Swierstra, Wouter",
  title = {{The Power of Pi}},
  link = "\url{https://cs.ru.nl/~wouters/Publications/ThePowerOfPi.pdf}",
  booktitle = "Int. Conf. on Functional Programming",
  year = "2008",
  abstract =
    "This paper exhibits the power of programming with dependent types by
    dint of embedding three domain-specific languages: Cryptol, a
    language for cryptographic protocols; a small data description
    language; and relational algebra. Each example demonstrates
    particular design patterns inherent to dependently-typed programming.
    Documenting these techniques paves the way for further research in
    domain-specific embedded type systems.",
  paper = "Oury08.pdf",
  keywords = "printed"
}
```

———-

1.17.16 P

— **axiom.bib** —

```
@article{Parn72,
  author = "Parnas, David L.",
  title = {{A Technique for Software Module Specification with
            Examples}},
  journal = "CACM",
  volume = "15",
  number = "5",
  year = "1972",
  pages = "330-336",
  abstract =
    "This paper presents an approach to writing specifications for
    parts of software systems. The main goal is to provide
    specifications sufficiently precise and complete that other pieces
    of software can be written to interact with the piece specified
    without additional information. The secondary goal is to include
    in the specification no more information than necessary to meet
    the first goal. The technique is illustrated by means of a variety
    of examples from a tutorial system.",
  paper = "Parn72.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@incollection{Parn10,
```

```
  author = "Parnas, David Lorge",
  title = {{Precise Documentation: The Key to Better Software}},
  booktitle = "The Future of Software Engineering",
  publisher = "Springer",
  year = "2010",
  pages = "125-148",
  abstract =
    "The prime cause of the sorry 'state of the art' in software
    development is our failure to produce good design documentation. Poor
    documentation is the cause of many errors and reduces efficiency in
    every phase of a software product's development and use. Most software
    developers believe that 'documentation' refers to a collection of
    wordy, unstructured, introductory descriptions, thousands of pages
    that nobody wanted to write and nobody trusts. In contrast, Engineers
    in more traditional disciplines think of precise blueprints, circuit
    diagrams, and mathematical specifications of component
    properties. Software developers do not know how to produce precise
    documents for software. Software developments also think that
    documentation is something written after the software has been
    developed. In other fields of Engineering much of the documentation is
    written before and during the development. It represents forethought
    not afterthought. Among the benefits of better documentation would be:
    easier reuse of old designs, better communication about requirements,
    more useful design reviews, easier integration of separately written
    modules, more effective code inspection, more effective testing, and
    more efficient corrections and improvements. This paper explains how
    to produce and use precise software documentation and illustrate the
    methods with several examples.",
  paper = "Parn10.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Paul83,
  author = "Paulson, Lawrence C.",
  title = {{A Higher-Order Implementation of Rewriting}},
  journal = "Science of Computer Programming",
  volume = "3",
  pages = "119-149",
  year = "1983",
  abstract =
    "Many automatic theorem-provers rely on rewriting. Using theorems as
    rewrite rules helps to simplify the subgoals that arise during a
    proof.  LCF is an interactive theorem-prover intended For reasoning
    about computation. Its implementation of rewriting is presented in
    detail. LCF provides a Family of rewriting Functions, and operators to
    combine them. A succession of Functions is described, From pattern
    matching primitives to the rewriting tool that performs most
    inferences in LCF proofs.  The design is highly modular. Each function
    performs a basic, specific task, such as recognizing a certain form of
```

```
    tautology. Each operator implements one method of building a rewriting
    Function From simpler ones. These pieces can be put together in
    numerous ways, yielding a variety of rewriting strategies. The
    approach involves programming with higher-order Functions. Rewriting
    Functions are data values, produced by computation on other rewriting
    Functions. The code is in daily use at Cambridge, demonstrating the
    practical use of Functional programming.",
  paper = "Paul83.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Paul87,
  author = "Paulson, Lawrence C.",
  title = {{Logic and Computation}},
  publisher = "Press Syndicate of Cambridge University",
  year = "1987",
  isbn = "0-521-34632-0",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@incollection{Paul90a,
  author = "Paulson, Lawrence C.",
  title = {{Isabelle: The Next 700 Theorem Provers}},
  booktitle = "Logic and Computer Science",
  publisher = "Academic Press",
  pages = "361-386",
  year = "1990",
  paper = "Paul90a.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Paul96,
  author = "Paulson, L.C.",
  title = {{ML for the Working Programmer 2nd Edition}},
  year = "1996",
  publisher = "Cambridge University Press",
  isbn = "0-521-56543-X",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Peyt93,
  author = "Peyton-Jones, Simon and Wadler, Philip",
  title = {{Imperative Functional Programming}},
  booktitle = "Principles of Programming Languages POPL'93",
  publisher = "ACM",
  year = "1993",
  pages = "71-84",
  abstract =
    "We present a new model, based on monads, for performing
    input/output in a non-strict, purely functional language. It is
    composable, extensible, efficient, requires no extensions to the
    type system, and extends smoothly to incorporate mixed-language
    working and in-place array updates.",
  paper = "Peyt93.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Peyt17,
  author = "Peyton-Jones, Simon",
  title = {{Escape from the ivory tower: the Haskell journey}},
  link = "\url{https://www.youtube.com/watch?v=re96UgMk6GQ}",
  year = "2017"
}
```

———-

— **axiom.bib** —

```
@article{Pfen01,
  author = "Pfenning, Frank and Davies, Rowan",
  title = {{A Judgemental Reconstruction of Modal Logic}},
  journal = "Mathematical Structures in Computer Science",
  volume = "11",
  pages = "511-540",
  year = "2001",
  abstract =
    "We reconsider the foundations of modal logic, following Martin-Lof's
    methodology of distinguishing judgments from propositions. We give
    constructive meaning explanations for necessity and possibility which
    yields a simple and uniform system of natural deduction for
    intuitionistic modal logic which does not exhibit anomalies found in
    other proposals. We also give a new presentation of lax logic and nd
    that the lax modality is already expressible using possibility and
    necessity. Through a computational interpretation of proofs in modal
    logic we further obtain a new formulation of Moggi's monadic
```

```
    metalanguage.",
  paper = "Pfen01.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen04a,
  author = "Pfenning, Frank",
  title = {{Lecture Notes on Bidirectional Type Checking}},
  year = "2004",
  paper = "Pfen04a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Pfen87,
  author = "Pfenning, Frank",
  title = {{Proof Transformations in Higher-Order Logic}},
  year = "1987",
  school = "Carnegie Mellon University",
  link = "\url{http://www-cgi.cs.cmu.edu/~fp/papers/thesis87.pdf}",
  abstract =
    "We investigate the problem of translating between different styles of
    proof systems in higher-order logic: analytic proofs which are well
    suited for automated theorem proving, and non-analytic deductions
    which are well suited for the mathematician. Analytic proofs are
    represented as expansion proofs, $H$ , a form of the sequent calculus we
    define, non-analytic proofs are represented by natural deductions. A
    non-deterministic translation algorithm between expansion proofs and
    $H$-deductions is presented and its correctness is proven. We also
    present an algorithm for translation in the other direction and prove
    its correctness. A cut-elimination algorithm for expansion proofs is
    given and its partial correctness is proven. Strong termination of
    this algorithm remains a conjecture for the full higher-order
    system, but is proven for the first-order fragment. We extend the
    translations to a non-analytic proof system which contains a primitive
    notion of equality, while leaving the notion of expansion proof
    unaltered. This is possible, since a non-extensional equality is
    definable in our system of type theory. Next we extend analytic and
    non-analytic proof systems and the translations between them to
    include extensionality.  Finally, we show how the methods and notions
    used so far apply to the problem of translating expansion proofs into
    natural deductions. Much care is taken to specify this translation in
    a modular way (through tactics) which leaves a large number of choices
    and therefore a lot of room for heuristics to produce elegant natural
    deductions. A practically very useful extension, called symmetric
```

```
    simplification, produces natural deductions which make use of lemmas
    and are often much more intuitive than the normal deductions which
    would be created by earlier algorithms.",
  paper = "Pfen87.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pfen88a,
  author = "Pfenning, Frank and Elliott, Conal",
  title = {{Higher-Order Abstract Syntax}},
  booktitle = "Symp on Language Design and Implementation PLDI'88",
  publisher = "ACM",
  link = "\url{https://www.cs.cmu.edu/~fp/papers/pldi88.pdf}",
  year = "1988",
  abstract =
    "We describe motivation, design, use, and implementation of
    higher-order abstract syntax as a central representation for
    programs, formulas, rules, and other syntactic objects in program
    manipulation and other formal systems where matching and substitution
    or unification are central operations.  Higher-order abstract syntax
    incorporates name binding information in a uniform and language
    generic way.  Thus it acts as a powerful link integrating diverse
    tools in such formal environments.  We have implemented higher-order
    abstract syntax, a supporting matching and unification algorithm, and
    some clients in Common Lisp in the framework of the Ergo project at
    Carnegie Mellon University.",
  paper = "Pfen88a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Phel92,
  author = "Phelps, Tom",
  title = {{A Common Lisp CGOL}},
  year = "1992",
  link = "\url{https://people.eecs.berkeley.edu/~fateman/cgol/cgol.1/cgol.ps}",
  abstract =
    "CGOL is an Algol-like notation for Lisp. The original version,
    written by Vaughan Pratt at M.I.T. is the early 70s was written in
    Maclisp. This new version is based on Common Lisp. CGOL was translated
    to Common Lisp in the following four-stage process:

    (1) cgol.tok, the tokenizer has been almost completely rewritten; (2)
    cgoll.l, the main translation loop with library of translation schemas
    has been converted from Maclisp to Common Lisp; (3) the code that
    cgol1.l produces has been converted to Common Lisp; (4) selected
```

```
    examples of CGOL programs themselves were rewritten, since certain
    aspects of the semantics of Maclisp would otherwise not be modelled in
    the expected fashion. Maclisp differs from Common Lisp in a variety of
    respects, and some of them are apparent from CGOL including direct
    escapes to Lisp, variable scoping, function definitions and numerous
    other aspects.

    In contrast to the programming described above, the major contribution
    of this paper is annotation of selected code from the CGOL
    translator.",
  paper = "Phil92.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Pier91b,
  author = "Pierce, Benjamin C.",
  title = {{Basic Category Theory for Computer Scientists}},
  publisher = "MIT Press",
  year = "1991",
  isbn = "0-262-66071-7"
}
```

———-

— **axiom.bib** —

```
@book{Pier02,
  author = "Pierce, Benjamin C.",
  title = {{Types and Programming Languages}},
  publisher = "MIT Press",
  year = "2002",
  isbn = "978-0-262-16209-8",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Piro08,
  author = "Piroi, Florina and Buchberger, Bruno and Rosenkranz, Camelia",
  title = {{Mathematical Journals as Reasoning Agents: Literature Review}},
  year = "2008",
  link = "\url{http://www.risc.jku.at/publications/download/risc_3442/Math-Agents-for-SFB-2008-03-10-12h00.pdf}"
  abstract =
    "This report reviews the literature relevant for the research project
    'MathAgents: Mathematical Journals as Reasoning Agents' proposed by
```

```
    Bruno Buchberger as a technology transfer project based on the results
    of the SFB Project 'Scientific Computing', in particular the project
    SFB 1302, 'Theorema'. The project aims at computersupporting the
    refereeing process of mathematical journals by tools that are mainly
    based on automated reasoning and also at building up the knowledge
    archived in mathematical journals in such a way that they can act as
    interactive and active reasoning agents later on.  In this report,
    we review current mathematical software systems with a focus on the
    availability of tools that can contribute to the goals of the Math
    Agents project.",
  paper = "Piro08.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Pitt18,
  author = "Pittman, Dan",
  title = {{Proof Theory Impressionism: Blurring the Curry-Howard Line}},
  year = "2018",
  comment = "StrangeLoop 2018",
  link = "\url{https://www.youtube.com/watch?v=jrVPB-Ad5Gc}"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Poll94,
  author = "Pollack, Robert",
  title = {{The Theory of LEGO - A Proof Checker for the Extended Calculus
            of Constructions}},
  school = "University of Edinburgh",
  link =
     "\url{http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.2610}",
  year = "1994",
  abstract =
    "LEGO is a computer program for interactive typechecking in the
    Extended Calculus of Constructions and two of its subsystems. LEGO
    also supports the extension of these three systems with inductive
    types. These type systems can be viewed as logics, and as meta
    languages for expressing logics, and LEGO is intended to be used for
    interactively constructing proofs in mathematical theories presented
    in these logics. I have developed LEGO over six years, starting from
    an implementation of the Calculus of Constructions by Gerard
    Huet. LEGO has been used for problems at the limits of our abilities
    to do formal mathematics.

    In this thesis I explain some aspects of the meta-theory of LEGOs
    type systems leading to a machine-checked proof that typechecking is
```

```
    decidable for all three type theories supported by LEGO, and to a
    verified algorithm for deciding their typing judgements, assuming only
    that they are normalizing. In order to do this, the theory of Pure
    Type Systems (PTS) is extended and formalized in LEGO. This extended
    example of a formally developed body of mathematics is described, both
    for its main theorems, and as a case study in formal mathematics. In
    many examples, I compare formal definitions and theorems with their
    informal counterparts, and with various alternative approaches, to
    study the meaning and use of mathematical language, and suggest
    clarifications in the informal usage.

    Having outlined a formal development far too large to be surveyed in
    detail by a human reader, I close with some thoughts on how the human
    mathematicians state of understanding and belief might be affected by
    posessing such a thing.",
  paper = "Poll94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Poll94a,
  author = "Pollack, Robert",
  title = {{On Extensibility of Proof Checkers}},
  journal = "LNCS",
  volume = "996",
  pages = "140-161",
  year = "1994",
  abstract =
    "My suggestion is little different from LCF, just replacing one
    computational meta language (ML) with another (ECC, FSO,...). The
    philosophical point is that it is then possible to accept non-
    canonical proof notations as object level proofs, removing the need to
    actually normalize them. There are problems to be worked out in
    practice, such as extraction of programs from constructive proof, and
    efficient execution of pure, total programs. Although this approach
    doesn't address the difficulty of proving correctness of tactics in
    the meta level, it is immediatly useful for tactics with structural
    justification (e.g. weakening) which are not even representable in
    LCF, and are infeasible in the Nuprl variant of LCF. Since it can be
    used for any object system without adding new principles such as
    reflection, and is compatible with other approaches to extensibility
    (especially partial reflection), it should be considered as part of
    the answer to extensibility in proof checkers.",
  paper = "Poll94a.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Pott98,
  author = "Pottier, Francois",
  title = {{Type Inference in the Presence of Subtyping: From Theory
            to Practice}},
  school = "Universite Paris 7",
  year = "1988",
  comment = "INRIA Research Report RR-3483",
  abstract =
    "From a purely theoretical point of view, type inference for a
    functional language with parametric polymorphism and subtyping
    poses little difficulty. Indeed, it suffices to generalize the
    inference algorithm used in the ML language, so as to deal with
    type inequalities, rather than equalities. However, the number of
    such inequalities is linear in the program size -- whence, from a
    practical point of view, a serious efficiency and readability
    problem.

    To solve this problem, one must simplify the inferred
    constraints. So, after studying the logical properties of
    subtyping constraints, this work proposes several simplifcation
    algorithms. They combine seamlessly, yielding a homogeneous, fully
    formal framework, which directly leads to an efficient
    implementation. Although this theoretical study is performed in a
    simplified setting, numerous extensions are possible. Thus, this
    framework is realistic and should allow a practical appearance of
    subtyping in languages with type inference.",
  paper = "Pott98.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Popo03,
  author = "Popov, Nikolaj",
  title = {{Verification Using Weakest Precondition Strategy}},
  comment = "Talk at Comp. Aided Verfication of Information Systems",
  year = "2003",
  location = "Timisoara, Romania",
  abstract =
    "We describe the weakest precondition strategy for verifying
     programs.  This is a method which takes a specification and an
     annotated program and generates so-called verification conditions:
     mathematical lemmata which have to be proved in order to obtain a
     formal correctness proof for the program with respect to its
     specification.  There are rules for generating the intermediate pre
     and post conditions algorithmically.  Based on these rules, we have
     developed a package for generating verification conditions.",
  paper = "Popo03.pdf",
 keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Prat79,
  author = "Pratt, Vaughan",
  title = {{A Mathematician's View of Lisp}},
  journal = "Byte Magazine",
  year = "1979",
  pages = "162-169",
  paper = "Byte79.pdf"
}
```

———-

1.17.17 Q

1.17.18 R

— **axiom.bib** —

```
@misc{Remy17,
  author = "Remy, Didier",
  title = {{Type Systems for Programming Languages}},
  ywar = "2017",
  link = "\url{http://gallium.inria.fr/~remy/mpri/cours1.pdf}",
  paper = "Remy17.tgz"
}
```

———-

— **axiom.bib** —

```
@article{Reyn85,
  author = "Reynolds, J.C.",
  title = {{Three Approaches to Type Structure}},
  journal = "LNCS",
  volume = "185",
  year = "1985",
  abstract =
    "We examine three disparate views of the type structure of
    programming languages: Milner's type deduction system and polymorphic
    let construct, the theory of subtypes and generic operators, and
    the polymorphic or second-order typed lambda calculus.  These
    approaches are illustrated with a functional language including
    product, sum and list constructors.  The syntactic behavior of types
    is formalized with type inference rules, bus their semantics is
    treated intuitively.",
  paper = "Reyn85.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Reyn93,
  author = "Reynolds, John C.",
  title = {{The Discoveries of Continuations}},
  journal = "Lisp and Symbolic Computation",
  volume = "6",
  pages = "233-248",
  year = "1993",
  abstract =
    "We give a brief account of the discoveries of continuations and
    related concepts.",
  paper = "Reyn93.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Robi01,
  author = "Robinson, Alan and Voronkov, Andrei",
  title = {{Handbook of Automated Reasoning (2 Volumes)}},
  year = "2001",
  publisher = "MIT Press",
  isbn = "0-262-18223-8",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Roch11,
  author = "Roche, Daniel S.",
  title = {{Chunky and Equal-Spaced Polynomial Multiplication}},
  journal = "J. Symbolic Computation",
  volume = "46",
  pages = "791-806",
  year = "2011",
  abstract =
    "Finding the product of two polynomials is an essential and basic
    problem in computer algebra. While most previous results have
    focused on the worst-case complexity, we instead employ the
    technique of adaptive analysis to give an improvement in many
    easy cases. We present two adaptive measures and methods
    for polynomial multiplication, and also show how to effectively
    combine them to gain both advantages. One useful feature of these
    algorithms is that they essentially provide a gradient between
    existing sparse and dense methods. We prove that these
```

```
    approaches provide significant improvements in many cases but
    in the worst case are still comparable to the fastest existing
    algorithms.",
  paper = "Roch11.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Roor00,
  author = "Roorda, Jan-Willem",
  title = {{Pure Type Systems for Functional Programming}},
  year = "2000",
  school = "University of Utrecht",
  abstract =
    "We present a functional programming language based on Pure Type
    Systems (PTSs).  We show how we can define such a language by
    extending the PTS framework with algebraic data types, case
    expressions and definitions. To be able to experiment with our
    language we present an implementation of a type checker and an
    interpreter for our language.  PTSs are well suited as a basis for a
    functional programming language because they are at the top of a
    hierarchy of increasingly stronger type systems. The concepts of
    'existential types', 'rank-n polymorphism' and 'dependent types' arise
    naturally in functional programming languages based on the systems in
    this hierarchy. There is no need for ad-hoc extensions to incorporate
    these features.  The type system of our language is more powerful than
    the Hindley-Milner system.  We illustrate this fact by giving a number
    of meaningful programs that cannot be typed in Haskell but are typable
    in our language. A 'real world' example of such a program is the
    mapping of a specialisation of a Generic Haskell function to a Haskell
    function.  Unlike the description of the Henk language by Simon Peyton
    Jones and Erik Meijer we give a complete formal definition of the type
    system and the operational semantics of our language. Another
    difference between Henk and our language is that our language is
    defined for a large class of Pure Type Systems instead of only for the
    systems of the $\lambda$-cube.",
  paper = "Roor00.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Rose07,
  author = "Rosenkranz, Markus",
  title = {{Gr\"obner Bases in Symbolic Analysis}},
  publisher = "Walter de Gruyter, Berlin Germany",
  year = "2007",
  isbn = "978-3-11.019323-7",
```

```
  keywords = "axiomref, TPDref"
}
```

———-

— **axiom.bib** —

```
@book{Royx03,
  author = "Roy, Peter Van and Haridi, Seif",
  title = {{Concepts, Techniques, and Models of Computer Programming}},
  year = "2003",
  link =
  "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.7366&rep=rep1&type=pdf}",
  publisher = "MIT",
  isbn = "978-0262220699",
  paper = "Royx03.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Russ77,
  author = "Russell, Bruce D.",
  title = "Implementation Correctness Involving a Language with GOTO
           statements",
  journal = "SIAM Journal of Computing",
  volume = "6",
  number = "3",
  year = "1977",
  abstract =
    "Two languages, one a simple structured programming language, the
    other a simple goto language, are defined. A denotational semantics is
    given for each language. An interpreter for the goto language is given
    and is proved correct with respect to the denotational semantics. A
    compiler from the structured to the goto language is defined and
    proved to be a semantically invariant translation of programs. The
    proofs are by computational induction.",
  paper = "Russ77.pdf",
  keywords = "printed"
}
```

———-

1.17.19 S

— **axiom.bib** —

```
@techreport{Sabr92,
  author = "Sabry, Amr and Felleisen, Matthias",
```

```
  title = {{Reasoning About Programs in Continuation-Passing Style}},
  type = "technical report",
  number = "TR 92-180",
  institution = "Rice University",
  year = "1992",
  abstract =
    "Plotkin's $\lambda$-value calculus is sound but incomplete for
    reasoning about $\beta\nu$-transformations on programs in continuation-
    passing style (CPS). To find a complete extension, we define a new,
    compactifying CPS transformation and an 'inverse' mapping,
    $un$-CPS, both of which are interesting in their own right. Using the
    new CPS transformation, we can determine the precise language of CPS
    terms closed under $\beta\nu$ transformations. Using the $un$-CPS
    transformation, we can derive a set of axioms such that every equation
    between source programs is provable if and only if $\beta\nu$ can
    prove the corresponding equation between CPS programs. The extended
    calculus is equivalent to an untyped variant of Moggi's computational
    $\lambda$-calculus.",
  paper = "Sabr92.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@article{Sarm17,
  author = "Sarma, Gopal and Hay, Nick J.",
  title = {{Robust Computer Algebra, Theorem Proving, and Oracle AI}},
  journal = "Informatica",
  volume = "41",
  number = "3",
  link = "\url{https://arxiv.org/pdf/1708.02553.pdf}",
  year = "2017",
  abstract =
    "In the context of superintelligent AI systems, the term 'oracle' has
    two meanings.  One refers to modular systems queried for
    domain-specific tasks.  Another usage, referring to a class of systems
    which may be useful for addressing the value alignment and AI control
    problems, is a superintelligent AI system that only answers questions.
    The aim of this manuscript is to survey contemporary research problems
    related to oracles which align with long-term research goals of AI
    safety.  We examine existing question answering systems and argue that
    their high degree of architectural heterogeneity makes them poor
    candidates for rigorous analysis as oracles.  On the other hand, we
    identify computer algebra systems (CASs) as being primitive examples
    of domain-specific oracles for mathematics and argue that efforts to
    integrate computer algebra systems with theorem provers, systems which
    have largely been developed independent of one another, provide a
    concrete set of problems related to the notion of provable safety that
    has emerged in the AI safety community.  We review approaches to
    interfacing CASs with theorem provers, describe well-defined
    architectural deficiencies that have been identified with CASs, and
```

```
    suggest possible lines of research and practical software projects for
    scientists interested in AI safety.",
  paper = "Sarm17.pdf",
  keywords = "printed, axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Schm86,
  author = "Schmitt, P.H.",
  title = {{Computational Aspects of Three-Valued Logic}},
  journal = "LNCS",
  volume = "230",
  year = "1986",
  abstract =
    "This paper investigates a three-valued logic $L_3$, that has been
    introduced in the study of natural language semantics. A complete
    proof system based on a three-valued analogon of negative resolution
    is presented.  A subclass of $L_3$ corresponding to Horn clauses in
    two-valued logic is defined. Its model theoretic properties are
    studied and it is shown to admit a PROLOG-style proof procedure.",
  paper = "Schm86.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Scho88,
  author = "Schonhage, A.",
  title = {{Probabilistic Computation of Integer Polynomial GCDs}},
  journal = "J. of Algorithms",
  volume = "9",
  pages = "365-371",
  year = "1988",
  abstract =
    "We describe a probabilistic algorithm for the computation of the
    gcd of two univariate integer polynomials of degrees $\le n$ with
    their $l^1$-norms being bounded by $2^h$ and estimate its expected
    running time by a worst-case bound of
    $O(n(n+h)^{l+\epsilon(l)})$.",
  paper = "Scho88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Scot93,
  author = "Scott, Dana S.",
  title = {{A Type-Theoretical Alternative to ISWIM, CUCH, OWHY}},
  journal = "Theoretical Computer Science",
  volume = 121,
  number = "1-2",
  year = "1993",
  pages = "411-440",
  abstract =
    "The paper (first written in 1969 and circulated privately) concerns
    the definition, axiomatization, and applications of the hereditarily
    monotone and continuous functionals generated from the integers and
    the Booleans (plus undefined elements). The system is formulated as
    a typed system of combinators (or as a typed -calculus) with a
    recursion operator (the least fixed-point operator), and its proof
    rules are contrasted to a certain extent with those of the untyped
    -calculus. For publication (1993), a new preface has been added, and
    many bibliographical references and comments in footnotes have been
    appended.",
  paper = "Scot93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Scot71,
  author = "Scott, Dana S. and Strachey, C.",
  title = {{Towards a Mathematical Semantics for Computer Languages}},
  journal = "Proc. Symp. on Computers and Automata",
  volume = "21",
  year = "1971",
  abstract =
    "Compilers for high-level languages are generally constructed to
    give a complete translation of the programs into machine
    lanugage. As machines merely juggle bit patterns, the concepts of
    the original language may be lost or at least obscured during this
    passage. The purpose of a mathematical semantics is to give a
    correct and meaningful correspondence between programs and
    mathematical entities in a way that is entirely independent of an
    implementation. This plan is illustrated in a very elementary
    method with the usual idea of state transformations. The next
    section shows why the mathematics of functions has to be modified
    to accommodate recursive commands. Section 3 explains the
    modification. Section 4 introduces the environments for handling
    variables and identifiers and shows how the semantical equations
    define equivalence of programs. Section 5 gives an exposition of
    the new type of mathematical function spaces that are required for
    the semantics of procedures when these are allowed in assignment
    statements. The conclusion traces some of the background of the
    project and points the way to future work.",
  paper = "Scot71.pdf",
```

```
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Sedj13,
  author = "Sedjelmaci, Sidi M.",
  title = {{Fast Parallel GCD Algorithm of Many Integers}},
  journal = "ACM Comm in Computer Algebra",
  volume = "47",
  number = "3",
  year = "2013",
  abstract =
    "We present a new parallel algorithm which computes the GCD of $n$
    integers of $O(n)$ bits in $O(n/log~n)$ time with
    $O(n^{2+\epsilon})$ processors, for any $\epsilon > 0$ on CRCW
    PRAM model.",
  paper = "Sedj13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Shie95,
  author = "Shieber, Stuart M. and Schabes, Yves and Pereira, Fernando C.N.",
  title = {{Principles and Implementation of Deductive Parsing}},
  journal = "J. Logic Programming",
  volume = "24",
  number = "1-2",
  pages = "3-36",
  year = "1995",
  abstract =
    "We present a system for generating parsers based directly on the
    metaphor of parsing as deduction.  Parsing algorithms can be
    represented directly as deduction systems, and a single deduction
    engine can interpret such deduction systems so as to implement the
    corresponding parser.  The method generalizes easily to parsers for
    augmented phrase structure formalisms, such as definite-clause
    grammars and other logic grammar formalisms, and has been used for
    rapid prototyping of parsing algorithms for a variety of formalisms
    including variants of tree-adjoining grammars, categorial grammars,
    and lexicalized context-free grammars.",
  paper = "Shie95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Shan94,
  author = "Shankar, N.",
  title = {{Metamathematics, Machines, and Godel's Proof}},
  publisher = "Cambridge University Press",
  year = "1994",
  isbn = "0-521-58533-3",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Shie02,
  author = "Shields, Mark and Jones, Simon Peyton",
  title = {{First-Class Modules for Haskell}},
  booktitle = "9th Int. Conf. on Foundations of Object-Oriented Languages",
  pages = "28-40",
  year = "2002",
  link = "\url{http://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/first_class_modules.pdf}",
  abstract =
    "Though Haskell's module language is quite weak, its core language
    is highly expressive. Indeed, it is tantalisingly close to being
    able to express much of the structure traditionally delegated to a
    separate module language. However, the encoding are awkward, and
    some situations can't be encoded at all.

    In this paper we refine Haskell's core language to support
    {\sl first-class modules} with many of the features of ML-style
    modules. Our proposal cleanly encodes signatures, structures and
    functors with the appropriate type abstraction and type sharing,
    and supports recursive modules. All of these features work across
    compilation units, and interact harmoniously with Haskell's class
    system. Coupled with support for staged computation, we believe
    our proposal would be an elegant approach to run-time dynamic
    linking of structured code.

    Our work builds directly upon Jones' work on parameterised
    signatures, Odersky and Laufer's system of higher-ranked type
    annotations, Russo's semantics of ML modules using ordinary
    existential and universal quantifications, and Odersky and
    Zenger's work on nested types. We motivate the system by examples,
    and include a more formal presentation in the appendix.",
  paper = "Shie02.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Shoe67,
  author = "Shoenfield, Joseph R.",
  title = {{Mathematical Logic}},
  publisher = "Association for Symbolic Logic",
  year = "1967",
  isbn = "1-56881-135-7"
}
```

———-

— **axiom.bib** —

```
@misc{Shul18,
  author = "Shulman, Michael",
  title = {{Linear Logic for Constructive Mathematics}},
  link = "\url{https://arxiv.org/pdf/1805.07518.pdf}",
  year = "2018",
  abstract =
    "We show that numerous distinctive concepts of constructive
    mathematics arise automatically from an interpretation of 'linear
    higher-order logic' into intuitionistic higher-order logic via a Chu
    construction.  This includes apartness relations, complemented
    subsets, anti-subgroups and anti-ideals, strict and non-strict order
    pairs, cut-valued metrics, and apartness spaces. We also explain the
    constructive bifurcation of classical concept s using the choice
    between multiplicative and additive linear connectives. Linear logic
    thus systematically 'constructivizes' classical definitions and deals
    automatically with the resulting bookkeeping, and could potentially
    be used directly as a basis for constructive mathematics in place
    of intuitionistic logic.",
  paper = "Shul18.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Smit90,
  author = "Smith, A.",
  title = {{The Knuth-Bendix Completion Algorithm and its Specification in Z}},
  type = "technical report",
  institution = "Ministry of Defence, RSRE Malvern WORCS",
  year = "1990",
  number = "RSRE Memorandum 4323",
  abstract =
    "Proving that something is a consequence of a set of axioms can be
    very difficult. The Knuth-Bendix completion algorithm attempts to
    automate this process when the axioms are equations. The algorithm
    is bound up in the area of term rewriting, and so this memorandum
    contains an introduction to the tehory of term rewriting, followed
    by an overview of the algorithm. Finally a formal specification of
```

```
    the algorithm is given using the language Z.",
  paper = "Smit90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Smit15,
  author = "Smith, Peter",
  title = {{Some Big Books on Mathematical Logic}},
  year = "2015",
  link = "\url{}",
  paper = "Smit15.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Soko87,
  author = "Sokolowski, Stefan",
  title = {{Soundness of Hoare's Logic: An Automated Proof Using LCF}},
  journal = "Trans. on Programming Languages and Systems",
  volume = "9",
  number = "1",
  pages = "100-120",
  year = "1987",
  abstract =
    "This paper presents a natural deduction proof of Hoare's logic
    carried out by the Edinburgh LCF theorem prover. The emphasis is
    on the way Hoare's theory is presented to the LCF, which looks
    very much like an exposition of syntax and semantics to human
    readers; and on the programmable heuristics (tactics). We also
    discuss some problems and possible improvements to the LCF.",
  paper = "Soko87.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Sore94,
  author = "Sorenson, Jonathan",
  title = {{Two Fast GCD Algorithms}},
  journal = "Journal of Algorithms",
  volume = "16",
  pages = "110-144",
```

```
  year = "1994",
  abstract =
    "We present two new algorithms for computing geatest common
    divisors: the right- and left-shift $k$-ary GCD algorithms. These
    algorithms are generalizations of the binary and left-shift binary
    algorithms. Interestingly, both new algorithms have sequential
    versions that are practical and efficient and parallel versions
    that rival the best previous parallel GCD algorithms. We show that
    sequential versions of both algorithms take $O(n^2/log~n)$ bit
    operations in the worst case to compute the GCD of two n-bit
    integers. This compares favorably to the Euclidean and moth
    precision versions of these GCD algorithms, and we found that both
    $k$-ary algorithms are faster than the Euclidean and binary
    algorithms on inputs of 100 to 1000 decimal digits in length. We
    show that parallel versions of both algorithms match the
    complexity of the best previous parallel GCD algorithm due to Chor
    and Goldreich. Specifically, if $log~n\le k \le 2^n$ and $k$ is a
    power of two, then both algorithms run in $O(n/log~n+log^2~n)$
    time using a number of processors bounded by a polynomial in $n$
    and $k$ on a common CRCW PRAM. We also discuss extended versions
    of both algorithms.",
  paper = "Sore94.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Stee78,
  author = "Steele, Guy Lewis",
  title = {{RABBIT: A Compiler for SCHEME}},
  institution = "MIT",
  type = "technical report",
  number = "AITR-474",
  year = "1978",
  abstract =
    "We have developed a compiler for the lexically-scoped dialect of LISP
    known as SCHEME. The compiler knows relatively little about specific
    data manipulation primitives such as arithmetic operators, but
    concentrates on general issues of environment and control. Rather than
    having specialized knowledge about a large variety of control and
    environment constructs, the compiler handles only a small basis set
    which reflects the semantics of lambda-calculus. All of the
    traditional imperative constructs, such as sequencing, assignment,
    looping, GOTO, as well as many standard LISP constructs such as AND,
    OR, and COND, are expressed in macros in terms of the applicative
    basis set. A small number of optimization techniques, coupled with the
    treatment of function calls as GOTO statements, serve to produce code
    as good as that produced by more traditional compilers. The macro
    approach enables speedy implementation of new constructs as desired
    without sacrificing efficiency in the generated code.
```

```
    A fair amount of analysis is devoted to determining whether
    environments may be stack-allocated or must be heap-allocated.
    Heap-allocated environments are necessary in general because SCHEME
    (unlike Algol 60 and Algol 68, for example) allows procedures with
    free lexically scoped variables to be returned as the values of other
    procedures; the Algol stack-allocation environment strategy does not
    suffice. The methods used here indicate that a heap-allocating
    generalization of the 'display' technique leads to an efficient
    implementation of such 'upward funargs'. Moreover, compile-time
    optimization and analysis can eliminate many 'funargs' entirely, and
    so far fewer environment structures need be allocated at run time than
    might be expected.

    A subset of SCHEME (rather than triples, for example) serves as the
    representation intermediate between the optimized SCHEME code and the
    final output code; code is expressed in this subset in the so-called
    continuation-passing style. As a subset of SCHEME, it enjoys the same
    theoretical properties; one could even apply the same optimizer used
    on the input code to the intermediate code. However, the subset is so
    chosen that all temporary quantities are made manifest as variables,
    and no control stack is needed to evaluate it. As a result, this
    apparently applicative representation admits an imperative
    interpretation which permits easy transcription to final imperative
    machine code. These qualities suggest that an applicative language
    like SCHEME is a better candidate for an UNCOL than the more
    imperative candidates proposed to date.",
  paper = "Stee78.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Stee78a,
  author = "Steele, Guy Lewis and Sussman, Gerald Jay",
  title = {{The Art of the Interpreter}},
  institution = "MIT",
  type = "technical report",
  number = "AI Memo No. 453",
  year = "1978",
  abstract =
    "We examine the effects of various language design decisions on
    the programming styles available to a user of the language, with
    particular emphasis on the ability to incrementatlly construct
    modular systems. At each step we exhibit an interactive
    meta-circular interpreter for the language under
    consideration. Each new interpreter is the result of an
    incremental change to the previous interpreter.

    We explore the consequences of various variable binding
    disciplines and the introduction of side effects. We find that
    dynamic scoping is unsuitable for constructing procedural
    abstractions, but has another role as an agent of modularity,
```

```
    being a structured form of side effect. More general side effects
    are also found to be necessary to promote modular style. We find
    that the notion of side effect and the notion of equality (object
    identity) are mutually constraining; to define one is to define
    the other.

    The interpreters we exhibit are all written in a simple dialect of
    LISP, and all implement LISP-like languages. A subset of these
    interpreters constitute a partial historical reconstruction of the
    actual evolution of LISP.",
  paper = "Stee78a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Stou11,
  author = "Stoutemyer, David R.",
  title = {{Ways to Implement Computer Algebra Compactly}},
  journal = "ACM Comm. in Computer Algebra",
  volume = "45",
  number = "4",
  year = "2011",
  abstract =
    "Computer algebra had to be implemented compactly to fit on early
    personal computers and hand-held calculators. Compact implementation
    is still important for portable hand-held devices. Also, compact
    implementation increases comprehensibility while decreasing develop-
    ment and maintenance time and cost, regardless of the platform. This
    article describes several ways to achieve compact implementations,
    including:
    \begin{itemize}
    \item Exploit evaluation followed by interpolation to avoid
    implementing a parser, such as in PicoMath
    \item Use contiguous storage as an expression stack to avoid garbage
    collection and pointer-space overhead, such as in Calculus Demon
    and TI-Math-Engine
    \item Use various techniques for saving code space for linked-storage
    representation of expressions and functions, such as muMath and Derive
    \end{itemize}",
  paper = "Stou11.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Stou11a,
  author = "Stoutemyer, David R.",
  title = {{Ten Commandments for Good Default Expression Simplification}},
```

```
  journal = "J. Symbolic Computation",
  volume = "46",
  pages = "859-887",
  year = "2011",
  abstract =
    "This article provides goals for the design and improvement of default
    computer algebra expression simplification. These goals can also help
    users recognize and partially circumvent some limitations of their
    current computer algebra systems. Although motivated by computer
    algebra, many of the goals are also applicable to manual
    simplification, indicating what transformations are necessary and
    sufficient for good simplification when no particular canonical result
    form is required.

    After motivating the ten goals, the article then explains how the
    Altran partially factored form for rational expressions was extended
    for Derive and for the computer algebra in Texas Instruments products
    to help fulfill these goals. In contrast to the distributed Altran
    representation, this recursive partially factored semi-fraction form:
    \begin{itemize}
    \item does not unnecessarily force common denominators
    \item discovers and preserves significantly more factors
    \item can represent general expressions
    \item can produce an entire spectrum from fully factored over a
    common denominator through complete multivariate partial fractions,
    including a dense subset of all intermediate forms.
    \end{itemize}",
  paper = "Stou11a.pdf",
  keywords = "printed, DONE"
}
```

———-

— axiom.bib —

```
@article{Stou12,
  author = "Stoutemyer, David R.",
  title = {{Series Crimes}},
  journal = "ACM Comm. in Computer Algebra",
  volume = "46",
  number = "2",
  year = "2012",
  abstract =
    "Puiseux series are power series in which the exponents can be
    fractional and/or negative rational numbers. Several computer algebra
    systems have one or more built-in or loadable functions for computing
    truncated Puiseux series. Some are generalized to allow coefficients
    containing functions of the series variable that are dominated by any
    power of that variable, such as logarithms and nested logarithms of
    the series variable. Some computer algebra systems also have built-in
    or loadable functions that compute infinite Puiseuxseries.
    Unfortunately, there are some little-known pitfalls in
    computing Puiseux series. The most serious of these is expansions
```

```
    within branch cuts or at branch points that are incorrect for some
    directions in the complex plane. For example with each series
    implementation accessible to you:

    Compare the value of $(z^2 + z^3)^{3/2}$  with that of its
    truncated series expansion about $z = 0$, approximated at
    $z = 0.01$. Does the series converge to a value that is
    the negative of the correct value?

    Compare the value of $ln(z^2 + z^3)$ with its truncated series
    expansion about $z = 0$, approximated at $z= 0.01 + 0.1i$. Does
    the series converge to a value that is incorrect by $2\pi i$?

    Compare $arctanh(2 + ln(z)z)$ with its truncated series
    expansion about $z = 0$, approximated at $z = 0.01$. Does the series
    converge to a value that is incorrect by about $\pi i$?

    At the time of this writing, most implementations that accommodate
    such series exhibit such errors. This article describes how to avoid
    these errors both for manual derivation of series and when
    implementing series packages.",
  paper = "Stou12.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Stou12a,
  author = "Stoutemyer, David R.",
  title = {{Can the Eureqa Symbolic Regression Program, Computer
            Algebra and Numerical Analysis help each other?}},
  link = "\url{https://arxiv.org/pdf/1203.1023.pdf}",
  year = "2012",
  abstract =
    "The free Eureqa program has recently received extensive press
    praise. A representitive quote is
    \begin{quote}
      There are very clever 'thinking machines' in existence today,
      such as Watson, the IBM computer that conquered {\sl Jeopardy!}
      last year. But next to Eureqa, Watson is merely a glorified
      search engine.
    \end{quote}

    The program is designed to work with noisy experimental data,
    searching for then returning a set of result expressions that
    attempt to optimally trade off conciseness with accuracy.

    However, if the data is generated from a formula for which there
    exists more concise equivalent formulas, sometimes some of the
    candidate Eureqa expressions are one or more of those more concise
    equivalents expressions. If not, perhaps one or more of the
    returned Eureqa expressions might be a sufficiently accurate
```

```
    approximation that is more concise than the given
    formula. Moreover, when there is no known closed form expression,
    the data points can be generated by numerical methods, enabling
    Eureqa to find expressions that concisely fit those data points
    with sufficient accuracy. In contrast to typical regression
    software, the user does not have to explicitly or implicitly
    provide a specific expression or class of expressions containing
    unknown constants for the software to determine.

    Is Eureqa useful enough in these regards to provide an additional
    tool for experimental mathematics, computer algebra users and
    numerical analysts? Yes, if used carefully. Can computer algebra
    and numerical methods help Eureqa? Definitely.",
  paper = "Stou12a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Stou13,
  author = "Stoutemyer, David R.",
  title = {{A Computer Algebra User Interface Manifesto}},
  journal = "ACM Communications in Computer Algebra",
  volume = "47",
  number = "4",
  year = "2013",
  abstract =
    "Many computer algebra systems have more than 1000 built-in functions,
    making expertise difficult. Using mock dialog boxes, this article
    describes a proposed interactive general-purpose wizard for organizing
    optional transformations and allowing easy fine grain control over the
    form of the result  even by amateurs. This wizard integrates ideas
    including:
    \begin{itemize}
    \item flexible subexpression selection;
    \item complete control over the ordering of variables and
    commutative operands, with well-chosen defaults;
    \item interleaving the choice of successively less main
    variables with applicable function choices to provide detailed
    control without incurring a combinatorial number of applicable
    alternatives at any one level;
    \item quick applicability tests to reduce the listing of
    inapplicable transformations;
    \item using an organizing principle to order the alternatives
    in a helpful manner;
    \item  labeling quickly-computed alternatives in dialog boxes
    with a preview of their results, using ellipsis elisions if
    necessary or helpful;
    \item allowing the user to retreat from a sequence of choices
    to explore other branches of the tree of alternatives  or to
    return quickly to branches already visited;
    \item allowing the user to accumulate more than one of the
```

```
    alternative forms;
    \item integrating direct manipulation into the wizard; and
    \item supporting not only the usual input-result pair mode, but
    also the useful alternative derivational and in situ replacement
    modes in a unified window.
    \end{itemize}",
  paper = "Stou13.pdf",
  kryeotfd = "printed, DONE"
}p
```

———-

— **axiom.bib** —

```
@article{Stuc05,
  author = "Stuckey, Peter J. and Sulzmann, Martin",
  title = {{A Theory of Overloading}},
  journal = "ACM Trans. on Programming Languages and Systems",
  volume = "27",
  number = "6",
  pages = "1-54",
  year = "2005",
  abstract =
    "We present a novel approach to allow for overloading of
    identifiers in the spirit of type classes. Our approach relies on
    the combination of the HM(X) type system framework with Constraint
    Handling Rules (CHRs). CHRs are a declarative language for writing
    incremental constraint solvers, that provide our scheme with a
    form of programmable type language. CHRs allow us to precisely
    describe the relationships among overloaded identifiers. Under
    some sufficient conditions on the CHRs we achieve decidable type
    inference and the semantic meaning of programs is unambiguous. Our
    approach provides a common formal basis for many type class
    extensions such as multi-parameter type classes and functional
    dependencies.",
  paper = "Stuc05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Suxx68,
  author = "Su, Jessica",
  title = {{Solving Recurrences}},
  link =
  "\url{https://web.stanford.edu/class/archive/cs/cs161/cs161.1168/lecture3.pdf}",
  year = "1968",
  comment = "Stanford CS161 Lecture 3 class notes",
  paper = "Suxx68.pdf",
  keywords = "printed"
```

```
}
```

———-

1.17.20 T

— **axiom.bib** —

```
@inproceedings{Talp92,
  author = "Talpin, Jean-Pierre and Jouvelot, Pierre",
  title = {{The Type and Effect Discipline}},
  booktitle = "Conf. on Logic in Computer Science",
  publisher = "Computer Science Press",
  year = "1992",
  abstract =
    "The {\sl type and effect discipline} is a new framework for
    reconstructing the principal type and the minimal effect of
    expressions in implicitly typed polymorphic functional languages that
    support imperative constructs. The type and effect discipline
    outperforms other polymorphic type systems. Just as types abstract
    collections of concrete values, {\sl effects} denote imperative
    operations on regions. {\sl Regions} abstract sets of possibly aliased
    memory locations.

    Effects are used to control type generalization in the presence of
    imperative constructs while regions delimit observable
    side-effects. The observable effects of an expression range over the
    regions that are free in its type environment and its type; effects
    related to local data structures can be discarded during type
    reconstruction. The type of an expression can be generalized with
    respect to the variables that are not free in the type environment or
    in the observable effect.",
  paper = "Talp92.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Tard90,
  author = "Tarditi, David and Acharya, Anurag and Lee, Peter",
  title = {{No Assembly Required: Compiling Standard ML to C}},
  type = "technical report",
  institution = "Carnegie Mellon University",
  number = "CMU-CS-90-187",
  year = "1990",
  abstract =
    "C has been proposed as a portable target language for
    implementing higher-order languages. Previous efforts at compiling
    such languages to C have produced efficient code, but have had to
```

```
    compromise on either the portability or the preservation of the
    tail-recursive properties of the languages. We assert that neither
    of these compromises is necessary for the generation of efficient
    code. We offer a Standard ML to C compiler, which does not make
    wither of these compromises, as an existence proof. The generated
    code achieves an execution speed that is just a factor of two
    slower than the best native code compiler. In this paper, we
    describe the design, implementation and the performance of this
    compiler.",
  paper = "Tard90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Tars46,
  author = "Tarski, Alfred",
  title = {{Introduction to Logic}},
  publisher = "Dover",
  year = "1946",
  isbn = "13-978-0-486-28462-0"
}
```

———-

— **axiom.bib** —

```
@article{Tars69,
  author = "Tarski, Alfred",
  title = {{Truth and Proof}},
  journal = "Scientific American",
  volume = "1969",
  pages = "63-77",
  year = "1969",
  paper = "Tars69.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Tobi11,
  author = "Tobin-Hochstadt, Sam and Felleisen, Matthias",
  title = {{The Design and Implementation of Typed Scheme: From
            Scripts to Programs}},
  link = "\url{https://arxiv.org/pdf/1106.2575.pdf}",
  year = "2011",
  abstract =
```

```
    "When scripts in untyped languages grow into large programs,
    maintaining them becomes difficult. A lack of explicit type
    annotations in typical scripting languages forces programmers to must
    (re)discover critical pieces of design information every time they
    wish to change a program. This analysis step both slows down the
    maintenance process and may even introduce mistakes due to the
    violation of undiscovered invariants.

    This paper presents Typed Scheme, an explicitly typed extension of PLT
    Scheme, an untyped scripting language. Its type system is based on
    the novel notion of occurrence typing, which we formalize and
    mechanically prove sound. The implementation of Typed Scheme
    additionally borrows elements from a range of approaches, including
    recursive types, true unions and subtyping, plus polymorphism combined
    with a modicum of local inference.

    The formulation of occurrence typing naturally leads to a simple and
    expressive version of predicates to describe refinement types. A
    Typed Scheme program can use these refinement types to keep track of
    arbitrary classes of values via the type system. Further, we show how
    the Typed Scheme type system, in conjunction with simple recursive
    types, is able to encode refinements of existing datatypes, thus
    expressing both proposed variations of refinement types.",
  paper = "Tobi11.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Toft09,
  author = "Tofte, Mads",
  title = {{Tips for Computer Scientists on Standard ML (Revised)}},
  link = "\url{http://www.itu.dk/people/tofte/publ/tips.pdf}",
  year = "2009",
  paper = "Toft09.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Tuho18,
  author = "Tuhola, Henri",
  title = {{An another MLsub study}},
  link = "\url{http://boxbase.org/entries/2018/mar/12/mlsub-study/}",
  year = "2018",
  comment = "Subtyping does not solve the problem of coercion"
}
```

———-

— **axiom.bib** —

```
@misc{Tuho18a,
  author = "Tuhola, Henri",
  title = {{Boxbase - Index}},
  link = "\url{http://boxbase.org/}",
  year = "2018",
  keywords = "axiomref"
}
```

———-

1.17.21 U

1.17.22 V

— **axiom.bib** —

```
@misc{Vien93,
  author = "Vienneau, Robert L.",
  title = {{A Review of Formal Methods}},
  year = "1993",
  paper = "Vien93.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Vuil73,
  author = "Vuillemin, Jean",
  title = {{Proof Techniques for Recursive Programs}},
  school = "Stanford University",
  year = "1973",
  abstract =
    "The concept of least fixed-point of a continuous function can be
    considered as the unifying thread of this dissertation.

    The connections between fixed-points and recursive programs are
    detailed in chapter 2, providing some insights on practical
    implementations of recursion. There are two usual characterizations
    of the least fixed-point of a continuous function. To the first
    characterization, due to Knaster and Tarski, corresponds a class
    of proof techniques for programs, as described in Chapter 3. The
    other characterization of least fixed points, better known as
    Kleene's first recursion theorem, is discussed in Chapter 4. It
    has the advantage of being effective and it leads to a wider class
    of proof techniques.",
```

```
  paper = "Vuil73.pdf"
}
```

———-

1.17.23 W

— **axiom.bib** —

```
@misc{Waal03,
  author = "Waaldijk, Frank",
  title = {{On the Foundations of Constructive Mathematics}},
  link = "\url{}",
  year = "2003",
  abstract =
    "We discuss the foundations of constructive mathematics, including
    recursive mathematics and intuitionism, in relation to classical
    mathematics.  There are connections with the foundations of physics,
    due to the way in which the different branches of mathematics reflect
    reality.  Many different axioms and their interrelationship are
    discussed.  We show that there is a fundamental problem in bish
    (Bishops school of constructive mathematics) with regard to its
    current definition of continuous function. This problem is closely
    related to the definition in bish of locally compact.

    Possible approaches to this problem are discussed. Topology seems to
    be a key to understanding many issues. We offer several new
    simplifying axioms, which can form bridges between the various
    branches of constructive mathematics and classical mathematics
    (reuniting the antipodes).  We give a simplification of basic
    intuitionistic theory, especially with regard to so-called bar
    induction.  We then plead for a limited number of axiomatic systems,
    which differentiate between the various branches of mathematics.
    Finally, in the appendix we offer bish an elegant topological
    definition of locally compact, which unlike the current definition
    is equivalent to the usual classical and/or intuitionistic definition
    in classical and intuitionistic mathematics respectively.",
  paper = "Waal03.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wadl95,
  author = "Wadler, Philip",
  title = {{Monads for Functional Programming}},
  journal = "LNCS",
  volume = "925",
  year = "1995",
  abstract =
```

```
    "The use of monads to structure functional programs is
    described. Monads provide a convenient framework for simulating
    effects found in other languages, such as global state, exception
    handling, output, or non-determinism. Three case studies are
    looked at in detail: how monads ease the modification of a simple
    evaluator; how monads act as the basis of a datatype of arrays
    subject to in-place update; and how monads can be used to build
    parsers.",
  paper = "Wadl95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Wadl92,
  author = "Wadler, Philip",
  title = {{Comprehending Monads}},
  journal = "Mathematical Structures in Computer Science",
  volume = "2",
  pages = "461-493",
  year = "1992",
  abstract =
    "Category theorists invented {\sl monads} in the 1960s to
    concisely express certain aspects of universal algebra. Functional
    programmers invented {\sl list comprehensions} in the 1970s ot
    concisely express certain programs involving lists. This paper
    shows how list comprehensions may be generalised to an arbitrary
    monad, and how the resulting programming feature can concisely
    express in a pure functional language some programs that
    manipulate state, handle exceptions, parse text, or invoke
    continuations. A new solution to the old problem of destructive
    array update is also presented. No knowledge of category theory is
    assumed.",
  paper = "Wadl92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wadl07,
  author = "Wadler, Philip",
  title = {{The Girard-Reynolds isomorphism (second edition)}},
  journal = "Theoretical Computer Science",
  volume = "375",
  pages = "201-226",
  year = "2007",
  abstract =
    "Jean-Yves Girard and John Reynolds independently discovered the
    second-order polymorphic lambda calculus, F2. Girard additionally
```

```
    proved a Representation Theorem : every function on natural numbers
    that can be proved total in second-order intuitionistic predicate
    logic, P2, can be represented in F2. Reynolds additionally proved an
    Abstraction Theorem : every term in F2 satisfies a suitable notion of
    logical relation; and formulated a notion of parametricity satisfied
    by well-behaved models.

    We observe that the essence of Girards result is a projection from P2
    into F2, and that the essence of Reynoldss result is an embedding of
    F2 into P2, and that the Reynolds embedding followed by the Girard
    projection is the identity. We show that the inductive naturals are
    exactly those values of type natural that satisfy Reynoldss notion of
    parametricity, and as a consequence characterize situations in which
    the Girard projection followed by the Reynolds embedding is also the
    identity.

    An earlier version of this paper used a logic over untyped terms. This
    version uses a logic over typed term, similar to ones considered by
    Abadi and Plotkin and by Takeuti, which better clarifies the
    relationship between F2 and P2.",
  paper = "Wadl07.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wadl07a,
  author = "Wadler, Philip and Findler, Robert Bruce",
  title = {{Well-Typed Programs Can't Be Blamed}},
  booktitle = "Workshop on Scheme and Functional Programming",
  year = "2007",
  pages = "15-26",
  abstract =
    "We show how {\sl contracts} with blame fit naturally with recent
    work on {\sl hybrid types} and {\sl gradual types}. Unlike hybrid
    types or gradual types, we require casts in the source code, in
    order to indicate where type errors may occur. Two (perhaps
    surprising) aspects of our approach are that refined types can
    provide useful static guarantees even in the absence of a theorem
    prover, and that type {\sl dynamic} should not be regarded as a
    supertype of all other types. We factor the well-known notion of
    subtyping into new notions of positive and negative subtyping, and
    use these to characterise where positive and negative blame may
    arise. Our approach sharpens and clarifies some recent results in
    the literature.",
  paper = "Wadl07a.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Webe05a,
  author = "Weber, Kenneth and Trevisan, Vilmar and Martins, Luiz Felipe",
  title = {{A Modular Integer GCD Algorithm}},
  journal = "J. of Algorithms",
  volume = "54",
  year = "2005",
  pages = "152-167",
  abstract =
    "This paper describes the first algorithm to compute the greatest
    common divisor (GCD) of two $n$-bit integers using a modular
    representation for intermediate values $U$, $V$ and also for the
    result. It is based on a reduction step, similar to one used in
    the accelerated algorithm when $U$ and $V$ are close to the same
    size, that replaces $U$ by $(U-bV)/p$, where $p$ is one of the
    prime moduli and $b$ is the unique integer in the interval
    $(-p/2,p/2)$ such that $b\equiv UV^{-1} (mod~p)$. When the
    algorithm is executed on a bit common CRCW PRAM with
    $O(n~log~n~log~log~log~n)$ processors, it takes $(O(n)$ time in
    the worst case. A heuristic model of the average case yields
    $O(n/log~n)$ time on the same number of processors.",
  paper = "Webe05a.pdf",
  keywords = "printed, axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Wehr05,
  author = "Wehr, Stefan",
  title = {{ML Modules and Haskell Type Classes: A Constructive
            Comparison}},
  school = "Albert Ludwigs Universitat",
  year = "2005",
  abstract =
    "Researchers repeatedly observed that the module system of ML and the
    type class mechanism of Haskell are related. So far, this relationship
    has received little formal investigation. The work at hand fills this
    gap: It introduces type-preserving translations from modules to type
    classes and vice versa, which enable a thorough comparison of the two
    concepts.

    The source language of the first translation is a subset of
    Standard ML. The target language is Haskell with common extensions
    and one new feature, which was deeloped as part of this work. The
    second translation maps a subset of Haskell 98 to ML, with
    well-established extensions. I prove that the translations
    preserve type correctness and provide implementations for both.

    Building on the insights obtained from the translations, I present
    a thorough comparison between ML modules and Haskell type
    classes. Moreover, I evaluate to what extent the techniques used
```

```
    in the translations can be exploited for modular programming in
    Haskell and for programming with ad-hoc polymorphism in ML.",
  paper = "Wehr05.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wehr08,
  author = "Wehr, Stefan and Chakravarty, Maneul M.T.",
  title = {{ML Modules and Haskell Type Classes: A Constructive
            Comparison}},
  journal = "LNCS",
  volume = "5356",
  pages = "188-204",
  year = "2008",
  abstract =
    "Researchers repeatedly observed that the module system of ML and the
    type class mechanism of Haskell are related. So far, this relationship
    has received little formal investigation. The work at hand fills this
    gap: It introduces type-preserving translations from modules to type
    classes and vice versa, which enable a thorough comparison of the two
    concepts.",
  paper = "Wehr08.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Weir12,
  author = "Weirich, Jim",
  title = {{Y Not? -- Adventures in Functional Programming}},
  year = "2012",
  link = "\url{https://www.infoq.com/presentations/Y-Combinator}",
  abstract = "Explains the Y-Combinator"
}
```

———-

— **axiom.bib** —

```
@article{Wied03a,
  author = "Wiedijk, Freek and Zwanenburg, J.",
  title = {{First Order Logic with Domain Conditions}},
  journal = "LNCS",
  volume = "2758",
  pages = "221-237",
  year = "2003",
```

```
  abstract =
    "This paper addresses the crucial issue in the design of a proof
    development system of how to deal with partial functions and the
    related question of how to treat undefined terms. Often the problem is
    avoided by artificially making all functions total. However, that does
    not correspond to the practice of everyday mathematics.

    In type theory partial functions are modeled by giving functions extra
    arguments which are proof objects. In that case it will not be
    possible to apply functions outside their domain. However, having
    proofs as first class objects has the disadvantage that it will be
    unfamiliar to most mathematicians. Also many proof tools (like the
    theorem prover Otter) will not be usable for such a logic. Finally
    expressions get difficult to read because of these proof objects.

    The PVS system solves the problem of partial functions
    differently. PVS generates type-correctness conditions (TCCs) for
    statements in its language. These are proof obligations that have to
    be satisfied on the side to show that statements are well-formed.

    We propose a TCC-like approach for the treatment of partial functions
    in type theory. We add domain conditions (DCs) to classical
    first-order logic and show the equivalence with a first order system
    that treats partial functions in the style of type theory.",
  paper = "Wied03a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Wied18,
  author = "Wiedijk, Freek",
  title = {{Formaizing 100 Theorems}},
  link = "\url{http://www.cs.ru.nl/~freek/100/}",
  year = "2018"
}
```

———-

— **axiom.bib** —

```
@misc{Wied08,
  author = "Wiedijk, Freek",
  title = {{Formai Proof -- Geting Started}},
  link = "\url{https://www.ams.org/notices/200811/tx081101408p.pdf}",
  year = "2008",
  paper = "Wied08.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wins12,
  author = "Winston, Patrick Henry",
  title = {{The Nex 50 Years: A Personal View}},
  booktitle = "Biologially Inspired Cognitive Architectures 1",
  year = "2012",
  pages = "92-99",
  publisher = "Elsevier B.V",
  abstract =
    "I review history, starting with Turings seminal paper, reaching
    back ultimately to when our species started to outperform other
    primates, searching for the questions that will help us develop a
    computational account of human intelligence. I answer that the
    right questions are: Whats different between us and the other
    primates and whats the same.  I answer the {\sl whats different}
    question by saying that we became symbolic in a way that enabled
    story understanding, directed perception, and easy communication,
    and other species did not. I argue against Turings
    reasoning-centered suggestions, offering that reasoning is just a
    special case of story understanding. I answer the {\sl whats the
    same} question by noting that our brains are largely engineered in
    the same exotic way, with information flowing in all directions at
    once. By way of example, I illustrate how these answers can
    influence a research program, describing the Genesis system, a
    system that works with short summaries of stories, provided in
    English, together with low-level {\sl common-sense rules} and
    higher-level {\sl concept patterns}, likewise expressed in
    English. Genesis answers questions, notes abstract concepts such
    as {\sl revenge}, tells stories in a listener-aware way, and fills
    in story gaps using precedents. I conclude by suggesting,
    optimistically, that a genuine computational theory of human
    intelligence will emerge in the next 50 years if we stick to the
    right, biologically inspired questions, and work toward
    biologically informed models.",
  paper = "Wins12.pdf"
}
```

———-

1.17.24 X

— **axiom.bib** —

```
@inproceedings{Xixx99,
  author = "Xi, Hongwei and Pfenning, Frank",
  title = {{Dependent Types in Practical Programming}},
  booktitle = "SIGPLAN-SIGACT Symp. on Principles of Programming
               Languages",
  year = "1990",
  link = "\url{https://www.cs.cmu.edu/~fp/papers/popl99.pdf}",
  publisher = "ACM",
```

```
  abstract =
    "We present an approach to enriching the type system of ML with a
    restricted form of dependent types, where type index objects are drawn
    from a constraint domain C , leading to the DML( C ) language schema.
    This allows specification and inference of significantly more precise
    type information, facilitating program error detection and compiler
    optimization.  A major complication resulting from introducing
    dependent types is that pure type inference for the enriched system is
    no longer possible, but we show that type-checking a suciently
    annotated program in DML( C ) can be reduced to constraint
    satisfaction in the constraint domain C .  We exhibit the
    un-obtrusiveness of our approach through practical examples and prove
    that DML( C ) is conservative over ML. The main contribution of the
    paper lies in our language design, including the formulation of
    type-checking rules which makes the approach practical.  To our
    knowledge, no previous type system for a general purpose programming
    language such as ML has combined dependent types with features including
    datatype declarations, higher-order functions, general recursions,
    let-polymorphism, mutable references, and exceptions.  In addition,
    we have finished a prototype implementation of DML( C ) for an integer
    constraint domain C , where constraints are linear inequalities (Xi
    and Pfenning 1998).",
  paper = "Xixx99.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@phdthesis{Xuex13,
  author = "Xue, Tao",
  title = {{Theory and Implementation of Coercive Subtyping}},
  school = "University of London",
  year = "2013",
  abstract =
    "Coercive subtyping is a useful and powerful framework of subtyping
    for type theories.  In this thesis, we point out the problem in the
    old formulation of coercive subtyping in [Luo99], give a new and
    adequate formulation $T[C]$, the system that extends a type theory $T$
    with coercive subtyping based on a set $C$ of basic subtyping
    judgements, and show that coercive subtyping is a conservative
    extension and, in a more general sense, a definitional extension.

    We introduce an intermediate system, the star-calculus $T[C]^$ , in
    which the positions that require coercion insertions are marked, and
    show that $T[C]^$ is a conservative extension of $T$ and that $T[C]^*$
    is equivalent to $T[C]$.  Further more, in order to capture all the
    properties of the coercive subtyping framework, we introduce another
    intermediate system $T[C]_{OK}$ which does not contain the coercion
    application rules.  We show that $T[C]^$ is actually a definitional
    extension of $T[C]_{OK}$, which is a conservative extension of $T$.
    This makes clear what we mean by coercive subtyping being a
```

```
    conservative extension and amends a technical problem that has led to
    a gap in the earlier conservativity proof.

    Another part of the work in this thesis concerns the implementation of
    coercive subtyping in the proof assistant Plastic.  Coercive subtyping
    was implemented in Plastic by Paul Callaghan [CL01].  We have done
    some improvement based on that work, fixed some problems of Plastic,
    and implemented a new kind of data type called {\sl dot-types}, which are
    special data types useful in formal semantics to describe interesting
    linguistic phenomena such as copredication, in Plastic.",
  paper = "Xuex13.pdf"
}
```

———-

1.17.25 Y

1.17.26 Z

— **axiom.bib** —

```
@phdthesis{Zeil09,
  author = "Zeilberger, Noam",
  title = {{The Logical Basis of Evaluation Order and Pattern-Matching}},
  school = "Carnegie Mellon University",
  year = "2009",
  link = "\url{https://www.cs.cmu.edu/~rwh/theses/zeilberger.pdf}",
  abstract =
    "An old and celebrated analogy says that writing programs is like
    proving theorems. This analogy has been productive in both
    directions, but in particular has demonstrated remarkable utility in
    driving progress in programming languages, for example leading towards
    a better understanding of concepts such as abstract data types and
    polymorphism.  One of the best known instances of the analogy actually
    rises to the level of an isomorphism: between Gentzens natural
    deduction and Churchs lambda calculus.  However, as has been
    recognized for a while, lambda calculus fails to capture some of the
    important features of modern programming languages. Notably, it does
    not have an inherent notion of evaluation order, needed to make sense
    of programs with side effects. Instead, the historical descendents of
    lambda calculus (languages like Lisp, ML, Haskell, etc.) impose
    evaluation order in an ad hoc way.

    This thesis aims to give a fresh take on the proofs-as-programs
    analogyone which better accounts for features of modern programming
    languagesby starting from a different logical foundation.  Inspired
    by Andreolis focusing proofs for linear logic, we explain how to
    axiomatize certain canonical forms of logical reasoning through a
    notion of pattern.  Propositions come with an intrinsic polarity,
    based on whether they are defined by patterns of proof, or by patterns
    of refutation.  Applying the analogy, we then obtain a programming
    language with built-in support for pattern-matching, in which
```

```
    evaluation order is explicitly reflected at the level of typesand
    hence can be controlled locally, rather than being an ad hoc, global
    policy decision. As we show, different forms of continuation-passing
    style (one of the historical tools for analyzing evaluation order)
    can be described in terms of different polarizations.  This language
    provides an elegant, uniform account of both untyped and
    intrinsically-typed computation (incorporating ideas from infinitary
    proof theory), and additionally, can be provided an extrinsic type
    system to express and statically enforce more refined properties of
    programs. We conclude by using this framework to explore the theory of
    typing and subtyping for intersection and union types in the presence
    of effects, giving a simplified explanation of some of the unusual
    artifacts of existing systems.",
  paper = "Zeil09.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Zeil10,
  author = "Zeilberger, Doron",
  title = {{Against Rigor}},
  journal = "LNCS",
  volume = "6167",
  year = "2010",
  pages = "262-262",
  abstract =
    "The ancient Greeks gave (western) civilization quite a few
    gifts, but we should beware of Greeks bearing gifts. The gifts of theatre
    and democracy were definitely good ones, and perhaps even the gift of
    philosophy, but the 'gift' of the so-called 'axiomatic method' and the
    notion of 'rigorous' proof did much more harm than good. If we want
    to maximize Mathematical Knowledge, and its Management, we have to
    return to Euclid this dubious gift, and give-up our fanatical insistence
    on perfect rigor. Of course, we should not go to the other extreme, of
    demanding that everything should be non-rigorous. We should encourage
    diversity of proof-styles and rigor levels, and remember that nothing is
    absolutely sure in this world, and there does not exist an absolutely
    rigorous proof, nor absolute certainty, and 'truth' has many shades and
    levels.",
  paper = "Zeil10.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Zeil16,
  author = "Zeilberger, Noam",
  title = {{Towards a Mathematical Science of Programming}},
```

```
  year = "2016",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Zeil16a,
  author = "Zeilberger, Noam",
  title = {{Principles of Type Refinement}},
  booktitle = "OPLSS 2106",
  link = "\url{http://noamz.org/oplss16/refinements-notes.pdf}",
  year = "2016",
  paper = "Zeil16a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Zimm02,
  author = "Zimmer, Jurgen and Dennis, Louise A.",
  title = {{Inductive Theorem Proving and Computer Algebra in the
            MathWeb Software Bus}},
  journal = "LNCS",
  volume = "2385",
  year = "2002",
  abstract =
    "Reasoning systems have reached a high degree of maturity in the last
    decade. However, even the most successful systems are usually not
    general purpose problem solvers but are typically specialised on
    problems in a certain domain. The MathWeb Software Bus (MathWeb-SB) is a
    system for combining reasoning specialists via a common software bus.
    We describe the integration of the $\lambda$-Clam system, a reasoning
    specialist for proofs by induction, into the MathWeb-SB. Due to this
    integration, $\lambda$-Clam now offers its theorem proving expertise
    to other systems in the MathWeb-SB. On the other hand,
    $\lambda$-Clam can use the
    services of any reasoning specialist already integrated. We focus on
    the latter and describe first experiments on proving theorems by
    induction using the computational power of the Maple system within
    $\lambda$-Clam.",
  paper = "Zimm02.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Zeng11,
  author = "Zengler, Christoph and Kubler, Andreas and Kuchlin, Wolfgang",
  title = {{New Approaches to Boolean Quantifier Elimination}},
  journal = "ACM Comm. in Computer Algebra",
  volume = "45",
  number = "2",
  year = "2011",
  abstract =
    "We present four different approaches for existential Boolean
    quantifier elimination, based on model enumeration, resolution,
    knowledge compilation with projection, and substitution. We point out
    possible applications in the area of verification and we present
    preliminary benchmark results of the different approaches.",
  paper = "Zeng11.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Zhan02,
  author = "Zhang, Ting",
  title = {{A Survey of Quantifier Elimination: Syntactic and Semantic
            Approaches}},
  year = "2002",
  link = "\url{http://theory.stanford.edu/~tingz/talks/qe.ps}",
  paper = "Zhan02.pdf",
  keywords = "printed"
}
```

———-

1.18 Proving Axiom Sane – Coercion in CAS-Proof Systesms

— **axiom.bib** —

```
@incollection{Soze06,
  author = "Sozeau, Matthieu",
  title = {{Subset Coercions in Coq}},
  booktitle = "TYPES: Int. Workshop on Types for Proofs and Programs",
  publisher = "ACM Press",
  journal = "LNCS",
  volume = "4502",
  pages = "237-252",
  year = "2006",
  abstract =
    "We propose a new language for writing programs with dependent types
    on top of the Coq proof assistant. This language permits to establish
```

```
    a phase distinction between writing and proving algorithms in the Coq
    environment. Concretely, this means allowing to write algorithms as
    easily as in a practical functional programming language whilst giving
    them as rich a specification as desired and proving that the code
    meets the specification using the whole Coq proof apparatus.  This is
    achieved by extending conversion to an equivalence which relates
    types and subsets based on them, a technique originating from the
    ‘‘Predicate subtyping’’ featureof PVS and following mathematical
    convention. The typing judgements can be translated to the Calculus of
    (Co-)Inductive Constructions (Cic) by means of an interpretation
    which inserts coercions at the appropriate places.These coercions
    can contain existential variables representing the propositional
    parts of the final term, corresponding to proof obligations (or
    PVS type-checking conditions). A prototype implementation of this
    process is integrated with the Coq environment.",
  paper = "Soze06.pdf",
  keywords = "printed"
}
```

———-

1.19 Proving Axiom Correct – CAS-Proof System Survey

1.19.1 A

— **axiom.bib** —

```
@article{Aban16,
  author = "Abanades, M. and Botana, F. and Kovacs, Z. and Recio, T. and
            Solyom-Gecse, C.",
  title = {{Development of automatic reasoning tools in GeoGebra}},
  journal = "ACM Comm. Computer Algebra",
  volume = "50",
  pages = "85-88",
  year = "2016",
  abstract =
    "Much effort has been put into the implementation of automatic proving
    in interactive geometric environments (e.g. Java Geometry Expert,
    GeoGebra). The closely related concept of automatic discovery, remains
    however almost unexplored.

    This software presentation will demonstrate our results towards the
    incorporation of automatic discovery capabilities into GeoGebra, an
    educational software with tens of millions of users worldwide. As main
    result, we report on a new command, currently available in the
    official version, that allows the automatic discovery of loci of
    points in diagrams defined by implicit conditions. This represents an
    extension of a previous command, similar in nature, but restricted to
    loci defined by the standard mover-tracer construction. Our proposal
```

```
    successfully automates the 'dummy locus dragging' in dynamic
    geometry. This makes the cycle conjecturing-checking-proving
    accessible for general users in elementary geometry.",
  paper = "Aban16.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Abra15,
  author = "Abraham, Erika",
  title = {{Building Bridges between Symbolic Computation and Satisfiability
           Checking}},
  booktitle = "ISSAC 15",
  year = "2015",
  pages = "1-6",
  publisher = "ACM",
  isbn = "978-1-4503-3435-8",
  abstract =
    "The satisfiability problem is the problem of deciding whether a
    logical formula is satisfiable. For first-order arithmetic theories,
    in the early 20th century some novel solutions in form of decision
    procedures were developed in the area of mathematical logic. With the
    advent of powerful computer architectures, a new research line started
    to develop practically feasible implementations of such decision
    procedures. Since then, symbolic computation has grown to an extremely
    successful scientific area, supporting all kinds of scientific
    computing by efficient computer algebra systems.

    Independently, around 1960 a new technology called SAT solving started
    its career. Restricted to propositional logic, SAT solvers showed to
    be very efficient when employed by formal methods for verification. It
    did not take long till the power of SAT solving for Boolean problems
    had been extended to cover also different theories. Nowadays, fast
    SAT-modulo-theories (SMT) solvers are available also for arithmetic
    problems.

    Due to their different roots, symbolic computation and SMT solving
    tackle the satisfiability problem differently. We discuss differences
    and similarities in their approaches, highlight potentials of
    combining their strengths, and discuss the challenges that come with
    this task.",
  paper = "Abra15.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Abra16a,
```

```
  author = "Abraham, Erika",
  title = {{Symbolic Computation Techniques in Satisfiability Checking}},
  booktitle = "SYNASC 2016",
  publisher = "IEEE Press",
  year = "2016",
  isbn = "978-1-5090-5707-8",
  pages = "3-10"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Abra17,
  author = "Abraham, Erika and Jebelean, Tudor",
  title = {{Adapting Cylindrical Algebraic Decomposition for Proof Specific
           Tasks}},
  booktitle = "IJCAI 2017",
  year = "2017",
  comment = "Extended Abstract",
  paper = "Abra17.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Abra16,
  author = "Abraham, Erika and Abbott, John and Becker, Bernd and
            Bigatti, Anna M. and Brain, Martin and Buchberger, Bruno
            and Cimatti, Alessandro and Davenport, James H. and
            England, Matthew and Fontaine, Pascal and Forrest, Stephen
            and Griggio, Alberto and Kroenig, Daniel and
            Seiler, Werner M. and Sturm, Thomas",
  title = {{Satisfiability Checking meesg Symbolic Computation}},
  year = "2016",
  link = "\url{http://arxiv.org/abs/1607.08028}",
  abstract =
    "Symbolic Computation and Satisfiability Checking are two research
    areas, both having their individual scientific focus but sharing also
    common interests in the development, implementation and application of
    decision procedures for arithmetic theories. Despite their
    commonalities, the two communities are rather weakly connected. The
    aim of our newly accepted SC-square project (H2020-FETOPEN-CSA) is to
    strengthen the connection between these communities by creating common
    platforms, initiating interaction and exchange, identifying common
    challenges, and developing a common roadmap from theory along the way
    to tools and (industrial) applications. In this paper we report on the
    aims and on the first activities of this project, and formalise some
    relevant challenges for the unified SC-square community.",
  paper = "Abra16.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Abraxx,
  author = "Abraham, Erika and Abbott, John and Becker, Bernd and
            Bigatti, Anna M. and Brain, Martin and Cimatti, Alessandro and
            Davenport, James H. and England, Matthew and Fontaine, Pascal
            and Forrest, Stephen and Ganesh, Vijay and Griggio, Alberto and
            Kroenig, Daniel and Seiler, Werner M.",
  title = {{SC2 challenges: when Satisfiability Checking and Symbolic
           Computation join forces}},
  year = "2017",
  abstract =
    "Symbolic Computation and Satisfiability Checking are two research
    areas, both having their individual scientific focus but with common
    interests, e.g., in the development, implementation and application
    of decision procedures for arithmetic theories.  Despite their
    commonalities, the two communities are rather weakly connected.  The
    aim of the SC 2 initiative is to strengthen the connection between
    these communities by creating common platforms, initiating interaction
    and exchange, identifying common challenges, and developing a common
    roadmap from theory along the way to tools and (industrial) applications.",
  paper = "Abraxx.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Acze13,
  author = "Aczel, Peter et al.",
  title = {{Homotopy Type Theory: Univalent Foundations of Mathematics}},
  publisher = "Institute for Advanced Study",
  year = "2013",
  link =
    "\url{https://hott.github.io/book/nightly/hott-letter-1075-g3c53219.pdf}",
  paper = "Acze13.pdf",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Adam99,
  author = "Adams, Andrew A. and Gottlieben, Hanne and Linton, Steve A. and
            Martin, Ursula",
  title = {{Automated theorem proving in support of computer algebra:
           symbolic definite integration as a case study}},
```

```
  booktitle = "ISSAC '99",
  pages = "253-260",
  year = "1999",
  abstract = "
    We assess the current state of research in the application of computer
    aided formal reasoning to computer algebra, and argue that embedded
    verification support allows users to enjoy its benefits without
    wrestling with technicalities. We illustrate this claim by considering
    symbolic definite integration, and present a verifiable symbolic
    definite integral table look up: a system which matches a query
    comprising a definite integral with parameters and side conditions,
    against an entry in a verifiable table and uses a call to a library of
    lemmas about the reals in the theorem prover PVS to aid in the
    transformation of the table entry into an answer. We present the full
    model of such a system as well as a description of our prototype
    implementation showing the efficacy of such a system: for example, the
    prototype is able to obtain correct answers in cases where computer
    algebra systems [CAS] do not. We extend upon Fateman's web-based table
    by including parametric limits of integration and queries with side
    conditions.",
  paper = "Adam99.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Adam01,
  author = "Adams, Andrew A. and Dunstan, Martin and Gottlieben, Hanne and
            Kelsey, Tom and Martin, Ursula and Owre, Sam",
  title = {{Computer Algebra meets Automated Theorem Proving: Integrating
           Maple and PVS}},
  booktitle = "Theorem proving in higher order logics",
  series = "TPHOLs 2001",
  year = "2001",
  location = "Edinburgh, Scotland",
  pages = "27-42",
  abstract =
    "We describe an interface between version 6 of the Maple computer
    algebra system with the PVS automated theorem prover. The interface is
    designed to allow Maple users access to the robust and checkable proof
    environment of PVS. We also extend this environment by the provision
    of a library of proof strategies for use in real analysis. We
    demonstrate examples using the interface and the real analysis
    library. These examples provide proofs which are both illustrative and
    applicable to genuine symbolic computation problems.",
  paper = "Adam01.pdf",
  keywords = "axiomref, CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Ager88,
  author = "Ager, Tryg A. and Ravaglia, R.A. and Dooley, Sam",
  title = {{Representation of Inference in Computer Algebra Systems with
           Applications to Intelligent Tutoring}},
  year = "1988",
  abstract =
    "Presently computer algebra systems share with calculators the
    property that a sequence of computations is not a unified
    computational sequence, thereby allowing fallacies to occur. We argue
    that if computer algebra systems operate in a framework of strict
    mathematical proof, fallacies are eliminated. We show that this is
    possible in a working interactive system REQD. We explain why
    computational algebra, done under the strict constraints of proof, is
    relevant to uses of computer algebra systems in instruction."
}
```

———-

— **axiom.bib** —

```
@phdthesis{Aker93,
  author = "Akers, Robert Lawrence",
  title = {{Strong Static Type Checking for Functional Common Lisp}},
  school = "Univerity of Texas at Austin",
  year = "1995",
  link = "\url{ftp://www.cs.utexas.edu/pub/boyer/cli-reports/096.pdf}",
  abstract =
    "This thesis presents a type system which supports the strong static
    type checking of programs developed in an applicative subset of the
    Common Lisp language. The Lisp subset is augmented with a guard
    construct for function definitions, which allows type restrictions to
    be placed on the arguments. Guards support the analysis and safe use
    of partial functions, like CAR, which are well-defined only for
    arguments of a certain type.

    A language of type descriptors characterizes the type
    domain. Descriptors are composed into function signatures which
    characterize the guard and which map various combinations of actual
    parameter types to possible result types. From a base of signatures
    for a small collection of primitive functions, the system infers
    signatures for newly submitted functions.

    The system includes a type inference algorithm which handles
    constructs beyond the constraints of ML-style systems. Most notable
    are the free use of CONS to construct objects of undeclared type and
    the use of IF forms whose two result components have unrelated types,
    resulting in ad hoc polymorphism. Accordingly, the type descriptor
    language accommodates disjunction, unrestricted CONS, recursive type
    forms, and ad hoc polymorphic function signatures. As with traditional
    type inference systems, unification is a central algorithm, but the
    richness of our type language complicates many component algorithms,
```

```
    including unification. Special treatment is given to recognizer
    functions, which are predicates determining whether an object is of a
    particular type. Type inference in this setting is undecidable, so the
    algorithm is heuristic and is not complete.

    The semantics of the system are in terms of a function which
    determines whether values satisfy descriptors with respect to a
    binding of type variables. The soundness of each signature emitted by
    the system is validated by a signature checker, whose properties are
    formally specified with respect to the formal semantics and proven to
    hold. The checker algorithm is substantially simpler than the
    inference algorithm, as it need not perform operations such as
    discovering closed recursive forms. Thus, its proof is both more
    straightforward to construct and easier to validate than a direct
    proof of the inference algorithm would be.",
  paper = "Aker93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Aran05,
  author = "Aransay, Jesus and Ballarin, Clemens and Rubio, Julio",
  title = {{Extracting Computer Algebra Programs from Statements}},
  booktitle = "Int. Conf. on Computer Aided System Theory",
  pages = "159-168",
  year = "2005",
  abstract =
    "In this paper, an approach to synthesize correct programs from
    specifications is presented. The idea is to extract code from
    definitions appearing in statements which have been mechanically
    proved with the help of a proof assistant. This approach has been
    found when proving the correctness of certain Computer Algebra
    programs (for Algebraic Topology) by using the Isabelle proof
    assistant. To ease the understanding of our techniques, they are
    illustrated by means of examples in elementary arithmetic.",
  paper = "Aran05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Arch93,
  author = "Archer, M. and Fink, G. and Yang, L.",
  title = {{Linking Other Theorem Provers to HOL Using PM: Proof Manager}},
  booktitle = "Int Workshop on the HOL Theorem Proving System and its
               Applications",
  publisher = "North-Holland",
```

```
  year = "1993"
}
```

———-

— **axiom.bib** —

```
@article{Aude02,
  author = "Audemard, Gilles and Bertoli, Piergiorgio and Cimatti,
            Alessandro and Kornilowicz, Artur and Sebastiani,
    Roberto",
  title = {{Integrating Boolean and Mathematical Solving: Foundations,
           Basic Algorithms, and Requirements}},
  journal = "LNCS",
  volume = "2385",
  pages = "231-245",
  year = "2002",
  abstract =
    "In the last years we have witnessed an impressive advance in the
    efficiency of boolean solving techniques, which has brought large
    previously intractable problems at the reach of state-of-the-art
    solvers.  Unfortunately, simple boolean expressions are not expressive
    enough for representing many real-world problems, which require
    handling also integer or real values and operators. On the other
    hand, mathematical solvers, like computer-algebra systems or
    constraint solvers, cannot handle efficiently problems involving
    heavy boolean search, or do not handle them at all. In this paper we
    present the foundations and the basic algorithms for a new class of
    procedures for solving boolean combinations of mathematical
    propositions, which combine boolean and mathematical solvers, and we
    highlight the main requirements that boolean and mathematical solvers
    must fulfill in order to achieve the maximum benefits from their
    integration. Finally we show how existing systems are captured by our
    framework.",
  paper = "Aude02.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Augu98,
  author = "Augustsson, Lennart",
  title = {{Cayenne -- a language with dependent types}},
  booktitle = "ICFP 98",
  year = "1998",
  pages = "239-250",
  isbn = "1-58113-024-4",
  link = "\url{http://fsl.cs.illinois.edu/images/5/5e/Cayenne.pdf}",
  abstract =
    "Cayenne is a Haskell-like language. The main difference between
    Haskell and Cayenne is that Cayenne has dependent types i.e.
```

```
    the result type of a function may depend on the argument value
    and types of record components which can be types or values
    may depend on other components. Cayenne also combines the
    syntactic categories for value expressions and type expressions thus
    reducing the number of language concepts.

    Having dependent types
    and combined type and value expressions makes the language very
    powerful. It is powerful enough that a special module concept
    is unnecessary; ordinary records suffice. It is also powerful enough to
    encode predicate logic at the type level allowing types to be
    used as specifications of programs. However this power comes at a
    cost: type checking of Cayenne is undecidable. While this may
    appear to be a steep price to pay it seems to work well in practice.",
  paper = "Augu98.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Avig12a,
  author = "Avigad, Jeremy",
  title = {{Type Inference in Mathematics}},
  journal = "European Association of Theoretical Computer Science",
  volume = "106",
  pages = "78-98",
  year = "2012",
  abstract =
    "In the theory of programming languages, type inference is the process
    of inferring the type of an expression automatically, often making use
    of information from the context in which the expression appears. Such
    mechanisms turn out to be extremely useful in the practice of
    interactive theorem proving, whereby users interact with a
    computational proof assistant to construct formal axiomatic
    derivations of mathematical theorems. This article explains some of
    the mechanisms for type inference used by the Mathematical
    Components project, which is working towards a verification of the
    Feit-Thompson theorem.",
  paper = "Avig12a.pdf",
  keywords = "coercion, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Avig14g,
  author = "Avigad, Jeremy",
  title = {{LEAN proof of GCD}},
  link =
 "\url{http://github.com/leanprover/lean2/blob/master/library/data/nat/gcd.lean}",
```

```
  year = "2014",
  paper = "Avig14g.txt",
  keywords = "CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Avig16,
  author = "Avigad, Jeremy",
  title = {{LEAN github repository}},
  link = "\url{http://github.com/leanprover}",
  year = "2016"
}
```

———-

— **axiom.bib** —

```
@misc{Avig17,
  author = "Avigad, Jeremy and de Moura, Leonardo and Kong, Soonho",
  title = {{Theorem Proving in Lean}},
  year = "2017",
  link = "\url{https://leanprover.github.io/tutorial/tutorial.pdf}",
  paper = "Avig17.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Avig17c,
  author = "Avigad, Jeremy and Holzl, Johannes and Serafin, Luke",
  title = {{A Formally Verified Proof of the Central Limit Theorem}},
  year = "2017",
  abstract =
    "We describe a proof of the Central Limit Theorem that has been
    formally verified in the Isabelle proof assistant.  Our formalization
    builds upon and extends Isabelles libraries for analysis and
    measure-theoretic probability.  The proof of the theorem uses
    characteristic functions , which are a kind of Fourier transform, to
    demonstrate that, under suitable hypotheses, sums of random
    variables converge weakly to the st andard normal distribution.  We
    also discuss the libraries and infrastructure that supported the
    formalization, and reflect on some of the lessons we have learned
    from the effort.",
  paper = "Avig17c.pdf",
  keywords = "printed"
}
```

———-

1.19.2 B

— **axiom.bib** —

```
@techreport{Back73,
  author = "Backus, John",
  title = {{Programming Language Semantics and Closed Applicative Language}},
  type = "technical report",
  institution = "IBM",
  number = "RJ 1245",
  year = "1973"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Ball95,
  author = "Ballarin, Clemens and Homann, Karsten and Calmet, Jacques",
  title =
    {{Theorems and Algorithms: An Interface between Isabelle and Maple}},
  booktitle = "ISSAC 95",
  year = "1995",
  pages = "150-157",
  publisher = "ACM",
  link = "\url{https://pdfs.semanticscholar.org/077e/606f92b4095637e624a9efc942c5c63c4bc2.pdf}",
  abstract =
    "Solving sophisticated mathematical problems often requires algebraic
    algorithms {\sl and} theorems. However, there are no environments
    integrating theorem provers and computer algebra systems which
    consistently provide the inference capabilities of the first and the
    powerful arithmetic of the latter systems.

    As an example for such a mechanized mathematics environment we describe
    a prototype implementation of an interface between Isabelle and Maple.
    It is achieved by extending the simplifier of Isabelle through the
    introduction of a new class of simplification rules called evaluation
    rules in order to make selected operations of Maple available, and
    without any modification to the computer algebra system. Additionally,
    we specify syntax translations for the concrete syntax of Maple
    which enables the communication between both systems illustrated by
    some examples that can be solved by theorems and algorithms",
  paper = "Ball95.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Ball98,
  author = "Ballarin, Clemens and Paulson, Lawrence C.",
  title = {{Reasoning about Coding Theory: The Benefits We Get from
           Computer Algebra}},
  year = "1998",
  link = "\url{http://www21.in.tum.de/~ballarin/publications/aisc98.pdf}",
  abstract =
    "The use of computer algebra is usually considered beneficial for
    mechanised reasoning in mathematical domains. We present a case study,
    in the application domain of coding theory, that supports this claim:
    the mechanised proof depends on non-trivial algorithms from computer
    algebra and increase the reasoning power of the theorem prover. The
    unsoundness of computer algebra systems is a major problem in
    interfacing them to theorem provers. Our approach to obtaining a sound
    overall system is not blanket distrust but based on the distinction
    between algorithms we call sound and {\sl ad hoc} respectively. This
    distinction is blurred in most computer algebra systems OUr
    experimental interface therefore uses a computer algebra library. It
    is based on theorem templates, which provide formal specifications for
    the algorithms.",
  paper = "Ball98.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@article{Ball99,
  author = "Ballarin, Clemens and Paulson, Lawrence C.",
  title = {{A Pragmatic Approach to Extending Provers by Computer Algebra --
           with Applications to Coding Theory}},
  journal = "Fundam. Inform.",
  volume = "39",
  number = "1-2",
  pages = "1-20",
  year = "1999",
  link = "\url{http://www.cl.cam.ac.uk/~lp15/papers/Isabelle/coding.pdf}",
  abstract = "
    The use of computer algebra is usually considered beneficial for
    mechanised reasoning in mathematical domains. We present a case study,
    in the application domain of coding theory, that supports this claim:
    the mechanised proofs depend on non-trivial algorithms from computer
    algebra and increase the reasoning power of the theorem prover.

    The unsoundness of computer algebra systems is a major problem in
    interfacing them to theorem provers. Our approach to obtaining a sound
    overall system is not blanket distrust but based on the distinction
    between algorithms we call sound and {\sl ad hoc} respectively. This
    distinction is blurred in most computer algebra systems. Our
    experimental interface therefore uses a computer algebra library. It
    is based on formal specifications for the algorithms, and links the
```

```
    computer algebra library Sumit to the prover Isabelle.

    We give details of the interface, the use of the computer algebra
    system on the tactic-level of Isabelle and its integration into proof
    procedures.",
  paper = "Ball99.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

— axiom.bib —

```
@phdthesis{Ball99a,
  author = "Ballerin, Clemens",
  title = {{Computer Algebra and Theorem Proving}},
  school = "Darwin College, University of Cambridge",
  year = "1999",
  abstract =
    "Is the use of computer algebra technology beneficial for
    mechanised reasoining in and about mathematical domains? Usually
    it is assumed that it is. Many works in this area, however, either
    have little reasoning conent, or use symbolic computation only to
    simplify expressions. In work that has achieved more, the used
    methods do not scale up. They trust the computer algebra system
    either too much or too little.

    Computer algebra systems are not as rigorous as many provers. They
    are not logically sound reasoning systems, but collections of
    algorithms. We classify soundness problems that occur in computer
    algebra systems. While many algorithms and their implementations
    are perfectly trustworthy, the semantics of symbols is often
    unclear and leads to errors. On the other hand, more robust
    approaches to interface external reasoners to provers are not
    always practical because the mathematical depth of proofs
    algorithms in computer algebra are based on can be enormous.

    Our own approach takes both trustworthiness of the overall system
    and efficiency into account. It relies on using only reliable
    parts of a computer algebra system, which can be achieved by
    choosing a suitable library, and deriving specifications for these
    algorithms from their literature.

    We design and implement an interface between the prover Isabelle
    and the computer algebra library Sumit and use it to prove
    non-trivial theorems from coding theory. This is based on the
    mechanisation of the algebraic theories of rings and
    polynomials. Coding theory is an area where proofs do have a
    substantial amount of computational content. Also, it is realistic
    to assume that the verification of an encoding or decoding device
    could be undertaken in, and indeed, be simplified by, such a
    system.
```

```
    The reason why semantics of symbols is often unclear in current
    computer algebra systems is not mathematical difficulty, but the
    design of those systems. For Gaussian elimination we show how the
    soundness problem can be fixed by a small extension, and without
    losing efficiency. This is a prerequisite for the efficient use of
    the algorithm in a prover.",
  paper = "Ball99a.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@article{Bare01,
  author = "Barendregt, Henk and Cohen, Arjeh M.",
  title = {{Electronic Communication of Mathematics and the Interaction
           of Computer Algebra Systems and Proof Assistants}},
  journal = "J. Symbolic Computation",
  volume = "32",
  pages = "3-22",
  year = "2001",
  abstract =
    "Present day computer algebra systems (CASs) and proof assistants
    (PAs) are specialized programs that help humans with mathematical
    computations and deductions. Although several such systems are
    impressive, they all have certain limitations. In most CASs side
    conditions that are essential for the truth of an equality are not
    formulated; moreover there are bugs. The PAs have a limited power for
    computing and hence also for assistance with proofs.  Almost all
    examples of both categories are stand alone special purpose systems
    and therefore they cannot communicate with each other.

    We will argue that the present state of the art in logic is such that
    there is a natural formal language, independent of the special purpose
    application in question, by which these systems can communicate
    mathematical statements. In this way their individual power will be
    enhanced.

    Statements received at one particular location from other sites fall
    into two categories: with or without the qualification ‘‘evidently
    impeccable’’, a notion that is methodologically precise and
    sound. For statements having this quality assessment the evidence may
    come from the other site or from the local site itself, but in both
    cases it is verified locally.  In cases where there is no evidence of
    impeccability one has to rely on cross checking. There is a trade-off
    between these two kinds of statements: for impeccability one has to
    pay the price of obtaining less power.

    Some examples of communication forms are given that show how the
    participants benefit",
  paper = "Bare01.pdf",
  keywords = "CAS-Proof, printed, DONE"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Baue16,
  author = "Bauer, Andrej",
  title = {{Five Stages of Accepting Constructive Mathematics}},
  year = "2016",
  journal = "Bull. of the American Mathematical Society",
  link = "\url{http://dx.doi.org/10.1090/bull/1556}",
  abstract =
    "On the odd day, a mathematician might wonder what constructive
    mathematics is all about. They may have heard arguments in favor of
    constructivism but are not at all convinced by them, and in any case
    they may care little about philosophy. A typical introductory text
    about constructivism spends a great deal of time explaining the
    principles and contains only trivial mathematics, while advanced
    constructive texts are impenetrable, like all unfamiliar mathematics.
    How then can a mathematician find out what constructive mathematics
    feels like? What new and relevant ideas does constructive mathematics
    have to offer, if any? I shall attempt to answer these questions.",
  paper = "Baue16.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Baum90,
  author = "Baumgartner, Gerald and Stansifer, Ryan",
  title = {{A Proposal to Study Type Systems for Computer Algebra}},
  institution = "RISC",
  year = "1990",
  type = "technical report",
  number = "90-07.0",
  abstract =
    "It is widely recognized that programming languages should offer
    features to help structure programs. To achieve this goal, languages
    like ADA, MODULA-2, object-oriented languages, and functional
    languages have been developed. The structuring techniques available so
    far (like modules, classes, parametric polymorphism) are still not
    enough or not appropriate for some application areas. In symbolic
    computation, in particular computer algebra, several problems occur
    that are difficult to handle with any existing programming language.
    Indeed, nearly all available computer algebra systems suffer from the
    fact that the underlying programming language imposes too many
    restricitons.

    We propose to develop a language that combines the essential features
    from functional language, object-oriented languages, and computer
```

```
    algebra systems in a semantically clean manner. Although intended for
    use in symbolic computation, this language should prove interesting as
    a general purpose programming language. The main innovation will be
    the application of sophisticated type systems to the needs of computer
    algebra systems. We will demonstrate the capabilities of the language
    by using it to implement a small computer algebra library. This
    implementation will be compared against a straightforward Lisp
    implementation and against existing computer algebra systems. Our
    development should have an impact both on the programming languages
    world and on the computer algebra world.",
  paper = "Baum90.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InCollection{Bees89,
  author = "Beeson, Michael J.",
  title = {{Logic and Computation in MATHPERT: An Expert System for
            Learning Mathematics}},
  booktitle = "Computers and Mathematics",
  publisher = "Springer-Verlag",
  year = "1989",
  pages = "299-307",
  isbn = "0-387-97019-3"
}
```

———-

— **axiom.bib** —

```
@article{Bees89a,
  author = "Beeson, Michael",
  title = {{Some Applications of Gentzen's Proof Theory in Automated
            Deduction}},
  journal = "LNCS",
  volume = "475",
  pages = "101-156",
  year = "1989",
  abstract =
    "We show that Prolog is intimately connected with Gentzen's cut-free
    sequent calculus G, analyzing Prolog computations as the construction
    of certain cut-free derivations. We introduce a theorem-proving
    program GENTZEN based on Gentzen's sequent calculus, which
    incorporates some features of Prolog's computation procedure. We show
    that GENTZEN has the following properties: (1) It is
    (non-deterministically) sound and complete for first-order
    intuitionistic predicate calculus; (2) Its successful computations
    coincide with those of Prolog on the Horn clause fragment (both
    deterministically and non-deterministically). The proofs of (1) and
```

```
    (2) contain a new proof of the completeness of (non-deterministic)
    Prolog for Horn clause logic, based on our analysis of Prolog in terms
    of Gentzen sequents instead of on resolution. GENTZEN has been
    implemented and tested on examples including some proofs by induction
    in number theory, an example constructed by J. McCarthy to show the
    limitations of Prolog, and ‘‘Schubert’s Steamroller.’’ An extension of
    GENTZEN also provides a decision procedure for intuitionistic
    propositional calculus (but at some cost in efficiency).",
  paper = "Bees89a.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@article{Bees92,
  author = "Beeson, M.",
  title = {{Mathpert: Computer support for learning algebra, trig, and
            calculus}},
  journal = "LNCS",
  volume = "624",
  pages = "454-456",
  year = "1992",
  abstract =
    "Mathpert is a computerized environment for learning algebra, trig,
    and one-variable calculus. It permits students to direct the
    step-by-step solution of problems, and is capable of helping them
    by solving the problems itself if necessary.",
  paper = "Bees92.pdf",
  keywords = "printed, DONE"
}
```

———-

— axiom.bib —

```
@article{Bees95,
  author = "Beeson, Michael",
  title = {{Using Nonstandard Analysis to Ensure the Correctness of
            Symbolic Computations}},
  journal = "Int. J. of Foundations of Computer Science",
  volume = "6",
  number = "3",
  pages = "299-338",
  year = "1995",
  abstract =
    "Nonstandard analysis has been put to use in a theorem-prover,
    where it assists in the analysis of formulae involving limits. The
    theorem-prover in question is used in the computer program
    Mathpert to ensure the correctness of calculations in
    calculus. Although non-standard analysis is widely viewed as
```

```
    non-constructive, it can alternately be viewed as a method of
    reducing logical manipulation (of formulae with quantifiers) to
    coputation (with rewrite rules). We give a logical theory of
    nonstandard analysis which is implemented in Mathpert. We describe
    a procedure for 'elimination of infinitesimals' (also implemented
    in Mathpert) and prove its correctness.",
  paper = "Bees95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bees98,
  author = "Beeson, Michael",
  title = {{Automatic Generation of Epsilon-Delta Proofs of Continuity}},
  link = "\url{http://www.michaelbeeson.com/research/papers/aisc.pdf}",
  year = "1998",
  abstract =
    "As part of a project on automatic generation of proofs involving both
    logic and computation, we have automated the production of some proofs
    involving epsilon-delta arguments. These proofs involve two or three
    quantifiers on the logical side, and on the computational side, they
    involve algebra, trigonometry, and some calculus. At the border of
    logic and computation, they involve several types of arguments
    involving inequalities, including transitivity chaining and several
    types of bounding arguments, in which bounds are sought that do not
    depend on certain variables. Control mechanisms have been developed
    for intermixing logical deduction steps with computational steps and
    with inequality reasoning. Problems discussed here as examples involve
    the continuity and uniform continuity of various specific functions.",
  paper = "Bees98.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bees02,
  author = "Beeson, Michael and Wiedijk, Freek",
  title = {{The Meaning of Infinity in Calculus and Computer Algebra
            Systems}},
  journal = "LNCS",
  volume = "2385",
  year = "2002",
  abstract =
    "We use filters of open sets to provide a semantics justifying the
    use of infinity in informal limit calculations in calculus, and in
    the same kind of calculations in computer algebra. We compare the
    behavior of these filters to the way Mathematica behaves when
    calculating with infinity.
```

```
    We stress the need to have a proper semantics for computer algebra
    expressions, especially if one wants to use results and methods
    from computer algebra in theorem provers. The computer algebra
    method under discussion in this paper is the use of rewrite rules
    to evaluate limits involving infinity.",
  paper = "Bees02.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Bees06,
  author = "Beeson, Michael",
  title = {{Mathematical Induction in Otter-Lambda}},
  journal = "J. Automated Reasoning",
  volume = "36",
  number = "4",
  pages = "311'344",
  year = "2006",
  abstract =
    "Otter-lambda is Otter modified by adding code to implement an
    algorithm for lambda unification. Otter is a resolution-based,
    clause-language first-order prover that accumulates deduced clauses
    and uses strategies to control the deduction and retention of
    clauses. This is the first time that such a first-order prover has
    been combined in one program with a unification algorithm capable of
    instantiating variables to lambda terms to assist in the
    deductions. The resulting prover has all the advantages of the
    proof-search algorithm of Otter (speed, variety of inference rules,
    excellent handling of equality) and also the power of lambda
    unification. We illustrate how these capabilities work well together
    by using Otter-lambda to find proofs by mathematical induction. Lambda
    unification instantiates the induction schema to find a useful
    instance of induction, and then Otter's first-order reasoning can be
    used to carry out the base case and induction step. If necessary,
    induction can be used for those, too. We present and discuss a variety
    of examples of inductive proofs found by Otter-lambda: some in pure
    Peano arithmetic, some in Peano arithmetic with defined predicates,
    some in theories combining algebra and the natural numbers, some
    involving algebraic simplification (used in the induction step) by
    simplification code from MathXpert, and some involving list induction
    instead of numerical induction. These examples demonstrate the
    feasibility and usefulness of adding lambda unification to a
    first-order prover.",
  papers = "Bees06.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bees14,
  author = "Beeson, M.",
  title = {{Mixing Computations and Proofs}},
  comment = "slides",
  year = "2014",
  link = "\url{http://www.cs.ru.nl/qed20/slides/beeson-qed20.pdf}",
  paper = "Bees14.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bees16,
  author = "Beeson, M.",
  title = {{Mixing Computations and Proofs}},
  journal = "J. of Formalized Reasoning",
  volume = "9",
  number = "1",
  pages = "71-99",
  year = "2016",
  link =
 "\url{http://www.michaelbeeson.com/research/papers/ProofAndComputation.pdf}",
  abstract =
    "We examine the relationship between proof and computation in
    mathematics, especially in formalized mathematics.  We compare the
    various approaches to proofs with a significant computational
    component, including (i) verifying the algorithms, (ii) verifying the
    results of the unverified algo- rithms, and (iii) trusting an external
    computation.",
  paper = "Bees16.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Benk03,
  author = "Benke, Marcin and Dybjer, Peter and Jansson, Patrik",
  title = {{Universes for generic programs and proofs in dependent type
           theory}},
  journal = "Nordic Journal of Computing",
  volume = "10",
  year = "2003",
  pages = "265-269",
  link = "\url{http://www.cse.chalmers.se/~peterd/papers/generic.html}",
  abstract =
    "We show how to write generic programs and proofs in {Martin L\"of}
    type theory. To this end we considier several extensions of
```

```
    {Martin-L\"of}'s logical framework for dependent types. Each extension
    has a universe of codes (signatures) for inductively defined sets with
    generic formation, introduction, elimination, and equality
    rules. These extensions are modeled on Dybjer and Setzer's finitely
    axiomatized theories of inductive-recursive definitions, which also
    have universese of codes for sets, and generic formation,
    introduction, elimination, and equality rules. Here we consider
    several smaller universes of interest for generic programming and
    universal algebra. We formalize one-sorted and many-sorted term
    algebras, as well as iterated, generalized, parameterized, and indexed
    inductive definitions. We also show how to extend the techniques of
    generic programming to these universes. Furthermore, we give generic
    proofs of reflexivity and substitutivity of a generic equality
    test. Most of the definitions in the paper have been implemented using
    the proof assistant Alfa for dependent type theory.",
  paper = "Benk03.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Benz01,
  author = "Benzmuller, Christoph and Jamnik, Mateja and Kerber, Manfred
            and Sorge, Volker",
  title = {{An Agent-oriented Approach to Reasoning}},
  booktitle = "Proc. Calculemus Workshop 2001",
  pages = "48-63",
  link = "\url{http://christoph.benzmueller.di/papers/W11.pdf}",
  year = "2001",
  abstract =
    "This paper discusses experiments with an agent oriented approach to
    automated and interactive reasoning. The approach combines ideas from
    two subfields of AI (theorem proving/proof planning and multi-agent
    systems) and makes use of state of the art distribution techniques to
    decentralise and spread its reasoning agents over the internet. It
    particularly supports cooperative proofs between reasoning systems
    which are strong in different application areas, e.g., higher-order
    and first-order theorem provers and computer algebra systems.",
  paper = "Benz01.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Bert04,
  author = {Bertot, Yves Cast\'eran, Pierre},
  title = {{Interactive Theorem Proving and Program Development}},
  publisher = "Springer",
  year = "2004",
  isbn = "3-540-20854-2",
```

```
  abstract = "
    Coq is an interactive proof assistant for the development of
    mathematical theories and formally certified software. It is based on
    a theory called the calculus of inductive constructions, a variant of
    type theory.

    This book provides a pragmatic introduction to the development of
    proofs and certified programs using Coq. With its large collection of
    examples and exercies it is an invaluable tool for researchers,
    students, and engineers interested in formal methods and the
    development of zero-fault software.",
  keywords = "shelf"
}
```

———-

— axiom.bib —

```
@book{Bido91,
  author = "Bidoit, M. and Kreowski, H.-J. and Lescanne, P. and
            Orejas, F. and Sannella, D.",
  title = {{Algebraic System Specification and Development}},
  publisher = "Springer-Verlag",
  comment = "LNCS 501",
  year = "1991",
  isbn = "3-540-54060-1",
  abstract =
    "A great deal of work has been devoted to methods of specification
    based on the simple idea that a functional program can be modelled as
    a {\sl many-sorted algebra}, i.e. as a number of sets of data values
    (one set of values for each data type) together with a number of total
    functions on those sets corresponding to the functions in the
    program. This abstracts away from the algorithms used to compute the
    functions and how those algorithms are expressed in a given
    programming language, focusing instead on the representation of data
    and the input/output behavior of functions. The pioneering work in
    this area is [Zil 74],[Gut 75], [GTW 76], of which the latter -- the
    so-called {\sl initial algebra approach} -- is the most formal This
    idea was soon taken up by other workers, see e.g. [GGM 76], [GHM 76],
    [BG 77], [GHM 78]. Today the field of algebraic specification has
    grown into one of the major areas of research in theoretical computer
    science. More than fifteen years of research have led to an abundance
    of competing and complementary theories and approaches. The algebraic
    approach provides a conceptual basis, theoretical fundations, and
    prototype tools for the stepwise formal development of correct system
    components from their specifications, and thus covers the whole
    software development process from the specification of requirements to
    the finished system. These methos are potentially applicable to the
    development of correct hardware systems as well",
  keywords = "axiomref, CAS-Proof, shelf"

}
```

———-

— **axiom.bib** —

```
@article{Bled74,
  author = "Bledsoe, W.W. and Bruell, Peter",
  title = {{A Man-Machine Theorem-Proving System}},
  journal = "Artificial Intelligence",
  volume = "5",
  number = "1",
  pages = "51-72",
  year = "1974",
  abstract =
    "This paper describes a man-machine theorem-proving system at the
    University of Texas at Austin which has been used to prove a few
    theorems in general topology. The theorem (or subgoal) being proved is
    presented on the scope in its natural form so that the user can easily
    comprehend it and, by a series of interactive commands, can help with
    the proof when he desires. A feature called DETAIL is employed which
    allows the human to interact only when needed and only to the extent
    necessary for the proof.

    The program is built around a modified form of IMPLY, a
    natural-deduction-like theorem proving technique which has been
    described earlier.",
  paper = "Bled74.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Bled79,
  author = "Bledsoe, W.W. and Bruell, P. and Shostak, R.",
  title = {{A Prover for General Inequalities}},
  type = "technical report",
  institution = "Univ. of Texas at Austin",
  year = "1979",
  number = "ATP-40A"
}
```

———-

— **axiom.bib** —

```
@techreport{Bled83,
  author = "Bledsoe, W.W.",
  title = {{The UT Natural Deduction Prover}},
  type = "technical report",
  institution = "Univ. of Texas at Austin",
  year = "1983",
  number = "ATP-17B"
```

```
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bled84,
  author = "Bledsoe, W.W.",
  title = {{Some Automatic Proofs in Analysis}},
  booktitle = "Automated Theorem Proving: After 25 Years",
  publisher = "American Mathematical Society",
  year = "1984"
}
```

———-

— **axiom.bib** —

```
@misc{Bold07,
  author = "Boldo, Sylvie and Filliatre, Jean-Christophe",
  title = {{Formal Verification of Floating-Point programs}},
  link = "\url{http://www-lipn.univ-paris13.fr/CerPAN/files/ARITH.pdf}",
  paper = "Bold07.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Bold07a,
  author = "Boldo, Sylvie and Filliatre, Jean-Christophe",
  title = {{Formal Verification of Floating-Point programs}},
  link = "\url{http://www.lri.fr/~filliatr/ftp/publis/caduceus-floats.pdf}",
  abstract =
    "This paper introduces a methodology to perform formal verification of
    floating-point C programs.  It extends an existing tool for
    verification of C programs, Caduceus, with new annotations for
    specific floating-point arithmetic. The Caduceus first-order logic
    model for C programs is extended accordingly. Then verification
    conditions are obtained in the usual way and can be discharged
    interactively with the Coqa proof assistant, using an existing Coq
    formalization of floating-point arithmetic. This methodology is
    already implemented and has been successfully applied to several short
    floating-point programs, which are presented in this paper.",
  paper = "Bold07a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bold11,
  author = "Boldo, Sylvie and Marche, Claude",
  title = {{Formal verification of numerical programs: from C annotated
           programs to mechanical proofs}},
  year = "2011",
  publisher = "Springer",
  journal = "Mathematics in Computer Science",
  volume = "5",
  pages = "377-393",
  link = "\url{https://hal.archives-ouvertes.fr/hal-00777605/document}",
  abstract =
    "Numerical programs may require a high level of guarantee. This can be
    achieved by applying formal methods, such as machine-checked proofs.
    But these tools handle mathematical theorems while we are interested
    in C code, in which numerical computations are performed using
    floating-point arithmetic, whereas proof tools typically handle exact
    real arithmetic. To achieve this high level of confidence on C programs,
    we use a chain of tools: Frama-C, its Jessie plugin, Why and provers
    among Coq, Gappa, Alt-Ergo, CVC3 and Z3. This approach requires the C
    program to be annotated; each function must be precisely specified, and
    we prove the correctness of the program by proving both that it meets its
    specifications and that no runtime error may occur. The purpose of this
    paper is to illustrate, on various examples, the features of this
    approach.",
  paper = "Bold11.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Bold16,
  author = "Boldo, Sylvie",
  title = {{Formal verification of numerical analysis programs}},
  year = "2016",
  link = "\url{https://www.youtube.com/watch?v=7MDwpwD6Ts4}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Boni07,
  author = "Bonichon, R. and Delahaye, D. and Doligez, D.",
  title = {{Zenon: An Extensible Automated Theorem Prover Producing
           Checkable Proofs}},
  booktitle = "LPAR 2007",
  year = "2007",
  link = "\url{http://zenon.inria.fr/zenlpar07.pdf}",
  abstract =
    "We present Zenon, an automated theorem prover for first order
    classical logic (with equality), based on the tableau method. Zenon is
```

```
    intended to be the dedicated prover of the Focal environment, an
    object-oriented algebraic specification and proof system, which is
    able to produce OCaml code for execution and Coq code for
    certification. Zenon can directly generate Coq proofs (proof scripts
    or proof terms), which can be reinserted in the Coq specifications
    produced by Focal. Zenon can also be extended, which makes specific
    (and possibly local) automation possible in Focal.",
  paper = "Boni07,pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Boul00,
  author = "Boulme, S. and Hardin, T. and Rioboo, R.",
  title = {{Polymorphic Data Types, Objects, Modules and Functors,:
           is it too much?}},
  link = "\url{ftp://ftp.lip6.fr/lip6/reports/2000/lip6.2000.014.ps.gz}",
  abstract = "
    Abstraction is a powerful tool for developers and it is offered by
    numerous features such as polymorphism, classes, modules, and
    functors, $\ldots$ A working programmer may be confused by this
    abundance. We develop a computer algebra library which is being
    certificed. Reporting this experience made with a language (Ocaml)
    offering all these features, we argue that the are all needed
    together. We compare several ways of using classes to represent
    algebraic concepts, trying to follow as close as possible mathematical
    specification. Then we show how to combine classes and modules to
    produce code having very strong typing properties.  Currently, this
    library is made of one hundred units of functional code and behaves
    faster than analogous ones such as Axiom.",
  paper = "Boul00.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Boul00a,
  author = "Boulme. Sylvain",
  title = {{Specifying in Coq inheritance used in Computer Algebra
            Libraries}},
  year = "2000",
  institution = "LIP6, Paris",
  number = "2000.013",
  abstract =
    "This paper is part of FOC[3], a project for developing Computer
    Algebra libraries, certified in Coq[2]. FOC has developed a
    methodology for programming Computer Algebra libraries, using modules
    and objects in Ocaml. In order to specify modularity features used by
```

```
    FOC in Ocaml, we are coding in Coq a theory for extensible records
    with dependent fields. This theory intends to express especially the
    kind of inheritance with method redefinition and late binding, that
    FOC uses in its Ocaml programs.

    The unit of FOC are coded as records. As we want to encode sematic
    information on units, the fields of our records may be proofs. Thus,
    our fields may depend on each others. We call the
    {\sl Drecords}. Then, we introduce a new datatype, called
    {\sl mixDrec}, to represent FOC classes. Actually, mixDrecs are useful
    for describing a hierarchy of Drecords in an incremental way. In
    mixDrecs, fields can be only declared or they can be
    redefined. MixDrecs can be extended by inheritance.",
  paper = "Boul00a.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Boul99,
  author = "Boulme, S. and Hardin, T. and Hirschkoff, D. and Rioboo, Renaud",
  title = {{On the way to certify Computer Algebra Systems}},
  booktitle = "Systems for integrated computation and deduction",
  series = "Calculemus 99",
  year = "1999",
  publisher = "Elsevier",
  location = "Trento, Italy",
  pages = "11-12",
  abstract =
    "The FOC project aims at supporting, within a coherent software system,
    the entire process of mathematical computation, starting with proved
    theories, ending with certified implementations of algorithms. In this
    paper, we explain our design requirements for the implementation,
    using polynomials as a running example. Indeed, proving correctness of
    implementations depends heavily on the way this design allows
    mathematical properties to be truly handled at the programming level.

    The FOC project, started at the fall of 1997, is aimed to build a
    programming environment for the development of certified symbolic
    computation. The working languages are Coq and Ocaml. In this paper,
    we present first the motivations of the project. We then explain why
    and how our concern for proving properties of programs has led us to
    certain implementation choices in Ocaml. This way, the sources express
    exactly the mathematical dependencies between different structures.
    This may ease the achievement of proofs.",
  paper = "Boul99.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Boul94,
  author = "Boulton, Richard John",
  title = {{Efficiency in a fully-expansive Theorem Prover}},
  year = "1994",
  type = "technical report",
  number = "UCAM-CL-TR-337",
  institution = "University of Cambridge",
  abstract =
    "The HOL system is a fully-expansive theorem prover: Proofs generated
    in the system are composed of applications of the primitive inference
    rules of the underlying logic. This has two main advantages. First,
    the soundness of the system depends only on the implementations of the
    primitive rules. Second, users can be given the freedom to write their
    own proof procedures without the risk of making the system unsound. A
    full functional programming language is provided for this purpose. The
    disadvantage with the approach is that performance is
    compromised. This is partly due to the inherent cost of fully
    expanding a proof but, as demonstrated in this thesis, much of the
    observed inefficiency is due to the way the derived proof procedures
    are written. This thesis seeks to identify sources of non-inherent
    inefficiency in the HOL system and proposes some general-purpose and
    some specialised techniques for eliminating it. One area that seems to
    be particularly amenable to optimisation is equational reasoning. This
    is significant because equational reasoning constitutes large portions
    of many proofs. A number of techniques are proposed that transparently
    optimise equational reasoning. Existing programs in the HOL system
    require little or no modification to work faster. The other major
    contribution of this thesis is a framework in which part of the
    computation involved in HOL proofs can be postponed. This enables
    users to make better use of their time. The technique exploits a form
    of lazy evaluation. The critical feature is the separation of the code
    that generates the structure of a theorem from the code that justifies
    it logically. Delaying the justification allows some non-local
    optimisations to be performed in equational reasoning. None of the
    techniques sacrifice the security of the fully-expansive approach. A
    decision procedure for a subset of the theory of linear arithmetic is
    used to illustrate many of the techniques. Decision procedures for
    this theory are commonplace in theorem provers due to the importance
    of arithmetic reasoning. The techniques described in the thesis have
    been implemented and execution times are given. The implementation of
    the arithmetic procedure is a major contribution in itself. For the
    first time, users of the HOL system are able to prove many arithmetic
    lemmas automatically in a practical amount of time (typically a second
    or two). The applicability of the techniques to other fully-expansive
    theorem provers and possile extensions of the ideas are considered."
}
```

———-

— **axiom.bib** —

```
@article{Bove05,
  author = "Bove, Ana and Capretta, Venanzio",
  title = {{Modelling General Recursion in Type Theory}},
  journal = "Math. Struct. in Comp. Science",
  volume = "15",
  pages = "671-708",
  year = "2005",
  abstract =
    "Constructive type theory is an expressive programming language in
    which both algorithms and proofs can be represented. A limitation of
    constructive type theory as a programming language is that only
    terminating programs can be defined in it. Hence, general recursive
    algorithms have no direct formalisation in type theory since they
    contain recursive calls that satisfy no syntactic condition
    guaranteeing termination. In this work, we present a method to
    formalise general recursive algorithms in type theory. Given a general
    recursive algorithm, our method is to define an inductive
    special-purpose accessibility predicate that characterises the inputs
    on which the algorithm terminates. The type-theoretic version of the
    algorithm is then defined by structural recursion on the proof that
    the input values satisfy this predicate.  The method separates the
    computational and logical parts of the definitions and thus the
    resulting type-theoretic algorithms are clear, compact and easy to
    understand. They are as simple as their equivalents in a functional
    programming language, where there is no restriction on recursive
    calls. Here, we give a formal definition of the method and discuss its
    power and its limitations.",
  paper = "Bove05.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@misc{Bove08,
  author = "Bove, Ana and Dybjer, Peter",
  title = {{Dependent Types at Work}},
  year = "2008",
  comment = "Lecture notes from LerNET Summer School, Piriapolis",
  link =
    "\url{http://www.cse.chalmers.se/~peterd/papers/DependentTypesAtWork.pdf}",
  abstract =
    "In these lecture notes we give an introduction to functional
    programming with dependent types. We use the dependently typed
    programming language Agda which is an extension of {Martin-L\"of} type
    theory. First we show how to do simply typed functional programming in
    the style of Haskell and ML. Some differences between Agda's type
    system and the Hindley-Milner type system of Haskell and ML are also
    discussed. Then we show how to use dependent types for programming and
    we explain the basic ideas behind type-checking dependent types. We go
    on to explain the Curry-Howard identification of propositions and
    types. This is what makes Agda a programming logic and not only a
```

```
    programming language. According to Curry-Howard, we identify programs
    and proofs, something which is possible only by requiring that all
    programs terminate. However, at the end of these notes we present a
    method for encoding partial and general recursive functions as total
    functions using dependent types.",
  paper = "Bove08.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Boye81,
  author = "Boyer, Bob and Kaufmann, Matt and Moore, J",
  title = {{Metafunctions in Nqthm and Acl2}},
  link = "\url{https://www.cs.utexas.edu/users/boyer/meta-nqthm-acl2.text}",
  year = "1981"
}
```

———-

— **axiom.bib** —

```
@techreport{Boye85,
  author = "Boyer, Robert S. and Moore, J Strother",
  title = {{Integrating Decision Procedures into Heuristic Theorem Provers}},
  type = "technical report",
  institution = "Inst. for Comp. Sci. University of Texas at Austin",
  number = "ICSCA-CMP-44",
  year = "1985",
  abstract =
    "We discuss the problem of incorporating into a heuristic theorem
    prover a decision procedure for a fragment of the logic.  An obvious
    goal when incorporating such a procedure is to reduce the search space
    explored by the heuristic component of the system, as would be
    achieved by eliminating from the systems data base some explicitly
    stated axioms.  For example, if a decision procedure for linear
    inequalities is added, one would hope to eliminate the explicit
    consideration of the transitivity axioms.  However, the decision
    procedure must then be used in all the ways the eliminated axioms
    might have been.  The difficulty of achieving this degree of
    integration is more dependent upon the complexity of the heuristic
    component than upon that of the decision procedure.  The view of the
    decision procedure as a ``black box'' is frequently destroyed by the
    need pass large amounts of search strategic information back and forth
    between the two components.  Finally, the efficiency of the decision
    procedure may be virtually irrelevant; the efficiency of the final
    system may depend most heavily on how easy it is to communicate
    between the two components.  This paper is a case study of how we
    integrated a linear arithmetic procedure into a heuristic theorem
    prover.  By linear arithmetic here we mean the decidable subset of
```

```
    number theory dealing with universally quantified formulas composed of
    the logical connectives, the identity relation, the Peano ‘‘less than’’
    relation, the Peano addition and subtraction functions, Peano
    constants, and variables taking on natural values.  We describe our
    system as it originally stood, and then describe chronologically the
    evolution of our linear arithmetic procedure and its interface to the
    heuristic theorem prover.  We also provide a detailed description of
    our final linear arithmetic procedure and the use we make of it.  This
    description graphically illustrates the difference between a
    stand-alone decision procedure and one that is of use to a more
    powerful theorem prover.",
  paper = "Boye85.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Boye88,
  author = "Boyer, R.S. and Moore, J.S.",
  title = {{A Computational Logic Handbook}},
  publisher = "Academic Press",
  year = "1988",
  isbn = "0-12-122952-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Brad07,
  author = "Bradley, Aaron R. and Manna, Zohar",
  title = {{The Calculus of Computation}},
  year = "2007",
  publisher = "Springer",
  isbn = "978-3-540-74112-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Bres93,
  author = "Bressoud, David",
  title = {{Review of The problems of mathematics}},
  journal = "Math. Intell.",
  volume = "15",
  number = "4",
  year = "1993",
```

```
  pages = "71-73"
}
```

———-

— **axiom.bib** —

```
@article{Brow12,
  author = "Brown, Christopher W.",
  title = {{Fast simplifications for Tarski formulas based on monomial
           inequalities}},
  year = "2012",
  journal = "Journal of Symbolic Computation",
  volume = "47",
  pages = "859-882",
  abstract =
    "We define the combinatorial part of a Tarski formula in which
    equalities and inequalities are in factored or partially-factored
    form. The combinatorial part of a formula contains only
    monomial inequalities,which are sign conditions on monomials.  We
    give efficient algorithms for answering some basic questions about
    conjunctions of monomial inequalities and prove the
    NP-Completeness/Hardness of some others. By simplifying the
    combinatorial part of a Tarski formula, and mapping the simplified
    combinatorial part back to a Tarski formula, we obtain non-trivial
    simplifications without algebraic operations.",
  paper = "Brow12.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Broy88,
  author = "Broy, Manfried",
  title = {{Equational Specification of Partial Higher-order Algebras}},
  journal = "Theoretical Computer Science",
  volume = "57",
  number = "1",
  year = "1988",
  pages = "3-45",
  abstract =
    "The theory of algebraic abstract types specified by positive
    conditional formulas formed of equations and a definedness predicate
    is outlined and extended to hierarchical types with ‘‘noustrict’’
    operations, partial and even infinite objects. Its model theory is
    based on the concept of partial interpretations. Deduction rules are
    given, too. Models of types are studied where all explicit equations
    have solutions. The inclusion of nigher-order types, i.e., types
    comprising higher-order functions leads to an algebraic (‘‘equational’’)
    specification of algebras including sorts with ‘‘infinite’’ objects and
    higher-order functions (‘‘functionals’’).",
```

```
  paper = "Broy88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Brui68,
  author = "de Bruijn, N.G.",
  title = {{The Mathematical Language Automath, its Usage, and Some of
           its Extensions}},
  journal = "Lecture Notes in Mathematics",
  publisher = "Springer",
  year = "1994",
  volume = "125",
  paper = "Brui68.pdf",
  keywords = "CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@techreport{Brui68a,
  author = "de Bruijn, N.G.",
  title = {{AUTOMATH, A Language for Mathematics}},
  year = "1968",
  type = "technical report",
  number = "68-WSK-05",
  institution = "Technische Hogeschool Eindhoven",
  paper = "Brui68a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Buch97,
  author = "Buchberger, Bruno",
  title = {{Mathematica: doing mathematics by computer?}},
  journal = "Advances in the design of symbolic computation systems",
  year = "1997",
  publisher = "Springer-Verlag",
  pages = "2-20",
  isbn = "978-3-211-82844-1",
  paper = "Buch97.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Buch01,
  author = "Buchberger, Bruno",
  title = {{Theorema: A Proving System Based on Mathematica}},
  journal = "The Mathematica Journal",
  volume = "8",
  number = "2",
  pages = "247-252",
  year = "2001",
  paper = "Buch01.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Buch00,
  author = "Buchberger, B. and Dupre, C. and Jebelean, T. and Kriftner, F.
            and Nakagawa, K. and Vasaru, D. and Windsteiger, W.",
  title = {{The Theorema Project: A Progress Report}},
  year = "2000",
  abstract =
    "The THEOREMA project aims at supporting, within one consistent logic
    and one coherent software system, the entire mathematical exploration
    cycle including the phase of proving. In this paper we report on some
    of the new features of THEOREMA that have been designed and
    implemented since the first expository version of THEOREMA in
    1997. These features are: - the THEOREMA formal text language - the
    THEOREMA computational sessions - the Prove-Compute-Solve (PCS) prover
    of THEOREMA - the THEOREMA set theory prover - special provers within
    THEOREMA - the cascade-meta-strategy for THEOREMA provers - proof
    simplification in THEOREMA. In the conclusion, we formulate design
    goals for the next version of THEOREMA",
  paper = "Buch00.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Buch06,
  author = "Buchberger,B and Craciun, A. and Jebelean, T. and
            Kovacs, L. and Kutsia, T. and Nakagawa, K. and Piroi, F.
            and Popov, N. and Robu, J. and Rosenkranz, M. and
            Windsteiger, W.",
  title = {{Theorema: Towards Computer-Aided Mathematical Theory
            Exploration}},
  journal = "J. of Applied Logic",
  volume = "4",
  number = "4",
```

```
  pages = "470-504",
  year = "2006",
  abstract =
    "Theorema is a project that aims at supporting the entire process of
    mathematical theory exploration within one coherent logic and software
    system. This survey paper illustrates the style of Theorema-supported
    mathematical theory exploration by a case study (the automated
    synthesis of an algorithm for the construction of Groebner Bases) and
    gives an overview on some reasoners and organizational tools for
    theory exploration developed in the Theorema project.",
  paper = "Buch06.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Buch03,
  author = "Buchberger, Bruno and Aigner, K. and Dupre, C. and
            Jebelean, T. and Kriftner, F. and Marin, M. and
            Nakagawa, K. and Podisor, O. and Tomuta, E. and
            Usenko, Y. and Vasaru, D. and Windsteiger, W.",
  title = {{Theorema: An Integrated System for Computation and
            Deduction in Natural Style}},
  abstract =
    "The Theorema project aims at integrating computation and deduction in
    a system that can be used by the working scientist for building and
    checking mathematical models, including the design and verication of
    new algorithms. Currently, the system uses the rewrite engine of the
    computer algebra system Mathematica for building and combining a
    number of automatic/interactive provers (high-order predicate-logic,
    induction for lists/tuples and natural numbers, etc.) in natural
    deduction style and in natural language presentation. These provers
    can be used for dening and proving properties of mathematical models
    and algorithms, while a specially provided ‘‘computing engine’’ can
    execute directly the logical description of these algorithms.",
  paper = "Buch03.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Buch16,
  author = "Buchberger, Bruno and Jebelean, Tudor and Kutsia, Temur
            and Maletzky, Alexander and Windsteiger, Wolfgang",
  title = {{Theorema 2.0: Computer-Assisted Natural-Style Mathematics}},
  journal = "J. of Formalized Reasoning",
  volume = "9",
  number = "1",
```

```
  pages = "149-155",
  year = "2016",
  abstract =
    "The Theorema project aims at the development of a computer assistant
    for the working mathematician. Support should be given throughout all
    phases of mathematical activity, from introducing new mathematical
    concepts by definitions or axioms, through first (computational)
    experiments, the formulation of theorems, their justification by an
    exact proof, the application of a theorem as an algorithm, until to
    the dissemination of the results in form of a mathematical
    publication, the build up of bigger libraries of certified
    mathematical content and the like. This ambitious project is exactly
    along the lines of the QED manifesto issued in 1994 and it was
    initiated in the mid-1990s by Bruno Buchberger. The Theorema system is
    a computer implementation of the ideas behind the Theorema
    project. One focus lies on the natural style of system input (in form
    of definitions, theorems, algorithms, etc.), system output (mainly in
    form of mathematical proofs) and user interaction. Another focus is
    theory exploration, i.e. the development of large consistent
    mathematical theories in a formal frame, in contrast to just proving
    single isolated theorems. When using the Theorema system, a user
    should not have to follow a certain style of mathematics enforced by
    the system (e.g. basing all of mathematics on set theory or certain
    variants of type theory), rather should the system support the user in
    her preferred flavour of doing math. The new implementation of the
    system, which we refer to as Theorema 2.0, is open-source and
    available through GitHub.",
  paper = "Buch16.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Bulo04,
  author = {Medina-Bulo, I. and Palomo-Lozano, F. and Alonso-Jim\'enez, J.A.
            and Ruiz-Reina, J.L.},
  title = {{Verified Computer Algebra in ACL2}},
  journal = "ASIC 2004, LNAI 3249",
  year = "2004",
  pages = "171-184",
  abstract = "In this paper, we present the formal verification of a
    Common Lisp implementation of Buchberger's algorithm for computing
    Groebner bases of polynomial ideals. This work is carried out in the
    ACL2 system and shows how verified Computer Algebra can be achieved
    in an executable logic.",
  paper = "Bulo04.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bulo10,
  author = "Medina-Bulo, I. and Palomo-Lozano, F. and Alonso-Jim\'enez, J.A.
           and Ruiz-Reina, J.L.",
  title = {{A verified Common Lisp implementation of Buchberger's algorithm
           in ACL2}},
  journal = "Journal of Symbolic Computation",
  year = "2010",
  pages = "96-123",
  abstract = "In this article, we present the formal verification of a
    Common Lisp implementation of Buchberger's algorithm or computing
    Groebner bases of polynomial ideals. This work is carried out in ACL2,
    a system which provices an integrated environment where programming
    (in a pure functional subset of Commmon Lisp) and formal verification
    of programs, with the assistance of a theorem prover, are possible. Our
    imlementation is written in a real programming language and it is
    directly executable within the ACL2 system or any compliant Common Lisp
    system. We provide here snippets o real verified code, discuss the
    formalization details in depth, and present quantitative data about
    the proof effort.",
  paper = "Bulo10.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bund75,
  author = "Bundy, Alan and Wallen, Lincoln",
  title = {{The UT Theorem Prover}},
  booktitle = "Catalogue of Artificial Intelligence Tools",
  pages = "132-133",
  year = "1975",
  abstract =
    "The UT theorem prover is probably the best known natural deduction
    <153> theorem prover. It was written in LISP <34> by woody Bledsoe and
    his co-workers at the University of Texas, and is best described in
    (Bledsoe and Tyson 75]. The theorem prover embodies a Gentzen-like
    deduction system for first-order predicate calculus, and many special
    purpose techniques, including: subgoaling, rewrite rules, controlled
    lorward chaining, controlled definition instantiation, conditional
    procedures, and induction. The prover, though powerful in its own
    right, is essentially interactive and thus allows the user of the
    prover to control the search for the proof in radical ways. The user
    can for example : add hypotheses, instruct the prover to instantiate
    certain variables with values, or instruct the prover as to which
    deduction rule to use next."
}
```

———-

— **axiom.bib** —

```
@article{Bund88,
  author = "Bundy, Alan",
  title = {{The Use of Explicit Plans to Guide Inductive Proofs}},
  journal = "LNCS 310",
  volume = "310",
  pages = "111-120",
  year = "1998",
  abstract =
    "We propose the use of explicit proof plans to guide the search for a
    proof in automatic theorem proving. By representing proof plans as the
    specifications of LCF-like tactics and by recording these
    specifications in a sorted meta-logic, we are able to reason about the
    conjectures to be proved and the methods available to prove them. In
    this way we can build proof plans of wide generality, formally account
    for and predict their successes and failures, apply them flexibly,
    recover from their failures, and learn them from example proofs.

    We illustrate this technique by building a proof plan based on a
    simple subset of the implicit proof plan embedded in the Boyer-Moore
    theorem prover.",
  paper = "Bund88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bund90,
  author = "Bundy, Alan and van Harmelen, Frank and Horn, Christian and
            Smaill, Alan",
  title = {{The Oyster-Clam System}},
  journal = "Lecture Notes in Artificial Intelligence",
  volume = "449",
  year = "1990",
  pages = "647-648",
  paper = "Bund90.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bund93b,
  author = "Bundy, Alan and Stevens, Andrew and van Hemelen, Frank and
            Ireland, Andrew and Smaill, Alan",
  title = {{Rippling: A heuristic for guiding inductive proofs}},
  journal = "Artifical Intelligence",
  volume = "62",
  number = "2",
  year = "1993",
  pages = "185-253",
```

```
  abstract =
    "We describe rippling: a tactic for the heuristic control of the key
    part of proofs by mathematical induction. This tactic significantly
    reduces the search for a proof of a wide variety of inductive
    theorems. We first present a basic version of rippling, followed by
    various extensions which are necessary to capture larger classes of
    inductive proofs. Finally, we present a generalised form of rippling
    which embodies these extensions as special cases. We prove that
    generalised rippling always terminates, and we discuss the
    implementation of the tactic and its relation with other inductive
    proof search heuristics.",
  paper = "Bund93b.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Bund05,
  author = "Bundy, Alan and Basin, David and Hutter, Dieter and
            Ireland, Andrew",
  title = {{Rippling: Meta-Level Guidance for Mathematical
            Reasoning}},
  publisher = "Cambridge University Press",
  year = "2005",
  isbn = "978-0-521-83449-0",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Byrd17,
  author = "Byrd, William",
  title = {{The Most Beautiful Program Ever Written}},
  link = "\url{https://www.youtube.com/watch?v=OyfBQmvr2Hc}",
  comment = "See miniKanren and Barliman (program synthesis with proof)"
}
```

———-

1.19.3 C

— **axiom.bib** —

```
@inproceedings{Calm95,
  author = "Calmet, Jacques and Homann, Karsten",
  title = {{Distributed Mathematical Problem Solving}},
  booktitle = "4th Bar-Han SYmp. on Foundations of Artificial Intelligence",
```

```
  pages = "220-230",
  year = "1995",
  abstract =
    "Coupling computer algebra systems and theorem provers enables to
    extend the capabilities they have when standing alone. We report on an
    ongoing research project whose long term goal is to provide an open
    environment for doing mathematics including reasoners and symbolic
    calculators. It is extensible by users which can construct complex
    systems by combination and insertion of existing packages. These
    systems may be based on different logics, formalisms, data structures,
    interfaces. A result of this work is illustrated by a prototype
    implementation of an interface between Isabelle and Maple.",
  paper = "Calm95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Calm96a,
  author = "Calmet, Jacques and Homann, Karsten",
  title = {{Proofs in Computational Algebra: An Interface between DTP
           and Magma}},
  booktitle = "Proc. 2nd Magma Conf. on Computational Algebra",
  year = "1996",
  comment = "Extended Abstract",
  link = "\url{https://pdfs.semanticscholar.org/c709/338dfde245e638690bc7414d8a191eae3a82.pdf}",
  paper = "Calm96a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Calm97a,
  author = "Calmet, Jacques and Homann, Karsten",
  title = {{Towards the Mathematics Software Bus}},
  journal = "Theoretical Computer Science",
  volume = "197",
  number = "1-2",
  year = "1997",
  pages = "221-230",
  abstract =
    "The Mathematics Software Bus is a software environment for combining
    heterogeneous systems performing any kind of mathematical
    computation. Such an environment will provide combinations of
    graphics, editing and computation tools through interfaces to already
    existing powerful software by flexible and powerful semantically
    integration.

    Communication and cooperation mechanisms for logical and symbolic
```

```
    computation systems enable to study and solve new classes of problems
    and to perform efficient computation in mathematics through
    cooperating specialized packages.

    We give an overview on the need for cooperation in solving
    mathematical problems and illustrate the advantages by several
    well-known examples. The needs and requirements for the Mathematics
    Software Bus and its architecture are demonstrated through some
    implementations of powerful interfaces between mathematical services.",
  paper = "Calm97a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Cann05,
  author = "Cannon, John and Bosma, Wieb",
  title = {{Handbook of Magma Functions}},
  year = "2005",
  publisher = "University of Sydney, School of Math and Statistics"
}
```

———-

— **axiom.bib** —

```
@article{Capr01,
  author = "Caprotti, Olga and Oostdijk, Martijn",
  title = {{Formal and Efficient Primality Proofs by Use of Computer
           Algebra Oracles}},
  journal = "J. Symbolic Computation",
  volume = "32",
  pages = "55-70",
  year = "2001",
  abstract =
    "This paper focuses on how to use Pocklington's criterion to
    produce efficient formal proof-objects for showing primality of
    large positive numbers. First, we describe a formal development of
    Pocklington's criterion, done using the proof assistant Coq. Then
    we present an algorithm in which computer algebra software is
    employed as oracle to the proof assistant to generate the
    necessary witnesses for applying the criterion. Finally, we
    discuss the implementation of this approach and tackle the proof
    of primality for some of the largest numbers expressible in Coq.",
  paper = "Capr01.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Cast16,
  author = "Casteran, Pierre and Sozeau, Mattieu",
  title = {{A Gentle Introduction to Type Classses and Relations in Coq}},
  year = "2016",
  link = "\url{http://www.labri.fr/perso/casteran/CoqArt/TypeClassesTut/typeclassestut.pdf}",
  paper = "Cast16.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Care08,
  author = "Carette, Jacques and Farmer, William M.",
  title = {{High-Level Theories}},
  journal = "LNCS",
  volume = "5144",
  pages = "232-245",
  year = "2008",
  abstract =
    "We introduce high-level theories in analogy with high-level
    programming languages. The basic point is that even though one can
    define many theories via simple, low-level axiomatizations , that is
    neither an effective nor a comfortable way to work with such theories.
    We present an approach which is closer to what users of mathematics
    employ, while still being based on formal structures.",
  paper = "Care08.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Care09,
  author = "Carette, Jacques and Farmer, William M.",
  title = {{A Review of Mathematical Knowledge Management}},
  journal = "LNCS",
  volume = "5625",
  pages = "233-246",
  year = "2009",
  abstract =
    "Mathematical Knowledge Management (MKM), as a field, has seen
    tremendous growth in the last few years. This period was one where
    many research threads were started and the field was defining
    itself. We believe that we are now in a position to use the MKM body
    of knowledge as a means to define what MKM is, what it worries about,
    etc. In this paper, we review the literature of MKM and gather various
    metadata from these papers. After offering some definitions
    surrounding MKM, we analyze the metadata we have gathered from these
    papers, in an effort to cast more light on the field of MKM and its
```

```
    evolution",
  paper = "Care09.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@article{Care10,
  author = "Carette, Jacques",
  title = {{Mechanized Mathematics}},
  journal = "LNCS",
  volume = "6167",
  year = "2010",
  pages = "157-157",
  abstract =
    "In the 50 years since McCarthys 'Recursive Functions of Symbolic
    Expressions and Their Computation by Machine', what have we
    learned about the realization of Leibnizs dream of just being
    able to utter 'Calculemus!' (Let us calculate!) when faced with a
    mathematical dilemma?

    In this talk, I will first present what I see as the most
    important lessons from the past which need to be heeded by modern
    designers. From the present, I will look at the context in which
    computers are used, and derive further requirements. In
    particular, now that computers are no longer the exclusive
    playground for highly educated scientists, usability is now more
    important than ever, and justifiably so.

    I will also examine what I see as some principal failings of
    current systems, primarily to understand some major mistakes to
    avoid. These failings will be analyzed to extract what seems to be
    the root mistake, and I will present my favourite solutions.

    Furthermore, various technologies have matured since the creation
    of many of our systems, and whenever appropriate, these should be
    used. For example, our understanding of the structure of
    mathematics has significantly increased, yet this is barely
    reflected in our libraries. The extreme focus on efficiency by the
    computer algebra community, and correctness by the (interactive)
    theorem proving community should no longer be considered viable
    long term strategies.  But how does one effectively bridge that
    gap?

    I personally find that a number of (programming) language-based
    solutions are particularly effective, and I will emphasize
    these. Solutions to some of these problems will be illustrated
    with code from a prototype of MathScheme 2.0, the system I am
    developing with Bill Farmer and our research group.",
  paper = "Care10.pdf",
  keywords = "printed, DONE"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Care10a,
  author = "Carette, Jacques and Sexton, Alan P. and Sorge, Volker and
            Watt, Stephen M.",
  title = {{Symbolic Domain Decomposition}},
  journal = "LNCS",
  volume = "6167",
  year = "2010",
  abstract =
    "Decomposing the domain of a function into parts has many uses in
    mathematics. A domain may naturally be a union of pieces, a
    function may be defined by cases, or different boundary conditions
    may hold on different regions. For any particular problem the
    domain can be given explicitly, but when dealing with a family of
    problems given in terms of symbolic parameters, matters become
    more difficult. This article shows how hybrid sets, that is
    multisets allowing negative multiplicity, may be used to express
    symbolic domain decompositions in an efficient, elegant and
    uniform way, simplifying both computation and reasoning. We apply
    this theory to the arithmetic of piecewise functions and symbolic
    matrices and show how certain operations may be reduced from
    exponential to linear complexity.",
  paper = "Care10a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Care11a,
  author = "Carette, Jacques and Farmer, William M. and Jeremic, Filip and
            Maccio, Vincent and O'Connor, Russell and Tran, Quang M.",
  title = {{The MathScheme Library: Some Preliminary Experiments}},
  year = "2011",
  link = "\url{https://arxiv.org/pdf/1106.1862.pdf}",
  abstract =
    "We present some of the experiments we have performed to best test our
    design for a library for MathScheme, the mechanized mathematics
    software system we are building. We wish for our library design to use
    and reflect, as much as possible, the mathematical structure present
    in the objects which populate the library.",
  paper = "Care11a.pdf",
  keywords = "axiomref, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Care11b,
  author = "Carette, Jacques and Farmer, William M. and Wajs, Jeremie",
  title = {{Trustable Communication Between Mathematics Systems}},
  year = "2011",
  link = "\url{https://pdfs.semanticscholar.org/0d0b/206edf7ef1c01d7bfa1284c85b469b2fbd29.pdf}",
  abstract =
    "This paper presents a rigorous, unified framework for facilitating
    communication between mathematics systems. A mathematics system is
    given one or more interfaces which offer deductive and computational
    services to other mathematics systems. To achieve communication
    between systems, a client interface is linked to a server interface by
    an asymmetric connection consisting of a pair of translations.
    Answers to requests are trustable in the sense that they are correct
    provided a small set of prescribed conditions are satisfied. The
    frame work is robust with respect to interface extension and can
    process requests for abstract services, where the server interface is
    not fully specified.",
  paper = "Care11b.pdf",
  keywords = "printed"
}
```

————-

— **axiom.bib** —

```
@misc{Care11c,
  author = "Carette, Jacques and Farmer, William M. and O'Connor, Russell",
  title = {{MathScheme: Project Description}},
  year = "2011",
  link = "\url{http://imps.mcmaster.ca/doc/cicm-2011-proj-desc.pdf}",
  paper = "Care11c.pdf",
  keywords = "printed, DONE"
}
```

————-

— **axiom.bib** —

```
@article{Care12,
  author = "Carette, Jacques and O'Connor, Russell",
  title = {{Theory Presentation Combinators}},
  journal = "LNCS",
  volume = "7362",
  year = "2012",
  abstract =
    "We motivate and give semantics to theory presentation combinators
    as the foundational building blocks for a scalable library of
    theories. The key observation is that the category of contexts and
    fibered categories are the ideal theoretical tools for this
    purpose.",
```

```
  paper = "Care12.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Chli10,
  author = "Chlipala, Adam",
  title = {{An Introduction to Programming and Proving with Dependent Types
           in Coq}},
  journal = "Journal of Formalized Reasoning",
  volume = "3",
  number = "2",
  pages = "1-93",
  year = "2010",
  paper = "Chli10.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Chli14,
  author = "Chlipala, Adam and Braibant, Thomas and Cuellar, Santiago and
            Delaware, Benjamin and Gross, Jason and Malecha, Gregory and
            Clement,-Claudel, Pit and Wang, Peng",
  title =
    {{Bedrock: A Software Development Ecosystem Inside a Proof Assistant}},
  year = "2014",
  link = "\url{https://www.youtube.com/watch?v=BSyrp-iYBMo}",
  abstract =
    "The benefits of formal correctness proofs for software are clear
    intuitively, but the high human costs of proof construction have
    generally been viewed as prohibitive. To support that integration, we
    need to rethink the familiar programming toolchains. The new world
    needn't be all about doing prodigious extra work to achieve the virtue
    of correct programs; formal methods also suggest new programming
    approaches that better support abstraction and modularity than do
    coarser-grained specification styles like normal static types. This
    talk overviews Bedrock, a framework for certified programming inside
    of the Coq proof assistant. Bedrock programs are implemented,
    specified, verified, and compiled inside of Coq. A single program may
    be divided into modules with formal interfaces, written in different
    programming languages and verified with different proof styles. The
    common foundation is an assembly language with an operational
    semantics (serving as the trusted code base) and a semantic module
    system (orchestrating linking of code and proofs across source
    languages). A few different programming styles have been connects to
    the shared foundation, including a C-like language with an ``array of
    bytes'' memory model, higher-level more C++-like languages with ``array
```

```
    of abstract data types'' memory models, a domain-specific language for
    XML processing, standard Coq functional programs, and even declarative
    specifications that are refined automatically into assembly code with
    correctness proofs. The talk will present Bedrock's shared foundation
    and sketch the pieces that go into refining declarative specifications
    into closed assembly programs, covering joint work with Thomas
    Braibant, Santiago Cuellar, Benjamin Delaware, Jason Gross, Gregory
    Malecha, Clement Pit-Claudel, and Peng Wang."

}
```

———-

— **axiom.bib** —

```
@book{Chli15,
  author = "Chlipala, Adam",
  title = {{Certified Programming with Dependent Types}},
  year = "2015",
  link = "\url{http://adam.chlipala.net/cpdt/cpdt.pdf}",
  publisher = "MIT Press",
  isbn = "9780262026659",
  paper = "Chli15.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Clar91,
  author = "Clarke, Edmund and Zhao, Xudong",
  title = {{Analytica -- A Theorem Prover in Mathematica}},
  year = "1991",
  link = "\url{http://www.cs.cmu.edu/~emc/papers/Conference%20Papers/Analytica%20A%20Theorem%20Prover%20in%20Mat
  paper = "Clar91.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@article{Clar93,
  author = "Clarke, Edmund and Zhao, Xudong",
  title = {{Analytica -- A Theorem Prover for Mathematica}},
  journal = "The Mathematica Journal",
  volume = "3",
  number = "1",
  year = "1993",
  pages = "761-765",
  abstract =
```

```
    "Analytica is an automatic theorem prover for theorems in
    elementary analysis. The prover is written in Mathematica language
    and runs in the Mathematica environment. The goal of the project
    is to use a powerful symbolic computation system to prove theorems
    that are beyond the scope of previous automatic theorem
    provers. The theorem prover is also able to guarantee the
    correctness of certain steps that are made by the symbolic
    computation system and therefore prevent common errors like
    division by a symbolic expression that could be zero. In this
    paper we describe the structure of Analytica and explain the main
    techniques that it uses to construct proofs. We have tried to make
    the paper as self-contained as possible so that it will be
    accessible to a wide audience of potential users. We illustrate
    the power of our theorem prover by several non-trivial examples
    including the basic properties of the stereographic projection and
    a series of three lemmas that lead to a proof of Weierstrass's
    example of a continuous nowhere differentiable function. Each of
    the lemmas in the latter example is proved completely
    automatically.",
  paper = "Clar93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Clem91,
  author = "Clement, Dominique and Prunet, Vincent and Montagnac, Francis",
  title =
    {{Integrated software components: A Paradigm for Control Integration}},
  journal = "LNCS",
  volume = "509",
  pages = "167-177",
  year = "1991",
  abstract =
    "This report describes how control integration between software
    components may be organised using an encapsulation technique combined
    with broadcast message passing : each software component, which is
    encapsulated within an integrated software component (IC),
    communicates by sending and receiving events. Events are emitted
    without the emitter knowing whether there are any receivers. The
    proposed mechanism can be used for intertool communication as well as
    for communication within a single tool.

    This programming architecture frees the code from dependencies upon
    the effective software components environments, and simplifies its
    extension.",
  paper = "Clem91.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Cohe93,
  author = "Cohen, Arjeh M. and Davenport, James H. and Heck, J.P.",
  title = {{An overview of computer algebra}},
  year = "1993",
  paper = "Cohe93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Como88,
  author = "Comon, H. and Lugiez, D. and Schnoebelen, P.H.",
  title = {{A Rewrite-based Type Discipline for a Subset of Computer Algebra}},
  journal = "J. Symbolic Computation",
  year = "1991",
  volume = "11",
  pages = "349-368",
  abstract =
    "This paper is concerned with the type structure of a system including
    polymorphism, type properties, and subtypes. This type system
    originates from computer algebra but it is not intended to be the
    solution of all type problems in this area.

    Types (or sets of types) are denoted by terms in some order-sorted
    algebra. We consider a rewrite relation in this algebra, which is
    intended to express subtyping. The relations between the semantics and
    the axiomatization are investigated. It is shown that the problem of
    type inference is undecidable but that a narrowing strategy for
    semi-decision procedures is described and studied.",
  paper = "Como88.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Como89,
  author = "Common, Hubert and Lescanne, Pierre",
  title = {{Equational Problems and Disunification}},
  journal = "J. Symbolic Computation",
  volume = "7",
  number = "3-4",
  year = "1989",
  pages = "371-425",
  abstract =
    "Roughly speaking, an equational problem is a first order formula
    whose only predicate symbol is $=$. We propose some rules for the
    transformation of equational problems and study their correctness in
```

```
    various models. Then, we give completeness results with respect to
    some 'simple' problems called solved forms. Such completeness results
    still hold when adding some control which moreover ensures
    termination. The termination proofs are given for a 'weak' control and
    thus hold for the (large) class of algorithms obtained by restricting
    the scope of the rules. Finally, it must be noted that a by-product of
    our method is a decision procedure for the validity in the Herbrand
    Universe of any first order formula with the only predicate symbol $=$.",
  paper = "Como89.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Cons85,
  author = "Constable, R.L. and Allen, S.F. and Bromley, H.M. and Cremer, J.F.
            and Harper, R.W. and Howe, D.J. and Knoblock, T.B. and
            Mendler, N.P. and Panagaden, P. and Tsaaki, J.T. and Smith, S.F.",
  title = {{Implementing Mathematics with The Nuprl Proof Development System}},
  publisher = "Prentice-Hall",
  year = "1985",
  paper = "Cons85.pdf",
  isbn = "9781468059106",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@techreport{Coqu86,
  author = "Coquand, Thierry",
  title = {{An Analysis of Girard's Paradox}},
  year = "1986",
  institution = "INRIA Centre de Rocquencourt",
  number = "531",
  abstract =
    "We study the consistency of a few formal systems specially some
    extensions of Church's calculus and the construction system. We
    show that Church's calculus is not compatible with the notion of
    second-order type. We apply this result for showing that the
    calculus of construction wit four levels is inconsistent. We
    suggest finally some consistent extensions of these two calculi.",
  paper = "Coqu86.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Coqu86a,
  author = {Coquand, Thierry and Huet, G\'erard},
  title = {{The Calculus of Constructions}},
  year = "1986",
  institution = "INRIA Centre de Rocquencourt",
  number = "530",
  link = "\url{https://hal.inria.fr/inria-00076024/document}",
  abstract =
    "The Calculus of Constructions is a higher-order formalism for
    constructive proofs in natural deduction style. Every proof is a
    $\lambda$-expression, typed with propositions of the underlying
    logic. By removing types we get a pure $\lambda$-expression,
    expressing its associated algorithm. Computing this
    $\lambda$-expression corresponds roughly to cut-elimination. It is our
    thesis that (as already advocated by Martin-Lof) the Curry-Howard
    correspondence between propositions and types is a powerful paradigm
    for Computer Science. In the case of Constructions, we obtain the
    notion of a very high-level functional programming language, with
    complex polymorphism well-suited for modules specification. The notion
    of type encompasses the usual notioin of data type, but allows as well
    arbitrarily complex algorithmic specifications. We develop the basic
    theory of a Calculus of Constructions, and prove a strong
    normalization theorem showing that all computations terminate.
    Finally, we suggest various extensions to stronger calculi.",
  paper = "Coqu86a.pdf"
}
```

———-

— axiom.bib —

```
@article{Coqu93,
  author = "Coquand, Thierry",
  title = {{Infinite Objects in Type Theory}},
  journal = "LNCS",
  volume = "806",
  year = "1993",
  pages = "62-78",
  abstract =
    "We show that infinite objects can be constructively understood
    without the consideration of partial elements, or greatest fixed-
    points, through the explicit consideration of proof objects.  We
    present then a proof system based on these explanations.  According to
    this analysis, the proof expressions should have the same structure
    as the program expressions of a pure functional lazy language:
    variable, constructor, application, abstraction, case expressions,
    and local let expressions.",
  paper = "Coqu93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Coqu96,
  author = "Coquand, Thierry and Dybjer, Peter",
  title = {{Intuitionistic Model Constructions and Normalization Proofs}},
  journal = "Mathematical Structures in Computer Science",
  volume = "7",
  pages = "75-94",
  year = "1996",
  link = "\url{http://www.cse.chalmers.se/~peterd/papers/Glueing.ps}",
  abstract =
    "The traditional notions of {\sl strong} and {\sl weak normalization}
    refer to properties of a binary {\sl reduction relation}. In this
    paper we explore an alternative approach to normalization, where we
    bypass the reduction relation and instead focus on the
    {\sl normalization function}, that is, the function which maps a term into
    its normal form.

    Such a normalization function can be constructed by building an
    appropriate model and a function ``quote'' which inverts the
    interpretation function. The normalization function is then obtained
    by composing the quote function with the interpretation function. We
    also discuss how to gt a simple proof of the property that
    constructors are one-to-one, which usually is obtained as a corollary
    of Church-Rosser and normalization in the traditional sense.

    We illustrate this approach by showing how a glueing model (closely
    related to the glueing construction used in category theory) gives
    rise to a normalization algorithm for a combinatory formlation of
    Godel System T. We then show how the method extends in a
    straightforward way when we add cartesian products and disjoint unions
    (full intuitionistic propositional logic under a Curry-Howard
    interpretation) and transfinite inductive types such as the Brouwer
    ordinals.",
  paper = "Coqu96.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Coqu96a,
  author = "Coquand, Thierry",
  title = {{An Algorithm for Type-Checking Dependent Types}},
  year = "1996",
  abstract =
    "We present a simple type-checker for a language with dependent
    types and let expressions, with a simple proof of correctness.",
  paper = "Coqu96a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Coqu16,
  author = {Coquand, Thierry and Huet, G\'erard and Paulin, Christine},
  title = {{The COQ Proof Assistant}},
  year = "2016",
  link = "\url{https://coq.inria.fr}"
}
```

———-

— **axiom.bib** —

```
@misc{COQR16,
  author = {Coquand, Thierry and Huet, G\'erard and Paulin, Christine},
  title = {{The COQ Proof Assistant Reference Manual}},
  year = "2016",
  link="\url{https://coq.inria.fr/distrib/current/files/Reference-Manual.pdf}",
  paper = "COQR16.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Coqu16a,
  author = {Coquand, Thierry and Huet, G\'erard and Paulin, Christine},
  title = {{COQ Proof Assistant Library Coq.ZArith.Znumtheory}},
  year = "2016",
  link = "\url{https://coq.inria.fr/library/Coq.ZArith.Znumtheory.html}"
}
```

———-

— **axiom.bib** —

```
@misc{COQnat,
  author = "COQ Proof Assistant",
  title = {{Library Coq.Init.Nat}},
  link = "\url{https://coq.inria.fr/library/Coq.Init.Nat.html}",
  abstract = "Peano natural numbers, defintions of operations",
  year = "2017"
}
```

———-

— **axiom.bib** —

```
@misc{Cons98,
  author = "Constable, Robert L. and Jackson, Paul B.",
  title = {{Towards Integrated Systems for Symbolic Algebra and Formal
           Constructive Mathematics}},
  link = "\url{http://www.nuprl.org/documents/Constable/towardsintegrated.pdf}",
  year = "1998",
  abstract =
    "The purpose of this paper is to report on our efforts to give a
    formal account of some of the algebra used in Computer Algebra Systems
    (CAS). In particular, we look at the concepts used in the so called
    3rd generation algebra systems, such as Axiom[4] and Weyl[9].  It is
    our claim that the Nuprl proof development system is especially well
    suited to support this kind of mathematics.",
  paper = "Cons98.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@book{Crol93,
  author = "Crole, R.L.",
  title = {{Categories for Types}},
  publisher = "Cambridge University Press",
  year = "1993",
  isbn = "978-0521457019"
}
```

———-

1.19.4 D

— **axiom.bib** —

```
@misc{Daly10,
  author = "Daly, Timothy",
  title = {{Intel Instruction Semantics Generator}},
  link = "\url{http://daly.axiom-developer.org/TimothyDaly_files/publications/sei/intel/intel.pdf}",
  abstract =
    "Given an Intel x86 binary, extract the semantics of the instruction
    stream as Conditional Concurrent Assignments (CCAs). These CCAs
    represent the semantics of each individual instruction. They can be
    composed to represent higher level semantics.",
  paper = "Daly10.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Dani06,
  author = "Danielsson, Nils Anders and Hughes, John and Jansson, Patrik and
            Gibbons, Jeremy",
  title = {{Fast and Loose Reasoning is Morally Correct}},
  booktitle = "Proc. of ACM POPL '06",
  series = "POPL '06",
  year = "2006",
  location = "Charleston, South Carolina",
  abstract =
    "Functional programmers often reason about programs as if they were
    written in a total language, expecting the results to carry over to
    non-toal (partial) languages. We justify such reasoning.

    Two languages are defined, one total and one partial, with identical
    syntax. The semantics of the partial language includes partial and
    infinite values, and all types are lifted, including the function
    spaces. A partial equivalence relation (PER) is then defined, the
    domain of which is the total subset of the partial language. For types
    not containing function spaces the PER relates equal values, and
    functions are related if they map related values to related values.

    It is proved that if two closed terms have the same semantics in the
    total language, then they have related semantics in the partial
    language. It is also shown that the PER gives rise to a bicartesian
    closed category which can be used to reason about values in the domain
    of the relation.",
  paper = "Dani06.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Dave93a,
  author = "Davenport, James H.",
  title = {{C9: Universal Algebra}},
  year = "1993",
  comment = "Lecture Notes for 2nd year undergrad and mather's course
             in Universal Algebra",
  school = "University of Bath"
}
```

———-

— **axiom.bib** —

```
@misc{Dave98,
  author = "Davenport, James H.",
  title = {{Is Computer Algebra the same as Computer Mathematics?}},
  year = "1998",
  comment = "Talk for British Colloquium for Theoretical Computer Science"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Dave08,
  author = "Davenport, James H.",
  title = {{Effective Set Membership in Computer Algebra and Beyond}},
  journal = "LNAI",
  volume = "5144",
  pages = "266-269",
  year = "2008",
  abstract =
    "In previous work, we showed the importance of distinguishing ``I know
    that $X \ne Y$'' from ``I don't know that $X = Y$''.  In this paper we
    look at effective set membership, starting with Groebner bases, where
    the issues are well-expressed in algebra systems, and going on to
    integration and other questions of 'computer calculus'.

    In particular, we claim that a better recognition of the role of set
    membership would clarify some features of computer algebra systems,
    such as 'what does an integral mean as output'.",
  paper = "Dave08.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave08a,
  author = "Davenport, James H.",
  title = {{AISC Meets Natural Typography}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
  abstract =
    "McDermott [12,13] introduced the concept 'Artificial Intelligence
    meets Natural Stupidity'. In this paper, we explore how Artificial
    Intelligence and Symbolic Computation can meet Natural Typography, and
    how the conventions for expressing mathematics that humans
    understand can cause us difficulties when designing mechanised
    systems.",
  paper = "Dave08a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Dave17a,
```

```
  author = "Davenport, James",
  title = {{Computer Algebra and Formal Proof}},
  year = "2017",
  comment = "BPR presentation, Cambridge, England",
  video = "Dave17a.mp4",
  paper = "Dave17a.pdf",
  keywords = "CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Dela05,
  author = "Delahaye, David and Mayero, Micaela",
  title = {{Dealing with algebraic expressions over a field in Coq
           using Maple}},
  journal = "J. Symbolic Computation",
  volume = "39",
  pages = "569-592",
  year = "2005",
  abstract =
    "We describe an interface between the Coq proof assistant and the
    Maple symbolic computations system, which mainly consists in
    importing, in Coq. Maple computations regarding algebraic
    expressions over fields. These can either be pure computations,
    which do not require any validation, or computations used during
    proofs, which must be proved (to be correct) within Coq. These
    correctneess proofs are completed automatically thanks to the
    tactic Field, which deals with equalities over fields. This
    tactic, which may generate side conditions (regarding the
    denominators) that must be proved by the user, has been
    implemented in a reflxive way, which ensures both efficiency and
    certification. The implementation of this interface is quite light
    and can be very easily extended to get other Maple functions (in
    addition to the four functions we have imported and used in the
    examples given here).",
  paper = "Dela05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Demi79,
  author = "DeMilo, Richard A. and Lipton, Richard J. and Perlis, Alan J.",
  title = {{Social Processes and Proofs of Theorems and Programs}},
  journal = "Communications of the ACM",
  volume = "22",
  number = "5",
  year = "1979",
```

```
  pages = "271-280",
  abstract =
    "It is argued that formal verifications of programs, no matter how
    obtained, will not play the same key role in the development of
    computer science and software engineering as proofs do in mathematics.
    Furthermore the absence of continuity, the inevitability of change, and
    the complexity of specification of significantly many real programs
    make the formal verification process difficult to justify and manage.
    It is felt that ease of formal verification should not dominate program
    language design.",
  paper = "Demi79.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Denn00,
  author = "Dennis, Louise A. and Collins, Graham and Norrish, Michael
            and Boulton, Richard and Slind, Konrad and
            Robinson, Graham and Gordon, Mike and Melham, Tom",
  title = {{The PROSPER Toolkit}},
  journal = "LNCS",
  volume = "1785",
  publisher = "Springer-Verlag",
  pages = "78-92",
  year = "2000",
  link =
     "\url{https://link.springer.com/content/pdf/10.1007/3-540-46419-0_7.pdf}",
  abstract =
    "The PROSPER (Proof and Specification Assisted Design Environments)
    project advocates the use of toolkits which allow existing
    verification tools to be adapted to a more flexible format so that
    they may be treated as components. A system incorporating such tools
    becomes another component that can be embedded in an application. This
    paper describes the PROSPER Toolkit which enables this. The nature of
    communication between components is specified in a
    language-independent way. It is implemented in several common
    programming languages to allow a wide variety of tools to have access
    to the toolkit.",
  paper = "Denn00.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Devl79,
  author = "Devlin, Keith J.",
  title = {{Fundamentals of Contemporary Set Theory}},
  publisher = "Springer-Verlag",
```

```
  year = "1979",
  isbn = "978-0387904412"
}
```

———-

— **axiom.bib** —

```
@misc{Dijk72,
  author = "Dijkstra, Edsger",
  title = {{The Humble Programmer}},
  year = "1972",
  number = "EWD340",
  comment = "ACM Turing Lecture 1972",
  paper = "Dijk72.txt"
}
```

———-

— **axiom.bib** —

```
@book{Dijk76,
  author = "Dijkstra, Edsger",
  title = {{A Discipline of Programming}},
  publisher = "Prentice-Hall",
  year = "1976",
  isbn = "0-13-215871-X"
}
```

———-

— **axiom.bib** —

```
@misc{Dijk83,
  author = "Dijkstra, Edsger",
  title = {{Fruits of Misunderstanding}},
  year = "1983",
  link = "\url{https://www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD854.html}",
  paper = "Dijk83.txt"
}
```

———-

— **axiom.bib** —

```
@misc{Dolz04a,
  author = "Dolzmann, A. and Seidl, A. and Sturm, T.",
  title = {{Redlog User Manual}},
  year = "2004",
  comment = "Edition 3.0",
  link = "\url{http://www.reduce-algebra.com/reduce38-docs/redlog.pdf}",
```

```
  paper = "Dolz04a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Dybj90,
  author = "Dybjer, Peter",
  title = {{Inductive Sets and Families in Marin-L\"of's Type Theory and
           Their Set-Theoretic Semantics}},
  booktitle = "Proc. First Workshop on Logical Frameworks",
  year = "1990",
  link =
    "\url{http://www.cse.chalmers.se/~peterd/papers/Setsem\_Inductive.pdf}",
  abstract =
    "{Martin-L\"of}'s type theory is presented in several steps. The kernel
    is a dependently typed $\lambda$-calculs. Then there are schemata for
    inductive sets and families of sets and for primitive recursive functions
    and families of functions. Finally, there are set formers (generic
    polymorphism) and universes. At each step syntax, inference rules, and
    set-theoretic sematics are given",
  paper = "Dybj90.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dybj03,
  author = "Dybjer, Peter and Setzer, Anton",
  title = {{Induction-recursion and initial algebras}},
  journal = "Annals of Pure and Applied Logic",
  volume = "124",
  year = "2003",
  pages = "1-47",
  abstract =
    "Induction-recursion is a powerful definition method in intuitionistic
    type theory. It extends (generalized) inductive definitions and allows us
    to define all standard sets of Martin-{L\"of} type theory as well as a
    large collection of commonly occuring inductive data structures. It also
    includes a variety of universes which are constructive analogues of
    inaccessibles and other large cardinals below the first Mahlo cardinal.
    In this article we give a new compact formalization of inductive-recursive
    definnitions by modeling them as initial algebras in slice categories. We
    give generic formation, introduction, elimination, and equality rules
    generalizing the usual rules of type theory. Moreover, we prove that the
    elimination and equality rules are equivalent to the principle of the
    existence of initial algebras for certain endofunctors. We also show the
    equivalence of the current formulation with the formulation of
    induction-recursion as a reflection principle given in Dybjer and
    Setzer (Lecture Notes in Comput. Sci. 2183 (2001) 93). Finally we discuss
```

```
    two type-theoretic analogues of Mahlo cardinals in set theory: an external
    Mahlo universe which is defined by induction-recursion and captured by our
    formalization, and an internal Mahlo universe, which goes beyond induction-
    recursion. We show that the external Mahlo universe, and therefore also
    the theory of inductive-recursive definitions, have proof-theoretical
    strength of at least Rathjen's theory KPM.",
  paper = "Dybj03.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Dolz97,
  author = "Dolzmann, Andreas and Sturm, Thomas",
  title = {{Guarded Expressions in Practice}},
  link = "\url{http://redlog.dolzmann.de/papers/pdf/MIP-9702.pdf}",
  year = "1997",
  abstract =
    "Computer algebra systems typically drop some degenerate cases when
    evaluating expressions, e.g. $x/x$ becomes 1 dropping the case
    $x=0$. We claim that it is feasible in practice to compute also the
    degenerate cases yielding {\sl guarded expressions}. We work over real
    closed fields but our ideas about handling guarded expressions can be
    easily transferred to other situations. Using formulas as guards
    provides a powerful tool for heuristically reducing the combinatorial
    explosion of cases: equivalent, redundant, tautological, and
    contradictive cases can be detected by simplification and quantifier
    elimination. Our approach allows to simplify the expressions on the
    basis of simplification knowledge on the logical side. The method
    described in this paper is implemented in the REDUCE package GUARDIAN,
    which is freely available on the WWW.",
  paper = "Dolz97.pdf"
}
```

———-

— **axiom.bib** —

```
@inbook{Dosr11,
  author = "Dos Reis, Gabriel and Matthews, David and Li, Yue",
  title = {{Retargeting OpenAxiom to Poly/ML: Towards an Integrated Proof
           Assistants and Computer Algebra System Framework}},
  booktitle = "Calculemus",
  pages = "15-29",
  year = "2011",
  publisher = "Springer",
  isbn = "978-3-642-22673-1",
  link = "\url{http://paradise.caltech.edu/~yli/paper/oa-polyml.pdf}",
  abstract = "
    This paper presents an ongoing effort to integrate the Axiom family of
    computer algebra systems with Poly/ML-based proof assistants in the
```

```
    same framework. A long term goal is to make a large set of efficient
    implementations of algebraic algorithms available to popular proof
    assistants, and also to bring the power of mechanized formal
    verification to a family of strongly typed computer algebra systems at
    a modest cost. Our approach is based on retargeting the code generator
    of the OpenAxiom compiler to the Poly/ML abstract machine.",
  paper = "Dosr11.pdf",
  keywords = "axiomref, CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Duns00,
  author = "Dunstan, Martin N.",
  title = {{Adding Larch/Aldor Specifications to Aldor}},
  abstract =
    "We describe a proposal to add Larch-style annotations to the Aldor
    programming language, based on our PhD research. The annotations
    are intended to be machine-checkable and may be used for a variety
    of purposes ranging from compiler optimizations to verification
    condition (VC) generation. In this report we highlight the options
    available and describe the changes which would need to be made to
    the compiler to make use of this technology.",
  paper = "Duns00.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Duns98,
  author = "Dunstan, Martin and Kelsey, Tom and Linton, Steve and
            Martin, Ursula",
  title = {{Lightweight Formal Methods For Computer Algebra Systems}},
  publisher = "ACM Press",
  booktitle = "Proc. ISSAC 1998",
  year = "1998",
  location = "Rostock, Germany",
  pages = "80-87",
  link = "\url{http://www.cs.st-andrews.ac.uk/~tom/pub/issac98.pdf}",
  abstract =
    "Demonstrates the use of formal methods tools to provide a semantics
    for the type hierarchy of the Axiom computer algebra system, and a
    methodology for Aldor program analysis and verification. There are
    examples of abstract specifications of Axiom primitives.",
  paper = "Duns98.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Duns99a,
  author = "Dunstan, Martin N.",
  title = {{Larch/Aldor - A Larch BISL for AXIOM and Aldor}},
  school = "University of St. Andrews",
  year = "1999",
  abstract = "
    In this thesis we investigate the use of lightweight formal methods
    and verification conditions (VCs) to help improve the reliability of
    components constructed within a computer algebra system. We follow the
    Larch approach to formal methods and have designed a new behavioural
    interface specification language (BISL) for use with Aldor: the
    compiled extension language of Axiom and a fully-featured programming
    language in its own right. We describe our idea of lightweight formal
    methods, present a design for a lightweight verification condition
    generator and review our implementation of a prototype verification
    condition generator for Larch/Aldor.",
  paper = "Duns99a.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Duns99,
  author = "Dunstan, Martin and Kelsey, Tom and Martin, Ursula and
            Linton, Steve A.",
  title = {{Formal Methods for Extensions to CAS}},
  booktitle = "Proc. of FME'99",
  series = "FME'99",
  location = "Toulouse, France",
  year = "1999",
  pages = "1758-1777",
  link = "\url{http://tom.host.cs.st-andrews.ac.uk/pub/fm99.ps}",
  abstract =
    "We demonstrate the use of formal methods tools to provide a semantics
    for the type hierarchy of the AXIOM computer algebra system, and a
    methodology for Aldor program analysis and verification. We give a
    case study of abstract specifications of AXIOM primitives, and provide
    an interface between these abstractions and Aldor code.",
  paper = "Duns99.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

1.19.5 F

— **axiom.bib** —

```
@article{Farm92a,
  author = "Farmer, William H.",
  title = {{IMPS: System Description}},
  journal = "Lecture Notes in Computer Science",
  volume = "607",
  pages = "701-705",
  year = "1992",
  paper = "Farm92a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Farm96,
  author = "Farmer, William M. and Guttman, Joshua D. and
            Fabrega, F. Javier Thayer",
  title = {{IMPS: An Updated System Description}},
  journal = "LNCS",
  volume = "1104",
  pages = "298-302",
  year = "1996",
  paper = "Farm96.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Farm92,
  author = "Farmer, William H.",
  title = {{Little Theories}},
  journal = "LNCS",
  volume = "607",
  year = "1992",
  pages = "567-581",
  abstract =
    "In the ‘‘little theories’’ version of the axiomatic method, different
    portions of mathematics are developed in various different formal
    axiomatic theories. Axiomatic theories may be related by inclusion or
    by theory interpretation. We argue that the little theories approach
    is a desirable way to formalize mathematics, and we describe how IMPS,
    an Interactive Mathematical Proof System, supports it.",
  paper = "Farm92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Farm93b,
  author = "Farmer, William M.",
  title = {{A simple type theory with partial functions and subtypes}},
  journal = "Annals of Pure and Applied Logic",
  volume = "64",
  pages = "211-240",
  year = "1993",
  abstract =
    "Simple type theory is a higher-order predicate logic for reasoning
    about truth values, individuals, and simply typed total functions.  We
    present in this paper a version of simple type theory, called PF*, in
    which functions may be partial and types may have subtypes.  We define
    both a Henkin-style general models semantics and an axiomatic system
    for PF*, and we prove that the axiomatic system is complete with
    respect to the general models semantics.  We also define a notion of
    an interpretation of one PF* theory in another.  PF* is intended as a
    foundation for mechanized mathematics.  It is the basis for the logic
    of IMPS, an Interactive Mathematical Proof System developed at The
    MITRE Corporation.",
  paper = "Farm93b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Farm93a,
  author = "Farmer, William M. and Guttman, Joshua D. and Thayer, Javier",
  title = {{IMPS: An Interactive Mathematical Proof Systems}},
  journal = "J. of Automated Reasoning",
  volume = "11",
  pages = "213-248",
  year = "1993",
  abstract =
    "IMPS is an interactive mathematical proof system intended as a
    general-purpose too! formulating and applying mathematics in a
    familiar fashion.  The logic of IMPS is based on a version of simple
    type theory with partial functions and subtypes.  Mathematical
    specification and inference are performed relative to axiomatic
    theories, which can be related to one another via inclusion and theory
    interpretation.  IMPS provides relatively large primitive inference
    steps to facilitate human control of the deductive process and human
    comprehension of the resulting proofs.  An initial theory library
    containing over a thousand repeatable proofs covers significant portions
    of logic, algebra, and analysis and provides some support for modeling
    applications in computer science.",
  paper = "Farm93a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Farm93,
  author = "Farmer, William M. and Guttman, Joshua D. and Thayer, Javier",
  title = {{Reasoning with contexts}},
  journal = "LNCS",
  volume = "722",
  year = "1993",
  abstract =
    "Contexts are sets of formulas used to manage the assumptions that
    arise in the course of a mathematical deduction or calculation. This
    paper describes some techniques for symbolic computation that are
    dependent on using contexts, and are implemented in IMPS, an
    Interactive Mathematical Proof System.",
  paper = "Farm93.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Farm94,
  author = "Farmer, William M. and Guttman, Joshua D. and Nadel, Mark E.
            and Fabrega, F. Javier Thayer",
  title = {{Proof Script Pragmatics in IMPS}},
  journal = "LNCS",
  volume = "814",
  pages = "356-370",
  year = "1994",
  abstract =
    "This paper instroduces the IMPS proof script mechanism and some
    practical methods for exploiting it.",
  paper = "Farm94.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate02a,
  author = "Fateman, Richard J.",
  title = {{Symbolic Execution Merges Construction, Debugging and Proving}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/symex.pdf}",
  year = "2002",
  abstract =
    "There is naturally an interest in any technology which promises to
    assist us in producing correct programs. Some efforts attempt to
    insure correct programs by making their construction simpler. Some
    efforts are oriented toward increasing the effectiveness of testing to
```

```
    make the programs appear to perform as required. Other efforts are
    directed to prove the correctness of the resulting program. Symbolic
    execution, in which symbols instead of numbers are used in what
    appears to be a numerical program, is an old but to-date still not
    widely-used technique. It has been available in various forms for
    decades from the computer algebra community. Symbolic execution has
    the potential to assist in all these phases: construction, debugging,
    and proof. We describe how this might work specifically with regard to
    our own recent experience in the construction of correct linear
    algebra programs for structured matrices and LU factorization. We show
    how developing these programs with a computer algebra system, and then
    converting incrementally to use more efficient forms. Frequent symbolic
    execution of the algorithms, equivalent to testing over infinite test
    sets, aids in debugging, while strengthening beliefs that the correctness
    of results is an algebraic truth rather than an accident.",
  paper = "Fate02a.pdf, CAS-Proof, printed"
}
```

———-

— axiom.bib —

```
@inproceedings{Fate03a,
  author = "Fateman, Richard J.",
  title = {{High-level proofs of mathematical programs using automatic
           differentiation, simplification, and some common sense}},
  booktitle = "Proc. ISSAC 2003",
  pages = "88-94",
  year = "2003",
  isbn = "1-58113-641-2",
  abstract =
    "One problem in applying elementary methods to prove correctness of
    interesting scientific programs is the large discrepancy in level of
    discourse between low-level proof methods and the logic of scientific
    calculation, especially that used in a complex numerical program. The
    justification of an algorithm typically relies on algebra or analysis,
    but the correctness of the program requires that the arithmetic
    expressions are written correctly and that iterations converge to
    correct values in spite of truncation of infinite processes or series
    and the commission of numerical roundoff errors. We hope to help
    bridge this gap by showing how we can, in some cases, state a
    high-level requirement and by using a computer algebra system (CAS)
    demonstrate that a program satisfies that requirement. A CAS can
    contribute program manipulation, partial evaluation, simplification or
    other algorithmic methods. A novelty here is that we add to the usual
    list of techniques automatic differentiation, a method already widely
    used in optimization contexts where algorithms are differentiated. We
    sketch a proof of a numerical program to compute sine, and display a
    related approach to a version of a Bessel function algorithm for J0(x)
    based on a recurrence.",
  paper = "Fate03a.pdf",
  keywords = "CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Fitt96,
  author = "Fitting, M.",
  title = {{First-order Logic and Automated Theorem Proving}},
  publisher = "Springer-Verlag",
  year = "1996",
  isbn = "13.978-1-4612-2360-3",
  paper = "Fitt96.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Floy86,
  author = "Floyd, W.",
  title = {{Toward Interactive Design of Correct Programs}},
  booktitle = "Reading in Artificial Intelligence and Software Engeering",
  publisher = "Elsevier",
  pages = "331-334",
  year = "1986",
  isbn = "0-934613-12-5"
}
```

———-

— **axiom.bib** —

```
@article{Frad08,
  author = "Frade, Maria Joao",
  title = {{Calculus of Inductive Construction. Software Formal Verification}},
  year = "2008",
  link = "\url{http://www4.di.uminho.pt/~jno/mfes/0809/SFV-CIC.pdf}",
  journal = "MFES",
  paper = "Frad08.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Freg1891,
  author = "Frege, Gottlob",
  title = {{Function and Concept}},
  year = "1891",
  link = "\url{http://fitelson.org/proseminar/frege_fac.pdf}",
  paper = "Frege.pdf",
```

```
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Frey90,
  author = "Freyd, Peter and Scedrov, Andre",
  title = {{Categories, Allegories}},
  year = "1990",
  publisher = "North-Holland",
  comment = "Mathematical Library Vol 39",
  isbn = "978-0-444-703368-2",
  abstract =
    "On the Categories side, the book centers on that part of categorical
    algebra that studies exactness properties, or other properties enjoyed
    by nice or convenient categories such as toposes, and their
    relationship to logic (for example, geometric logic). A major theme
    throughout is the possibility of representation theorems (aka
    completeness theorems or embedding theorems) for various categorical
    structures, spanning back now about five decades (as of this writing)
    to the original embedding theorems for abelian categories, such as the
    Freyd-Mitchell embedding theorem.

    On the Allegories side: it may be said they were first widely
    publicized in this book. They comprise many aspects of relational
    algebra corresponding to the categorical algebra studied in the first
    part of the book"
}
```

———-

— **axiom.bib** —

```
@book{Frie08,
  author = "Friedman, Daniel P. and Wand, MItchell",
  title = {{Essentials of Programming Languages}},
  publisher = "MIT Press",
  year = "2008",
  isbn = "978-0-262-06279-4",
  keywords = "shelf"
}
```

———-

1.19.6 G

— **axiom.bib** —

```
@book{Gall86,
  author = "Gallier, Jean H.",
  title = {{Logic for Computer Science: Foundations of Automatic
            Theorem Proving}},
  publisher = "Harper and Row",
  year = "1986",
  isbn = "978-0486780825"
}
```

———-

— **axiom.bib** —

```
@book{Gedd94,
  author = "Geddes, D.",
  title = {{The DTP Manual}},
  publisher = "Stanford University",
  year = "1994",
  comment = "Don's Theorem Prover in Common Lisp",
  paper = "Gedd94.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Gent35,
  author = "Gentzen, Gerhard",
  title = {{Investigations into Logical Deduction}},
  year = "1935",
  pages = "68-131",
  paper = "Gent35.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gent64,
  author = "Gentzen, Gerhard",
  title = {{Investigations into Logical Deduction}},
  journal = "American Philosophical Quarterly",
  volume = "1",
  number = "4",
  year = "1964",
  pages = "288-306",
  paper = "Gent64.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Gent65,
  author = "Gentzen, Gerhard",
  title = {{Investigations into Logical Deduction: II}},
  journal = "American Philosophical Quarterly",
  volume = "2",
  number = "3",
  year = "1965",
  pages = "204-218",
  paper = "Gent65.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Geuv92,
  author = "Geuvers, Herman",
  title = {{The Calculus of Constructions and Higher Order Logic}},
  year = "1992",
  link = "\url{http://www.cs.ru.nl/~herman/PUBS/CC\_CHiso.pdf}",
  abstract =
    "The Calculus of Constructions (CC) is a typed lambda calculus for
    higher order intuitionistic logic: proofs of the higher order logic
    are interpreted as lambda terms and formulas as types. It is also the
    union of Girard's system $F_omega$, a higher order typed lambda
    calculus, and a first order dependent typed lambda calculus in the
    style of de Bruijn's Automath or Martin-Lof's intuitionistic theory
    of types. Using the impredicative coding of data types in $F_omega$,
    the Calculus of Constructions thus becomes a higher order language for
    the typing of functional programs. We shall introduce and try to
    explain CC by exploiting especially the first point of view, by
    introducing a typed lambda calculus that faithfully represent higher
    order predicate logic (so for this system the Curry-Howard
    'formuals-as-types isomorphism' is really an isomorphism.) Then we
    discuss some propositions that are provable in CC but not in the
    higher order logic, showing that the formulas-as-types embedding of
    higher order predicate logic into CC is not an isomorphism. It is our
    intention that this chapter can be read without any specialist
    knowledge of higher order logic or higher order typed lambda calculi.",
  paper = "Geuv92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Geuv02,
  author = "Geuvers, Herman and Pollack, Randy and Wiedijk, Freek and
            Zwanenburg, Jan",
```

```
  title = {{A Constructive Algebraic Hierarchy in Coq}},
  year = "2002",
  journal = "Journal of Symbolic Computation",
  abstract =
    "We describe a framework of algebraic structures in the proof assistant
    Coq. We have developed this framework as part of the FTA project in
    Nijmegen, in which a constructive proof of the Fundamental Theorem of
    Algebra has been formalized in Coq.

    The algebraic hierarchy that is described here is both abstract and
    structured. Structures like groups and rings are port of it in an
    abstract way, defining e.g. a ring as a tuple consisting of a group, a
    binary operation and a constant that together satisfy the properties
    of a ring. In this way, a ring automatically inherits the group
    properties of the additive subgroup. The algebraic hierarchy is
    formalized in Coq by applying a combination of labeled record types
    and coercions. In the labeled record types of Coq, one can use
    {\sl dependent types}: the type of one label may depend on another
    label. This allows to give a type to a dependent-typed tuple like
    $\langle A, f, a \rangle$, where $A$ is a set, $f$ an operation on $A$
    and $a$ an element of $A$. Coercions are functions that are used
    implicitly (they are inferred by the type checker) and allow, for
    example, to use the structure $\mathcal{A} := \langle A, f, a \rangle$
    as a synonym or the carrier set $A$, as is often done in mathematical
    practice. Apart rom the inheritance and reuse of properties, the
    algebraic hierarchy has proven very useful for reusing notations.",
  paper = "Geuv02.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Gime16,
  author = "Gimenez, Eduardo and Casteran, Pierre",
  title = {{A Tutorial on [Co-]Inductive Types in Coq}},
  year = "2016",
  link = "\url{https://coq.inria.fr/distrib/current/files/RecTutorial.pdf}",
  abstract =
    "This document is an introduction to the definition and use of
    inductive and co-inductive types in the {\sl Coq} proof environment.
    It explains how types like natural numbers and infinite streams are
    defined in {\sl Coq}, and the kind of proof techniques that can be
    used to reason about them (case analysis, induction, inversion of
    predicates, co-induction, etc.) Each technique is illustrated
    through an executable and self-contained {\sl Coq} script.",
  paper = "Gime16.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Giun94,
  author = "Giunchiglia, F. and Pecchiari, P. and Talcott, C.",
  title = {{Reasoning Theories: Towards an Architecture for Open Mechanized
           Reasoning Systems}},
  type = "technical report",
  number = "9409-15",
  institution = "IRST Trento Italy",
  year = "1994",
  paper = "Giun94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gogu82,
  author = "Goguen, J.A. and Meseguer, J.",
  title = {{Completeness of Many-sorted Equational Logic}},
  journal = "ACM SIGPLAN Notices",
  volume = "17",
  number = "1",
  year = "1982",
  pages = "9-17",
  abstract =
    "The rules of deduction which are usually used for many-sorted
    equational logic in computer science, for example in the study of
    abstract data types, are not sound. Correcting these rules by
    introducing explicit quantifiers yields a system which, although it is
    sound, is not complete; some new rules are needed for the addition and
    deletion of quantifiers. This note is intended as an informal, but
    precise, introduction to the main issues and results. It gives an
    example showing the unsoundness of the usual rules; it also gives a
    completeness theorem for our new rules, and gives necessary and
    sufficient conditions for the old rules to agree with the new.",
  paper = "Gogu82.pdf",
}
```

———-

— **axiom.bib** —

```
@article{Gogu06,
  author = "Goguen, Healfdene and McBride, Conor and McKinna, James",
  title = {{Eliminating Dependent Pattern Matching}},
  year = "2006",
  journal = "Lecture Notes in Computer Science",
  volume = "4060",
  pages = "521-540",
  link = "\url{http://cs.ru.nl/~james/RESEARCH/goguen2006.pdf}",
  abstract =
    "This paper gives a reduction-preserving translation from Coquand's
```

```
    {\sl dependent pattern matching} into a traditional type theory
    with universes, inductive types and relations and the axiom K. This
    translation serves as a proof of termination for structurally
    recursive pattern matching programs, provides an implementable
    compilation technique in the style of functional programming languages,
    and demonstrates the equivelence with a more easily understood type
    theory.",
  paper = "Gogu06.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Good75,
  author = "Good, Donald I. and London, Ralph L. and Bledsoe, W.W.",
  title = {{An Interactive Program Verification System}},
  journal = "SIGPLAN Notices",
  volume = "10",
  number = "6",
  pages = "482-492",
  year = "1975",
  abstract =
    "This paper is an initial progress report on the development of an
    interactive system for verifying that computer programs meet given
    formal specifications. The system is based on the conventional
    inductive assertion method: given a program and its specifications,
    the object is to generate the verification conditions, simplify them,
    and prove what remains. The important feature of the system is that
    the human user has the opportunity and obligation to help actively in
    the simplifying and proving. The user, for example, is the primary
    source of problem domain facts and properties needed in the proofs. A
    general description is given of the overall design philosophy,
    structure, and functional components of the system, and a simple
    sorting program is used to illustrate both the behavior of major
    system components and the type of user interaction the system provides.",
  paper = "Good75.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Gord96,
  author = "Gordon, Mike",
  title = {{From LCF to HOL: a short history}},
  year = "1996",
  link = "\url{http://www.cl.cam.ac.uk/~mjcg/papers/HolHistory.pdf}",
  paper = "Gord96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Gord93,
  author = "Gordon, Mike J.C. and Melham, T.F.",
  title = {{Introduction to HOL: A Theorem Proving Environment for Higher
           Order Logic}},
  link = "\url{http://www.cs.ox.ac.uk/tom.melham/pub/Gordon-1993-ITH.html}",
  publisher = "Cambridge University Press",
  year = "1993",
  isbn = "0-521-44189-7",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Gott05,
  author = "Gottliebsen, Hanne and Kelsey, Tom and Martin, Ursula",
  title = {{Hidden verification for computational mathematics}},
  journal = "Journal of Symbolic Computation",
  volume = "39",
  number = "5",
  pages = "539-567",
  year = "2005",
  link =
   "\url{http://www.sciencedirect.com/science/article/pii/S0747717105000295}",
  abstract =
    "We present hidden verification as a means to make the power of
    computational logic available to users of computer algebra systems
    while shielding them from its complexity. We have implemented in PVS a
    library of facts about elementary and transcendental function, and
    automatic procedures to attempt proofs of continuity, convergence and
    differentiability for functions in this class. These are called
    directly from Maple by a simple pipe-lined interface. Hence we are
    able to support the analysis of differential equations in Maple by
    direct calls to PVS for: result refinement and verification, discharge
    of verification conditions, harnesses to ensure more reliable
    differential equation solvers, and verifiable look-up tables.",
  paper = "Gott05.pdf",
  keywords = "axiomref, CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Grab17,
  author = {Grabm\"uller, Martin},
  title = {{Algorithm W}},
```

```
  year = "2017",
  link = "\url{https://github.com/mgrabmueller/AlgorithmW}",
  abstract =
    "In this paper we develop a complete implementation of Algorithm W for
     Hindley-Milner polyhmorphic type inference in Haskell",
  paper = "Grab17.pdf"
}
```

— **axiom.bib** —

```
@misc{Gran11,
  author = "Grant, Ian",
  title = {{The Hindley-Milner Type Inference Algorithm}},
  year = "2011",
  link = "\url{http://steshaw.org/hm/hindley-milner.pdf}",
  abstract =
    "The Hindley-Milner algorithm is described and an implementation in
    Standard ML is presented.",
  paper = "Gran11.pdf",
  keywords = "printed"
}
```

— **axiom.bib** —

```
@book{Grie81,
  author = "Gries, David",
  title = {{The Science of Programming}},
  publisher = "Springer-Verlag",
  year = "1981",
  isbn = "0-387-90641-X"
}
```

— **axiom.bib** —

```
@book{Gutt93,
  author = "Guttag, John V. and Horning, James J.",
  title = {{LARCH: Languages and Tools for Formal Specifications}},
  publisher = "Springer-Verlag",
  year = "1993",
  isbn = "3-540-94006-5"
}
```

1.19.7 H

— **axiom.bib** —

```
@misc{Hame89,
  author = "Van Hamelen, Frank",
  title = {{The CLAM Proof Planner}},
  year = "1989",
  publisher = "Dept. of AI, Univ. of Edinburgh"
}
```

————-

— **axiom.bib** —

```
@misc{Hard13,
  author = "Hardin, David S. and McClurg, Jedidiah R. and Davis, Jennifer A.",
  title = {{Creating Formally Verified Components for Layered Assurance
           with an LLVM to ACL2 Translator}},
  link = "\url{http://www.jrmcclurg.com/papers/law\_2013\_paper.pdf}",
  abstract =
    "This paper describes an effort to create a library of formally
    verified software component models from code that have been compiled
    using the Low-Level Virtual Machine (LLVM) intermediate form. The idea
    is to build a translator from LLVM to the applicative subset of Common
    Lisp accepted by the ACL2 theorem prover. They perform verification of
    the component model using ACL2's automated reasoning capabilities.",
  paper = "Hard13.pdf"
}
```

————-

— **axiom.bib** —

```
@misc{Hard14,
  author = "Hardin, David S. and Davis, Jennifer A. and Greve, David A. and
            McClurg, Jedidiah R.",
  title = {{Development of a Translator from LLVM to ACL2}},
  link = "\url{http://arxiv.org/pdf/1406.1566}",
  abstract = "
    In our current work a library of formally verified software components
    is to be created, and assembled, using the Low-Level Virtual Machine
    (LLVM) intermediate form, into subsystems whose top-level assurance
    relies on the assurance of the individual components. We have thus
    undertaken a project to build a translator from LLVM to the
    applicative subset of Common Lisp accepted by the ACL2 theorem
    prover. Our translator produces executable ACL2 formal models,
    allowing us to both prove theorems about the translated models as well
    as validate those models by testing.  The resulting models can be
    translated and certified without user intervention, even for code with
    loops, thanks to the use of the def::ung macro which allows us to
    defer the question of termination. Initial measurements of concrete
```

```
    execution for translated LLVM functions indicate that performance is
    nearly 2.4 million LLVM instructions per second on a typical laptop
    computer. In this paper we overview the translation process and
    illustrate the translator's capabilities by way of a concrete example,
    including both a functional correctness theorem as well as a
    validation test for that example.",
  paper = "Hard14.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Harp11,
  author = "Harper, Robert",
  title = {{Programming in Standard ML}},
  year = "2011",
  publisher = "CMU",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Harp13,
  author = "Harper, Robert",
  title = {{15.819 Homotopy Type Theory Course}},
  link = "\url{http://www.cs.cmu.edu/~rwh/courses/hott}",
  year = "2013"
}
```

———-

— **axiom.bib** —

```
@incollection{Harr93,
  author = "Harrison, John and Thery, Laurent",
  title = {{Reasoning about the Reals: The Marriage of HOL and
            Maple}},
  booktitle = "Logic Programming and Automated Reasoning",
  publisher = "Springer-Verlag",
  year = "1993",
  pages = "351-353",
  link =
    "\url{https://link.springer.com/content/pdf/10.1007/3-540-56944-8_68.pdf}",
  abstract =
    "Computer algebra systems are extremely powerful and flexible, but
    often give results which require careful interpretation or are
    downright incorrect. By contrast, theorem provers are very reliable
    but lack the powerful specialized decision procedures and heuristics
```

```
    of computer algebra systems. In this paper we try to get the best of
    both worlds by careful exploitation of a link between a theorem prover
    and a computer algebra system.",
  paper = "Harr93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Harr94a,
  author = "Harrison, John",
  title = {{Constructing the Real Numbers in HOL}},
  journal = "Formal Methods in Systems Design",
  volume = "5",
  pages = "35-39",
  year = "1994",
  abstract =
    "This paper describes a construction of the real numbers in the HOL
    theorem-prover by strictly definitional means using a version of
    Dedekind's method. It also outlines the theory of mathematical
    analysis that has been built on top of it and discusses current and
    potential applications in verification and computer algebra.",
  paper = "Harr94a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Harr94,
  author = "Harrison, John and Thery, Laurent",
  title = {{Extending the HOL Thoerem Prover with a Computer Algebra System
           to Reason about the Reals}},
  booktitle = "Proc. Higher Order Logic Theorem Proving",
  year = "1994",
  publisher = "Springer",
  pages = "174-184",
  isbn = "978-3-540-48346-5",
  abstract =
    "In this paper we describe an environment for reasoning about the
    reals which combines the rigour of a theorem prover with the power of
    a computer algebra system.",
  paper = "Harr94.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Harr95,
  author = "Harrison, John",
  title = {{Metatheory and Reflection in Theorem Proving: A Survey
           and Critique}},
  institution = "SRI Cambridge",
  year = "1995",
  type = "technical report",
  number = "CRC-053",
  abstract =
    "One way to ensure correctness of the inference performed by computer
    theorem provers is to force all proofs to be done step by step in a
    simple, more or less traditional, deductive system. Using techniques
    pioneered in Edinburgh LCF, this can be made palatable. However, some
    believe such an approach will never be efficient enough for large,
    complex proofs. One alternative, commonly called reflection, is to
    analyze proofs using a second layer of logic, a metalogic, and so
    justify abbreviating or simplifying proofs, making the kinds of
    shortcuts humans often do or appealing to specialized decision
    algorithms. In this paper we contrast the fully-expansive LCF approach
    with the use of reflection. We put forward arguments to suggest that
    the inadequacy of the LCF approach has not been adequately
    demonstrated, and neither has the practical utility of reflection
    (notwithstanding its undoubted intellectual interest). The LCF system
    with which we are most concerned is the HOL proof assistant.

    The plan of the paper is as follows. We examine ways of providing user
    extensibility for theorem provers, which naturally places the LCF and
    reflective approaches in opposition. A detailed introduction to LCF is
    provided, emphasizing ways in which it can be made efficient. Next, we
    present a short introduction to metatheory and its usefulness, and,
    starting from Gdel's proofs and Feferman's transfinite progressions
    of theories, look at logical 'reflection principles'. We show how to
    introduce computational 'reflection principles' which do not extend
    the power of the logic, but may make deductions in it more efficient,
    and speculate about their practical usefulness. Applications or
    proposed applications of computational reflection in theorem proving
    are surveyed, following which we draw some conclusions. In an
    appendix, we attempt to clarify a couple of other notions of
    'reflection' often encountered in the literature.

    The paper questions the too-easy acceptance of reflection principles
    as a practical necessity. However I hope it also serves as an adequate
    introduction to the concepts involved in reflection and a survey of
    relevant work. To this end, a rather extensive bibliography is
    provided.",
  paper = "Harr95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Harr96,
  author = "Harrison, John Robert",
  title = {{Theorem Proving with the Real Numbers}},
  school = "Churchhill College",
  comment = "U. Cambrige Computer Lab Tech Report 408,
            also Springer-Verlag 1988 ISBN 3-548-762566-6",
  year = "1996",
  abstract =
    "This thesis discusses the use of the real numbers in theorem
    proving. Typically, theorem provers only support a few 'discrete'
    datatypes such as the natural numbers. However the availability of the
    real numbers opens up many interesting and important application
    areas, such as the verification of floating point hardware and hybrid
    systems. It also allows the formalization of many more branches of
    classical mathematics, which is particularly relevant for attempts to
    inject more rigour into computer algebra systems.

    Our work is conducted in a version of the HOL theorem prover. We
    describe the rigorous definitional construction of the real numbers,
    using a new version of Cantor's method, and the formalization of a
    significant portion of real analysis. We also describe an advanced
    derived decision procedure for the 'Tarski subset' of real algebra as
    well as some more modest but practically useful tools for automating
    explicit calculations and routine linear arithmetic reasoning.

    Finally, we consider in more detail two interesting application
    areas. We discuss the desirability of combining the rigour of theorem
    provers with the power and convenience of computer algebra systems,
    and explain a method we have used in practice to achieve this. We then
    move on to the verification of floating point hardware. After a
    careful discussion of possible correctness specifications, we report
    on two case studies, one involving a transcendental function.

    We aim to show that a theory of real numbers is useful in practice and
    interesting in theory, and that the 'LCF style' of theorem proving is
    well suited to the kind of work we describe. We hope also to convince
    the reader that the kind of mathematics needed for applications is
    well within the abilities of current theorem proving technology.",
  paper = "Harr96.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@misc{Harr06a,
  author = "Harrison, John",
  title = {{Formal Verification of Floating-point Arithmetic at Intel}},
  year = "2006",
  link = "\url{http://www.cl.cam.ac.uk/~jrh13/slides/jnao-02jun06/slides.pdf}",
  paper = "Harr06a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hear18,
  author = "O'Hearn, Peter W.",
  title = {{Continuous Reasoning: Scaling the Impact of Formal Methods}},
  booktitle = "LICS 18",
  year = "2018",
  publisher = "ACM",
  isbn = "978-1-4503-5583-4",
  abstract =
    "This paper describes work in continuous reasoning , where formal
    reasoning about a (changing) codebase is done in a fashion which
    mirrors the iterative, continuous model of software development that
    is increasingly practiced in indus- try. We suggest that advances in
    continuous reasoning will allow formal reasoning to scale to more
    programs, and more programmers. The paper describes the rationale for
    contin- uous reasoning, outlines some success cases from within
    industry, and proposes directions for work by the scientific
    community.",
  paper = "Hear18.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Hear12,
  author = "O'Hearn, Peter W.",
  title = {{A Primer on Separation Logic (and Automatic Program
           Verification and Analysis}},
  year = "2012",
  link =
"\url{www0.cs.ucl.ac.uk/staff/p.ohearn/papers/Marktoberdorf11LectureNotes.pdf}",
  abstract =
    "These are the notes to accompany a course at the Marktoberdorf PhD
    summer school in 2011. The course consists of an introduction to
    separation logic, with a slant towards its use in automatic program
    verification and analysis.",
  paper = "Hear12.pdf",
  keywords = "printed"
}
```

———-

,

— **axiom.bib** —

```
@misc{Heer02,
```

```
  author = "Heeren, Bastiaan and Hage, Jurriaan and Swierstra, Doaitse",
  title = {{Generalizing Hindley-Milner Type Inference Algorithms}},
  year = "2002",
  link = "\url{https://pdfs.semanticscholar.org/8983/233b3dff2c5b94efb31235f62bddc22dc899.pdf}",
  abstract =
    "Type inferencing according to the standard algorithms $W$ and $M$
    often yields uninformative error messages. Many times, this is a
    consequence of a bias inherent in the algorithms. The method
    developed here is to first collect constraints from the program, and
    to solve these afterwards, possibly under the influence of a
    heuristic.  We show the soundness and completeness of our algorithm.
    The algorithms $W$ and $M$ turn out to be deterministic instances of our
    method, giving the correctness for $W$ and $M$ with respect to the
    Hindley-Milner typing rules for free.  We also show that our algorithm
    is more flexible, because it naturally allows the generation of
    multiple messages",
  paper = "Heer02.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Hoar69,
  author = "Hoare, C. A. R.",
  title = {{An Axiomatic Basis for Computer Programming}},
  journal = "CACM",
  volume = "12",
  number = "10",
  pages = "576-580",
  year = "1969",
  link = "\url{https://www.cs.cmu.edu/~crary/819-f09/Hoare69.pdf}",
  abstract =
    "In this paper an attempt is made to explore the logical foundations
    of computer programming by use of techniques which were first applied
    in the study of geometry and have later been extended to other branches
    of mathematics. This involves the elucidation of sets of axioms and
    rules of inference which can be used in proofs of the properties of
    computer programs. Examples are given of such axioms and rules, and
    a formal proof of a simple theorem is displayed. Finally, it is argued
    that important advantages, both theoretical and practical, may follow
    from a pursuance of these topics",
  paper = "Hoar69.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Homa94a,
  author = "Homann, Karsten",
```

```
  title = {{Integrating Explanation-Based Learning in Symbolic
           Computing}},
  booktitle = "Advances in Artificial Intelligence -- Theory and
               Application II, Volume II",
  pages = "130-135",
  year = "1994"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Homa96a,
  author = "Homann, Karsten",
  title = {{Symbolisches L\"osen mathematischer Probleme durch
           Kooperation algorithmischer und logischer Systeme}},
  comment = {{Symboic Mathematical Problem Solving by Cooperation of Algorithmic and Logical Services (in German
  year = "1996",
  school = {Universit\"at Karlsruhe},
  paper = "Homa96a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Howa80,
  author = "Howard, W. A.",
  title = {{The Formulae-as-Types Notion of Construction}},
  link = "\url{http://lecomte.al.free.fr/ressources/PARIS8_LSL/Howard80.pdf}",
  year = "1980",
  abstract =
    "The following consists of notes which were privately circulated in
    1969. Since they have been referred to a few times in the literature,
    it seems worth while to publish them. They have been rearranged for
    easier reading, and some inessential corrections have been made.

    The ultimate goal was to develop a notion of construction suitable for
    the interpretation of intuitionistic mathematics. The notion of
    construction developed in the notes is certainly too crude for that,
    so the use of the word {\sl construction} is not very appropriate.
    However, the terminology has been kept in order to preserve the
    original title and also to preserve the character of the notes. The
    title has a second defect; namely, the {\sl type} should be regarded
    as a abstract object whereas a {\sl formula} is the name of a type.",
  paper = "Howa80.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Howe88,
  author = "Howe, Douglas J.",
  title = {{Computational Metatheory in Nuprl}},
  journal = "LNCS",
  volume = "310",
  pages = "238-257",
  year = "1988",
  publisher = "Springer-Verlag",
  abstract =
    "This paper describes an implementation within Nuprl of mechanisms
    that support the use of Nuprl's type theory as a language for
    constructing theorem-proving procedures. The main component of the
    implementation is a large library of definitions, theorems and
    proofs. This library may be regarded as the beginning of a book of
    formal mathematics; it contains the formal development and explanation
    of a useful subset of Nuprl's metatheory, and of a mechanism for
    translating results established about this embedded metatheory to the
    object level. Nuprl's rich type theory, besides permitting the
    internal development of this partial reflection mechanism, allows us
    to make abstractions that drastically reduce the burden of
    establishing the correctness of new theorem-proving procedures. Our
    library includes a formally verified term-rewriting system"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Huet87,
  author = {Huet, G\'erard},
  title = {{Induction Principles Formalized in the Calculus of Constructions}},
  booktitle = "TAPSOFT 87",
  publisher = "Springer-Verlag",
  series ="LNCS 249",
  year = "1987",
  pages = "276-286",
  abstract =
    "The Calculus of Constructions is a higher-order formalism for writing
    constructive proofs in a natural deduction style, inspired from work
    by de Bruijn, Girard, Martin-Lof, and Scott. THe calculus and its
    syntactic theory were presented in Coquand's thesis, and an
    implementation by the author was used to mechanically verify a
    substantial number of proofs demonstrating the power of expression of
    the formalism. The Calculus of Constructions is proposed as a
    foundation for the design of programming environments where programs
    are developed consistently with formal specifications. The current
    paper shows how to define inductive concepts in the calculus.

    A very general induction schema is obtained by postulating all
    elements of the type of interest to belong to the standard
    interpretation associated with a predicate map. This is similar to the
    treatment of D. Park, but the power of expression of the formallism
    permits a very direct treatment, in a language that is formalized
```

```
    enough to be actually implemented on a computer. Special instances of
    the induction schema specialize to Noetherian induction and Structural
    induction over any algebraic type.  Computational Induction is treated
    in an axiomatization of Domain Theory in Constructions. It is argued
    that the resulting principle is more powerful than LCF's, since the
    restriction on admissibility is expressible in the object language.",
  paper = "Huet87.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Heut16,
  author = {Huet, G\'erard  and Kahn, Gilles and Paulin-Mohring, Christine},
  title = {{The COQ Proof Assistant. A Tutorial}},
  year = "2016",
  link = "\url{https://coq.inria.fr/distrib/current/files/Tutorial.pdf}",
  paper = "Heut16.pdf"
}
```

———-

1.19.8 J

— **axiom.bib** —

```
@misc{Jack94,
  author = "Jackson, Paul",
  title = {{Exploring Abstract Algebra in Constructive Type Theory}},
  year = "1994",
  abstract =
    "I describe my implementation of computational abstract algebra in
    the Nuprl system. I focus on my development of multivariate
    polynomials. I show how I use Nuprl's expressive type theory to define
    classes of free abelian monoids and free monoid algebras. These
    classes are combined to create a class of all implementations of
    polynomials.  I discuss the issues of subtyping and computational
    content that came up in designing the class definitions. I give
    examples of relevant theory developments, tactics and proofs. I
    consider how Nuprl could act as an algebraic 'oracle' for a computer
    algebra system and the relevance of this work for abstract functional
    programming.",
  paper = "Jack94.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Jack95,
  author = "Jackson, Paul Bernard",
  title = {{Enhancing the NUPRL Proof Development System and Applying it to
           Computational Abstract Algebra}},
  school = "Cornell University",
  year = "1995",
  month = "1",
  abstract = "
    This thesis describes substantial enhancements that were made to the
    software tools in the Nuprl system that are used to interactively
    guide the production of formal proofs. Over 20,000 lines of code were
    written for these tools. Also, a corpus of formal mathematics was
    created that consists of roughly 500 definitions and 1300
    theorems. Much of this material is of a foundational nature and
    supports all current work in Nuprl. This thesis concentrates on
    describing the half of this corpus that is concerned with abstract
    algebra and that covers topics central to the mathematics of the
    computations carried out by computer algebra systems.

    The new proof tools include those that solve linear arithmetic
    problems, those that apply the properties of order relations, those
    that carry out inductive proof to support recursive definitions, and
    those that do sophisticated rewriting. The rewrite tools allow
    rewriting with relations of differing strengths and take care of
    selecting and applying appropriate congruence lemmas automatically.
    The rewrite relations can be order relations as well as equivalence
    relations. If they are order relations, appropriate monotonicity
    lemmas are selected.

    These proof tools were heavily used throughout the work on
    computational algebra. Many examples are given that illustrate their
    operation and demonstrate their effectiveness.

    The foundation for algebra introduced classes of monoids, groups, ring
    and modules, and included theories of order relations and
    permutations.  Work on finite sets and multisets illustrates how a
    quotienting operation hides details of datatypes when reasoning about
    functional programs. Theories of summation operators were developed
    that drew indices from integer ranges, lists and multisets, and that
    summed over all the classes mentioned above. Elementary factorization
    theory was developed that characterized when cancellation monoids are
    factorial. An abstract data type for the operations of multivariate
    polynomial arithmetic was defined and the correctness of an
    implementation of these operations was verified. The implementation is
    similar to those found in current computer algebra systems.

    This work was all done in Nuprl's constructive type theory. The thesis
    discusses the appropriateness of this foundation, and the extent to
    which the work relied on it.",
  paper = "Jack95.pdf",
  keyword = "axiomref, CAS-Proof, printed"
}
```

———-

— axiom.bib —

```
@book{Jone93,
  author = "Jones, C.",
  title = {{Completing the Rationals and Metric Spaces in LEGO}},
  booktitle = "Logical Frameworks",
  pages = "209-222",
  year = "1993",
  publisher = "Cambridge University Press"
}
```

———-

— axiom.bib —

```
@misc{JARx96,
  author = "various",
  title = {{Journal of Automated Reasoning}},
  publisher = "Springer",
  year = "1996",
  volume = "16",
  number = "1/2",
  comment = "Special Issue on Inductive Proof"
}
```

———-

— axiom.bib —

```
@misc{Juds16,
  author = "Judson, Thomas W.",
  title = {{Abstract Algebra: Theory and Applications}},
  link = "\url{http://abstract.ups.edu/download/aata-20160809-sage-7.3.pdf}",
  year = "2016"
}
```

———-

1.19.9 K

— axiom.bib —

```
@inproceedings{Kali07,
  author = "Kaliszyk, Cezary and Wiedijk, Freek",
  title = {{Certified Computer Algebra on Top of an Interactive Theorem
           Prover}},
  booktitle = "Toward Mecanized Mathematical Assistants",
  pages = "94-105",
```

```
  year = "2007",
  abstract =
    "We present a prototype of a computer algebra system that is built on
    top of a proof assistant, HOL Light. This architecture guarantees that
    one can be certain that the system will make no mistakes. All
    expressions in the system will have precise semantics, and the proof
    assistant will check the correctness of all simplifications according
    to this semantics. The system actually proves each simplification
    performed by the computer algebra system.

    Although our system is built on top of a proof assistant, we designed
    the user interface to be very close in spirit to the interface of
    systems like Maple and Mathematica. The system, therefore, allows the
    user to easily probe the underlying automation of the proof assistant
    for strengths and weaknesses with respect to the automation of
    mainstream computer algebra systems. The system that we present is a
    prototype, but can be straightforwardly scaled up to a practical
    computer algebra system.",
  paper = "Kali07.pdf",
  keywords = "axiomref, CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Kali08,
  author = "Kaliszyk, Cezary",
  title = {{Automating Side Conditions in Formalized Partial Functions}},
  journal = "LNCS",
  volume = "5144",
  year = "2008",
  abstract =
    "Assumptions about the domains of partial functions are necessary in
    state-of-the-art proof assistants. On the other hand when
    mathematicians write about partial functions they tend not to
    explicitly write the side conditions. We present an approach to
    formalizing partiality in real and complex analysis in total
    frameworks that allows keeping the side conditions hidden from the
    user as long as they can be computed and simplified
    automatically. This framework simplifies defining and operating on
    partial functions in formalized real analysis in HOL Light. Our
    framework allows simplifying expressions under partiality conditions
    in a proof assistant in a manner that resembles computer algebra
    systems.",
  paper = "Kali08.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kali08a,
  author = "Kaliszyk, Cezary and Wiedijk, Freek",
  title = {{Merging Procedural and Declarative Proof}},
  journal = "LNCS",
  volume = "5497",
  year = "2008",
  abstract =
    "There are two different styles for writing natural deduction proofs:
    the 'Gentzen' style in which a proof is a tree with the conclusion at
    the root and the assumptions at the leaves, and the 'Fitch' style
    (also called flag style) in which a proof consists of lines that are
    grouped together in nested boxes.

    In the world of proof assistants these two kinds of natural deduction
    correspond to procedural proofs (tactic scripts that work on one or
    more subgoals, like those of the Coq, HOL and PVS systems), and
    declarative proofs (like those of the Mizar and Isabelle/Isar
    languages).

    In this paper we give an algorithm for converting tree style proofs to
    flag style proofs. We then present a rewrite system that simplifies
    the results.

    This algorithm can be used to convert arbitrary procedural proofs to
    declarative proofs. It does not work on the level of the proof terms
    (the basic inferences of the system), but on the level of the
    statements that the user sees in the goals when constructing the
    proof.

    The algorithm from this paper has been implemented in the ProofWeb
    interface to Coq. In ProofWeb a proof that is given as a Coq proof
    script (even with arbitrary Coq tactics) can be displayed both as a
    tree style and as a flag style proof.",
  paper = "Kali08a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Kali09,
  author = "Kaliszyk, Cezary",
  title = {{Correctness and Availability: Building Computer Algebra on top
           of Proof Assistants and making Proof Assistants available over
           the Web}},
  year = "2009",
  school = "Radboud University, Nijmegen",
  link = "\url{http://cl-informatik.uibk.ac.at/users/cek/docs/09/kaliszyk\_thesis\_webdoc.pdf}",
  abstract =
    "In this thesis we present an approach to extending the usability
     of proof assistants in mathematics and computer science. We do it
     in two ways: by combining proof assistants with computer algebra
```

```
    systems and by providing interactive access to such systems on
    the web.",
  paper = "Kali09.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Kauf98,
  author = "Kaufmann, Matt and Moore, J Strother",
  title = {{A Precise Description of the ACL2 Logic}},
  year = "1998",
  link = "\url{www.cs.utexas.edu/users/moore/publications/km97a.pdf}",
  abstract = "The ACL2 logic is a first-order, essentially quantifier-free
   logic of total recursive functions providing mathematical induction
   and several extension principles, including symbol package definition
   and recursive function definition. In this document we describe the
   logic more precisely.",
  paper = "km97a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Kenn13,
  author = "Kennedy, Andrew and Benton, Nick and Jensen, Jonas B. and
            Dagand, Pierre-Evariste",
  title = {{Coq: The world's best macro assembler?}},
  year = "2013",
  link = "\url{http://research.microsoft.com/en-us/um/people/nick/coqasm.pdf}",
  abstract =
    "We describe a Coq formalization of a subset of the x86
     architecture. One emphasis of the model is brevity: using dependent
     types, type classes and notation we give the x86 semantics a makeover
     that counters its reputation for baroqueness. We model bits, bytes,
     and memory concretely using functions that can be computed inside Coq
     itself; concrete representations are mapped across to mathematical
     objects in the SSREFLECT library (naturals, and integers modulo 2n)
     to prove theorems. Finally, we use notation to support conventional
     assembly code syntax inside Coq, including lexically-scoped
     labels. Ordinary Coq definitions serve as a powerful ``macro'' feature
     for everything from simple conditionals and loops to stack-allocated
     local variables and procedures with parameters. Assembly code can be
     assembled within Coq, producing a sequence of hex bytes. The assembler
     enjoys a correctness theorem relating machine code in memory to a
     separation-logic formula suitable for program verification.",
  paper = "Kenn13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kerb96,
  author = "Kerber, Manfred and Kohlhase, Michael and Sorge, Volker",
  title = {{Integrating Computer Algebra with Proof Planning}},
  journal = "Lecture Notes in Computer Science",
  volume = "1128",
  pages = "204-215",
  year = "1996",
  abstract =
    "Mechanised reasoning systems and computer algebra systems have
    apparently different objectives.  Their integration is, however,
    highly desirable, since in many formal proofs both of the two
    different tasks, proving and calculating, have to be performed.  In
    the context of producing reliable proofs, the question how to ensure
    correctness when integrating a computer algebra system into a
    mechanised reasoning system is crucial.  In this contribution, we
    discuss the correctness problems that arise from such an integration
    and advocate an approach in which the calculations of the computer
    algebra system are checked at the calculus level of the mechanised
    reasoning system.  We present an implementation which achieves this
    by adding a verbose mode to the computer algebra system which produces
    high-level protocol information that can be processed by an interface
    to derive proof plans.  Such a proof plan in turn can be expanded to
    proofs at different levels of abstraction, so the approach is
    well-suited for producing a high-level verbalised explication as well
    as for a low-level (machine checkable) calculus-level proof.",
  paper = "Kerb96.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@article{Kerb07,
  author = "Kerber, Manfred and Pollet, Martin",
  title = {{Informal and Formal Representations in Mathewmatics}},
  journal = "Studies in Logic, Grammar and Rhetoric 10",
  volume = "23",
  year = "2007",
  isbn = "978-83-7431-128-1",
  abstract =
    "In this paper we discuss the importance of good representations in
    mathematics and relate them to general design issues. Good design
    makes life easy, bad design difficult. For this reason experienced
    mathematicians spend a significant amount of their time on the design
    of their concepts.  While many formal systems try to support this by
    providing a high-level language, we argue that more should be learned
    from the mathematical practice in order to improve the applicability
    of formal systems.",
  paper = "Kerb07.pdf",
  keywords = "printed, DONE"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Kome11,
  author = "Komendantsky, Vladimir and Konovalov, Alexander and Linton, Steve",
  title = {{View of Computer Algebra Data from Coq}},
  journal = "Lecture Notes in Computer Science",
  volume = "6824",
  pages = "74-80",
  year = "2011",
  publisher = "Springer-Verlag",
  abstract =
    "Data representation is an important aspect of software composition.
    It is often the case that different software components are
    programmed to represent data in the ways which are the most
    appropriate for their problem domains. Sometimes, converting data from
    one representation to another is a non-trivial task. This is the
    case with computer algebra systems and type-theory based interactive
    theorem provers such as Coq. We provide some custom instrumentation
    inside Coq to support a computer algebra system (CAS) communication
    protocol known as SC-SCP. We describe general aspects of viewing
    OpenMath terms produced by a CAS in the calculus of Coq, as well as
    viewing pure Coq terms in a simpler type system that is behind OpenMath.",
  paper = "Kome11.pdf",
  keywords = "CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Kowa79,
  author = "Kowalski, Robert",
  title = {{Algorithm = Logic + Control}},
  journal = "CACM",
  volume = "22",
  number = "7",
  year = "1979",
  paper = "Kowa79.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Koun90,
  author = "Kounalis, Emmanuel and Rusinowitch, Michael",
```

```
  title = {{A Proof System for Conditional Algebraic Specifications}},
  booktitle = "Conference Conditional and Typed Rewriting Systems",
  year = "1990",
  abstract =
    "Algebraic specifications provide a formal basis for designing
    data-structures and reasoning about their properties.
    Sufficient-completeness and consistency are fundamental
    notions for building algebraic specifications in a modular way. We
    give in this paper effective methods for testing these properties for
    specifications with conditional axioms."
}
```

———-

— **axiom.bib** —

```
@phdthesis{Kreb15,
  author = "Krebbers, Robbert Jan",
  title = {{The C standard formalized in Coq}},
  school = "Radboud University",
  year = "2015",
  file = "Kreb15.pdf",
  link = "\url{http://robbertkrebbers.nl/thesis.html}"
}
```

———-

— **axiom.bib** —

```
@misc{Kreb15a,
  author = "Krebbers, Robbert and Wiedijk, Freek",
  title = {{A Typed C11 Semantics for Interactive Theorem Proving}},
  year = "2015",
  link = "\url{http://robbertkrebbers.nl/research/articles/interpreter.pdf}",
  abstract =
    "We present a semantics of a significant fragment of the C programming
    language as described by the C11 standard. It consists of a small step
    semantics of a core language, which uses a structured memory model to
    capture subtleties of C11, such as strict-aliasing restrictions
    related to unions, that have not yet been addressed by others. The
    semantics of actual C programs is defined by translation into this
    core language. We have an explicit type system for the core language,
    and prove type preservation and progress, as well as type correctness
    of the translation.

    Due to unspecified order of evaluation, our operational semantics is
    non-deterministic. To explore all defined and undefined behaviors, we
    present an executable semantics that computes a stream of finite sets
    of reachable states. It is proved sound and complete with respect to
    the operational semantics.

    Both the translation into the core language and the executable
```

```
    semantics are defined as Coq programs.  Extraction to OCaml is used to
    obtain a C interpreter to run and test the semantics on actual C
    programs. All proofs are fully formalized in Coq.",
  paper = "Kreb15a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Kreb17,
  author = "Krebbers, Robbert Jan",
  title = {{The CH$_2$O formalization of ISO C11}},
  year = "2017",
  link = "\url{http://robbertkrebbers.nl/research/ch2o/}"
}
```

———-

— **axiom.bib** —

```
@article{Kupe14,
  author = "Kuper, Jan and Wester, Rinse",
  title = {{N Queens on an FPGA: Mathematics, Programming, or Both?}},
  year = "2014",
  link = "\url{http://doc.utwente.nl/94663/1/NQueensOnFPGA.pdf}",
  publisher = "Open Channel Publishing Ltd",
  journal = "Communicating Process Architectures 2014",
  abstract =
    "This paper presents a design methodology for deriving an FPGA
    implementation directly from a mathematical specification, thus
    avoiding the switch in semantic perspective as is present in widely
    applied methods which include an imperative implementation as an
    intermediate step.

    The first step in the method presented in this paper is to transform a
    mathematical specification into a Haskell program. The next step is to
    make repetition structures explicit by higher order functions, and
    after that rewrite the specification in the form of a Mealy
    Machine. Finally, adaptations have to be made in order to comply to
    the fixed nature of hardware. The result is then given to
    C$\lambda$aSH, a compiler which generates synthesizable VHDL from the
    resulting Haskell code. An advantage of the approach presented here is
    that in all phases of the process the design can be directly simulated
    by executing the defining code in a standard Haskell environment.

    To illustrate the design process, the N queens problem is chosen as a
    running example."
}
```

———-

— **axiom.bib** —

```
@article{Kell12,
  author = "Keller, Chantal and Lasson, Marc",
  title = {{The Refined Calculus of Inductive Construction: Parametricity and
           Abstraction}},
  journal = "arXiv",
  year = "2012",
  link = "\url{http://arxiv.org/pdf/1211.6341v1.pdf}",
  abstract =
    "We present a refinement of the Calculus of Inductive Constructions in
    which one can easily define a notion of relational parametricity. It
    provides a new way to automate proofs in an interactive theorem prover
    like Coq.",
  paper = "Kell12.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kuma91,
  author = "Kumar, Ramayya and Kropf, Thomas and Schneider, Klaus",
  title =
    {{Integrating a First-Order Automatic Prover in the HOL Environment}},
  booktitle = "Int. Workshop on the HOL Theorem Prover System and its
               Applications",
  publisher = "IEEE Computer Society Press",
  year = "1991",
  abstract =
    "The HOL system is a powerful tool for proving higher-order formulae.
    However, proofs have to be performed interactively and only little
    automation using tactics is possible.  Even though interaction is
    desirable to guide major and creative backward proof steps of complex
    proofs, a deluge of simple sub-goals may evolve which all have to be
    proven manually in order to accomplish the proof.  Although these
    sub-goals are often simple formulae, their proof has not yet been
    automated in HOL.

    In this paper it is shown how it is possible to
    automate these tasks by integrating a first-order automated theorem
    proving tool, called FAUST, into HOL.  It is based on an efficient
    variant of the well-known sequent calculus.  In order to maintain the
    high confdence in HOL-generated proofs, FAUST is able to generate HOL
    tactics which may be used to post-justifr the theorem derived by FAUST
    in HOL.  The underlying calculus of FAUST, the tactic generation, as
    well as experimental results are presented.",
  paper = "Kuma91.pdf"
}
```

———-

— axiom.bib —

```
@article{Kutz86,
  author = "Kutzler, B. and Stifter, S.",
  title = {{On the application of Buchberger's algorithm to automated
           geometry theorem proving}},
  journal = "J. Symbolic Computation",
  volume = "2",
  number = "4",
  year = "1986",
  pages = "389-397",
  abstract =
    "In this paper we present a new approach to automated geometry theorem
    proving that is based on Buchberger's Grbner bases method. The goal
    is to automatically prove geometry theorems whose hypotheses and
    conjecture can be expressed algebraically, i.e. by polynomial
    equations. After shortly reviewing the problem considered and
    discussing some new aspects of confirming theorems, we present two
    different methods for applying Buehberger's algorithm to geometry
    theorem proving, each of them being more efficient than the other on a
    certain class of problems. The second method requires a new notion of
    reduction, which we call pseudoreduction. This pseudoreduction yields
    results on polynomials over some rational function field by
    computations that are done merely over the rationals and, therefore,
    is of general interest also. Finally, computing time statistics on 70
    non-trivial examples are given, based on an implementation of the
    methods in the computer algebra system SAC-2 on an IBM 4341.",
  paper = "Kutz86.pdf"
}
```

———-

1.19.10 L

— axiom.bib —

```
@book{Lamp02,
  author = "Lamport, Leslie",
  title = {{Specifying Systems}},
  year = "2002",
  link = "\url{http://research.microsoft.com/en-us/um/people/lamport/tla/book-02-08-08.pdf}",
  publisher = "Addison-Wesley",
  isbn = "0-321-14306-X",
  paper = "Lamp02.pdf"
}
```

———-

— axiom.bib —

```
@misc{Lamp13,
```

```
  author = "Lamport, Leslie",
  title = {{Errata to Specifying Systems}},
  year = "2013",
  link = "\url{http://research.microsoft.com/en-us/um/people/lamport/tla/errata-1.pdf}",
  publisher = "Microsoft",
  abstract = "
    These are all the errors and omissions to the first printing (July
    2002) of the book {\sl Specifying Systems} reported as of 29 October
    2013.  Positions in the book are indicated by page and line number,
    where the top line of a page is number 1 and the bottom line is number
    $-1$. A running head and a page number are not considered to be lines,
    but all other lines are. Please report any additional errors to the
    author, whose email address is posted on {\tt http://lamport.org}. The
    first person to report an error will be acknowledged in any revised
    edition.",
  paper = "Lamp13.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Lamp93,
  author = "Lamport, Leslie",
  title = {{How to Write a Proof}},
  year = "1993",
  publisher = "Microsoft",
  link =
  "\url{https://www.microsoft.com/en-us/research/uploads/prod/2016/12/How-to-Write-a-Proof.pdf}",
  paper = "Lamp93.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Lamp11,
  author = "Lamport, Leslie",
  title = {{Euclid Writes an Algorithm: A Fairytale}},
  year = "2011",
  link = "\url{http://lamport.azurewebsites.net/pubs/euclid.pdf}",
  abstract =
    "How Euclid might have written and checked the correctness of his
    famous algorithm had he been a little further ahead of his time.",
  paper = "Lamp11.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Lamp14,
  author = "Lamport, Leslie",
  title = {{How to Write a $21^{st}$ Century Proof}},
  year = "2014",
  link = "\url{http://lamport.azurewebsites.net/pubs/proof.pdf}",
  abstract = "
   A method of writing proofs is described that makes it harder to prove
   things that are not true. The method, based on hierarchical
   structuring, is simple and practical. The author's twenty years of
   experience writing such proofs is discussed.",
  paper = "Lamp14.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Lamp14a,
  author = "Lamport, Leslie",
  title = {{Talk: How to Write a $21^{st}$ Century Proof}},
  year = "2014",
  link = "\url{http://hits.mediasite.com/mediasite/Play/29d825439b3c49f088d35555426fbdf81d}",
  comment = "2nd Heidelberg Laureate Forum Lecture Tuesday Sep 23, 2014"
}
```

———-

— **axiom.bib** —

```
@misc{Lamp16,
  author = "Lamport, Leslie",
  title = {{TLA+ Proof System}},
  year = "2016",
  link = "\url{https://tla.msr-inria.inria.fr/tlaps/content/Documentation/Tutorial/The_example.html}",
  abstract = "Demonstration of Euclid Algorithm Proof in TLA+"
}
```

———-

— **axiom.bib** —

```
@article{Wony18,
  author = "Lee, Wonyeol and Sharma, Rahul and Aiken, Alex",
  title = {{On Automatically Proving the Correctness of math.h
            Implementation}},
  journal = "Proc. ACM Programming Languages",
  volume = "2",
  number = "42",
  year = "2018",
```

```
  pages = "1-32",
  abstract =
    "Industry standard implementations of math.h claim (often without
    formal proof) tight bounds on floating-point errors. We demonstrate a
    novel static analysis that proves these bounds and verifies the
    correctness of these implementations. Our key insight is a reduction
    of this verification task to a set of mathematical optimization
    problems that can be solved by off-the-shelf computer algebra
    systems. We use this analysis to prove the correctness of
    implementations in Intels math library automatically. Prior to this
    work, these implementations could only be verified with significant
    manual effort.",
  paper = "Wony18.pdf",
  keywords = "printed, reviewed"
}
```

———-

— **axiom.bib** —

```
@misc{Lond74,
  author = "London, Ralph L. and Musser, David R.",
  title = {{The Application of a Symbolic Mathematical System to Program
           Verification}},
  publisher = "USC Information Sciences Institute",
  year = "1974",
  abstract =
    "Program verification is a relatively new application area for
    symbolic mathematical systems.  We report on an interactive
    program verification system, based on the inductive assertion method,
    which system is implemented using an existing symbolic mathematical
    language and supporting system, Reduce.  Reduce has been augmented
    with a number of capabilities which are important to program
    verification, particularly transformations on relational and Boolean
    expressions.  We believe these capabilities would be valuable in other
    contexts and should be incorporated more widely into symbolic
    mathematical systems for general use.  The program verification
    application can serve as a guide to an appropriate definition of
    such capabilities, particularly with regard to the need to distinguish
    between undefined program variables and polynomial indeterminates.
    Additional capabilities which would benefit the program verification
    application include representation of user-defined functions by
    internal forms which directly incorporate properties such as
    commutativity and associativity (as is commonly done with plus and
    times), and a comprehensive facility for defining conditionally
    applicable transformations.",
  paper = "Lond74.pdf",
  keywords = "axiomref, printed"
}
```

———-

1.19.11 M

— **axiom.bib** —

```
@article{Mahb06,
  author = "Mahboubi, Assia",
  title = {{Proving Formally the Implementation of an Efficient gcd
           Algorithm for Polynomials}},
  journal = "Lecture Notes in Computer Science",
  volume = "4130",
  year = "2006",
  pages = "438-452",
  abstract = "
    We describe here a formal proof in the Coq system of the structure
    theorem for subresultants which allows to prove formally the
    correctness of our implementation of the subresultants algorithm.
    Up to our knowledge it is the first mechanized proof of this result.",
  paper = "Mahb06.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Mahb18,
  author = "Mahboubi, Assia and Tassi, Enrico and Bertot, Yves and
            Gonthier, Georges",
  title = {{Mathematical Components}},
  year = "2018",
  publisher = "math-comp.github.io/mcb",
  link = "\url{https://math-comp.github.io/mcb/book.pdf}",
  abstract =
    "{\sl Mathematical Components} is the name of a library of formalized
    mathematic for the COQ system. It covers a veriety of topics, from the
    theory of basic data structures (e.g. numbers, lists, finite sets) to
    advanced results in various flavors of algebra. This library
    constitutes the infrastructure for the machine-checked proofs of the
    Four Color Theorem and the Odd Order Theorem.

    The reason of existence of this books is to break down the barriers to
    entry. While there are several books around covering the usage of the
    COQ system and the theory it is based on, the Mathematical Components
    library is build in an unconventional way. As a consequence, this book
    provides a non-standard presentation of COQ, putting upfront the
    formalization choices and the proof style that are the pillars of the
    library.

    This book targets two classes of public. On one hand, newcomers, even
    the more mathematically inclined ones, find a soft introduction to the
    programming language of COQ, Gallina, and the Ssreflect proof
    language.  On the other hand accustomed COQ users find a substantial
    accound of the formalization style that made the Mathematical
    Components library possible.
```

```
    By no means does this book pretend to be a complete description of COQ
    or Ssreflect: both tools already come with a comprehensive user
    manual.  In the course of the book, the reader is nevertheless invited
    to experiment with a large library of formalized concepts and she is
    given as soon as possible sufficient tools to prove non-trivial
    mathematical results by reusing parts of the library. By the end of
    the first part, the reader has learnt how to prove formally the
    infinitude of prime numbers, or the correctnes of the Euclidean's
    division algorithm, in a few lines of proof text.",
  paper = "Mahb18.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Male17,
  author = "Maletzky, Alexander",
  title = {{A New Reasoning Framework for Theorema 2.0}},
  year = "2017",
  institution = "RISC Linz",
  abstract =
    "We present a new add-on for the Theorema 2.0 proof assistant,
    consisting of a reasoning framework in the spirit of (though not
    exactly as) the well-known LCF approach to theorem proving: a small,
    trusted kernel of basic inferences complemented by an extensive
    collection of automatic and interactive proof methods that construct
    proofs solely in terms of the basic inferences. We explain why such an
    approach is desirable in the first place in Theorema (at least as a
    possible alternative to the existing paradigm), how it fits together
    with the current default set-up of the system, and how proof-checking
    with the inference kernel of the new framework proceeds. Since all
    this is heavily inspired by the Isabelle proof assistant, we in
    particular also highlight the differences between Isabelle and our
    approach.",
  paper = "Male17.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Male16,
  author = "Maletzky, Alexander",
  title = {{Interactive Proving, Higher-Order Rewriting, and Theory Analysis
           in Theorema 2.0}},
  journal = "LNCS",
  volume = "9725",
  pages = "59-66",
```

```
  year = "2016",
  abstract =
    "In this talk we will report on three useful tools recently
    implemented in the frame of the Theorema project: a graphical user
    interface for interactive proof development, a higher-order
    rewriting mechanism, and a tool for automatically analyzing the
    logical structure of Theorema-theories. Each of these three tools
    already proved extremely useful in the extensive formal exploration of
    a non-trivial mathematical theory, namely the theory of Groeobner
    bases and reduction rings, in Theorema 2.0.",
  paper = "Male16.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Mart80,
  author = {Martin-L\"of, Per},
  title = {{Intuitionistic Type Theory}},
  link = "\url{http://archive-pml.github.io/martin-lof/pdfs/Bibliopolis-Book-retypeset-1984.pdf}",
  year = "1980",
  paper = "Mart80.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mart85,
  author = {Martin-L\"of, Per},
  title = {{Costructive Mathematics and Computer Programming}},
  booktitle = "Proc Royal Soc. of London on Math. Logic and Programming Lang.",
  link = "\url{http://www.cs.tufts.edu/~nr/cs257/archive/per-martin-lof/constructive-math.pdf}",
  year = "1985",
  isbn = "0-13-561465-1",
  pages = "168-184",
  publisher = "Prentice-Hall",
  paper = "Mart85.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mart96,
  author = {Martin-L\"of, Per},
  title = {{On the Meaning of the Logical Constants and the Justifications
           of the Logical Laws}},
  year = "1996",
```

```
  journal = "Nordic Journal of Philosophical Logic",
  volume = "1",
  number = "1",
  pages = "11-60",
  abstract =
    "The following three lectures were given in the form of a short course
    at the meeting Teoria della Dimonstrazione e Filosofia della Logica,
    organized in Siena, 6-9 April 1983, by the Scuola di Specializzazione
    in Logica Matematica of the Universit\`a degli Studi di Siena. I am
    very grateful to Giovanni Sambin and Aldo Ursini of that school, not
    only for recording the lectures on tape, but, above all, for
    transcribing the tapes produced by the recorder: no machine could have
    done that work. This written version of the lectures is based on their
    transcription. The changes that I have been forced to make have mostly
    been of a stylistic nature, except at one point. In the second
    lecture, as I actually gave it, the order of conceptual priority
    between the notions of proof and immediate inference was wrong. Since
    I discovered my mistake later the same month as the meeting was held,
    I thought it better to let the written text diverge from the oral
    presentation rather than possibly confusing others by letting the
    mistake remain. The oral origin of these lectures is the source of the
    many redundancies of the written text. It is also my sole excuse for
    the lack of detailed references.",
  paper = "Mart96.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Mart97,
  author = "Martin, Ursula and  Shand, D",
  title = {{Investigating some Embedded Verification Techniques for
           Computer Algebra Systems}},
  link =
    "\url{http://www.risc.jku.at/conferences/Theorema/papers/shand.ps.gz}",
  abstract = "
    This paper reports some preliminary ideas on a collaborative project
    between St. Andrews University in the UK and NAG Ltd. The project aims
    to use embedded verification techniques to improve the reliability and
    mathematical soundness of computer algebra systems. We give some
    history of attempts to integrate computer algebra systems and
    automated theorem provers and discuss possible advantages and
    disadvantages of these approaches. We also discuss some possible case
    studies.",
  paper = "Mart97.ps",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@book{Maso86,
  author = "Mason, Ian A.",
  title = {{The Semantics of Destructive Lisp}},
  publisher = "Center for the Study of Language and Information",
  year = "1986",
  isbn = "0-937073-06-7",
  abstract = "
    Our basic premise is that the ability to construct and modify programs
    will not improve without a new and comprehensive look at the entire
    programming process. Past theoretical research, say, in the logic of
    programs, has tended to focus on methods for reasoning about
    individual programs; little has been done, it seems to us, to develop
    a sound understanding of the process of programming -- the process by
    which programs evolve in concept and in practice. At present, we lack
    the means to describe the techniques of program construction and
    improvement in ways that properly link verification, documentation and
    adaptability.",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Maur73,
  author = "Maurer, D.",
  title = {{Applications of Symbolic Mathematical Manipulation to
            Algorithmic Verification (Abstract)}},
  journal = "SIGSAM Bulletin",
  volume = "26",
  year = "1973",
  pages = "4"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mcal96,
  author = "McAllester, D. and Arkondas, K.",
  title = {{Walther Recursion}},
  booktitle = "CADE 13",
  publisher = "Springer-Verlag",
  year = "1996",
  abstract =
    "Primitive recursion is a well known syntactic restriction on
    recursion definitions which guarantees termination. Unfortunately
    many natural definitions, such as the most common definition of
    Euclid's GCD algorithm, are not primitive recursive. Walther has
    recently given a proof system for verifying termination of a
    broader class of definitions. Although Walther's system is highly
    automatible, the class of acceptable definitions remains only
```

```
    semi-decidable. Here we simplify Walther's calculus and give a
    syntactic criteria generalizes primitive recursion and handles
    most of the examples given by Walthar. We call the corresponding
    class of acceptable defintions ``Walther recursive''.",
  paper = Mcal96.pdf
}
```

———-

— **axiom.bib** —

```
@article{Mcbr06,
  author = "McBride, Conor and Goguen, Healfdene and McKinna, James",
  title = {{A Few Constructions on Constructors}},
  journal = "Lecture Notes in Computer Science",
  volume = "3839",
  pages = "186-200",
  year = "2006",
  link = "\url{http://www.strictlypositive.org/concon.ps.gz}",
  abstract =
    "We present four constructions for standard equipment which can be
    generated for every inductive datatype: case analysis, structural
    recursion, no confusion, acyclicity. Our constructions follow a
    two-level approach -- they require less work than the standard
    techniques which inspired them. Moreover, given a suitably
    heterogeneous notion of equality, they extend without difficulty to
    inductive families of datatypes. These constructions are vital
    components of the translation from dependently typed programs in
    pattern matching style to the equivalent programs expressed in terms
    of induction principles and as such play a crucial behind-the-scenes
    role in Epigram.",
  paper = "Mcbr06.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mcca78,
  author = "McCarthy, John",
  title = {{A Micro-Manual for Lisp -- Not The Whole Truth}},
  journal = "ACM SIGPLAN Notices",
  volume = "13",
  number = "8",
  year = "1978",
  paper = "Mcca78.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mccu93,
  author = "McCune, William W.",
  title = {{Single Axioms for Groups and Abelian Groups with Various
           Operations}},
  journal = "J. Automated Reasoning",
  volume = "10",
  number = "1",
  year = "1993",
  abstract =
    "This paper summarizes the results of an investigation into single
    axioms for groups, both ordinary and Abelian, with each of following
    six sets of operations: \{product, inverse\}, \{division\}, \{double
    division, identity\}, \{double division, inverse\}, \{division,
    identity\} , and \{division, inverse\}. In all but two of the twelve
    corresponding theories, we present either the rst single axioms known
    to us or single axioms shorter than those previously known to us. The
    automated theorem-proving program Otter was used extensively to
    construct sets of candidate axioms and to search for and nd proofs
    that given candidate axioms are in fact single axioms.",
  paper = "Mccu93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Medi04,
  author = "Medina-Bulo, Inmaculada and Lozano-Palomo, F. and
            Alonso-Jimenez, J.A. and Ruiz-Reina, J.L.",
  title = {{Verified Computer Algebra in ACL2}},
  journal = "LNAI",
  volume = "3249",
  year = "2004",
  pages = "171-184",
  abstract =
    "In this paper, we present the formal verification of a Common Lisp
    implementation of Buchbergers algorithm for computing Grbner bases
    of polynomial ideals. This work is carried out in the Acl2 system and
    shows how verified Computer Algebra can be achieved in an executable
    logic.",
  paper = "Medi04.pdf"
}
```

———-

— **axiom.bib** —

```
@mastersthesis{Mein13,
  author = "Meindl, Diana",
  title = {{Implementation of an Algorithm Computing the Greatest
            Common Divisor for Multivariate Polynomials}},
```

```
  year = "2013",
  school = "RISC Linz"
}
```

———-

— **axiom.bib** —

```
@article{Melq12,
  author = "Melquiond, Guillaume",
  title = {{Floating-point arithmetic in the Coq system}},
  journal = "Information and Computation",
  volume = "216",
  pages = "14-23",
  year = "2012",
  link = "\url{https://www.lri.fr/~melquion/doc/08-mc8-article.pdf}",
  abstract =
    "The process of proving some mathematical theorems can be greatly
    reduced by relying on numerically-intensive computations with a
    certified arithmetic. This article presents a formalization of
    floating-point arithmetic that makes it possible to efficiently
    compute inside the proofs of the Coq system. This certified library is
    a multi-radix and multi-precision implementation free from underflow
    and overflow. It provides the basic arithmetic operators and a few
    elementary functions.",
  paper = "Melq12.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Mend87,
  author = "Mendelson, Elliot",
  title = {{Introduction to Mathematical Logic}},
  publisher = "Wadsworth and Brooks/Cole",
  year = "1987",
  isbn = "978-1482237726",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mesh01,
  author = "Meshveliani, Sergei D.",
  title = {{Computer Algebra with Haskell: Applying
           Functional-Categorical-'Lazy' Programming}},
  booktitle = "Computer Algebra and its Application to Physics",
```

```
  year = "2001",
  pages = "203-211",
  link =
    "\url{compalg.jinr.ru/Confs/CAAP\_2001/Final/proceedings/proceed.pdf}",
  abstract =
    "We give an outline of a computer algebra program writting in a
     functional language Haskell and implementing certain piece of
     commutative algebra",
  paper = "Mesh01.pdf",
  keywords = "axiomref, printed"

}
```

———-

— **axiom.bib** —

```
@misc{Mesh05,
  author = "Meshveliani, Sergei D.",
  title = {{Term rewriting, Equationional Reasoning, Automatic proofs}},
  link = "\url{ftp://ftp.botik.ru/pub/local/Mechveliani/dumatel/1.02/}",
  year = "2005",
  paper = "Mesh05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Mesh13,
  author = "Meshveliani, Sergei D.",
  title = {{Dependent Types for an Adequate Programming of Algebra}},
  link = "\url{http://ceur-ws.org/Vol-1010/paper-05.pdf}",
  year = "2013",
  abstract =
    "This research compares the authors experience in programming
    algebra in Haskell and in Agda (currently the former experience is
    large, and the latter is small). There are discussed certain hopes and
    doubts related to the dependently typed and verified programming of
    symbolic computation. This concerns the 1) authors experience
    history, 2) algebraic class hierarchy design, 3) proof cost overhead
    in evaluation and in coding, 4) other subjects. Various examples are
    considered.",
  paper = "Mesh13.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Mesh10,
  author = "Meshveliani, Sergei D.",
  title = {{Haskell and computer algebra}},
  link = "\url{http://www.botik.ru/pub/local/Mechveliani/basAlgPropos/haskellinCA2.pdf.zip}",
  year = "2010",
  abstract =
    "We consider the ways to program mathematics in the Haskell language.
    To start a discussion, we pretend to propose certain basic algebra
    library BAL for Haskell. We also mention several desirable language
    features. Algebraic additions in BAL are divided into the 'ordinary'
    and 'advanced'. Standard algebraic classes are reorganized to make
    them mathematically meaningful. For the 'advanced' part a sample
    argument approach is introduced -- as certain alternative for the
    dependent type language extension. The library is implemented in the
    existing Haskell, by 'hiding' a certain part of the existing Prelude.",
  paper = "Mesh10.pdf",
  keywords = "axiomref, printed, reviewed"
}
```

———-

— **axiom.bib** —

```
@article{Mesh14,
  author = "Meshveliani, Sergei D.",
  title = {{On dependent types and intuitionism in programming mathematics}},
  journal = "Program systems: theory and applications",
  year = "2014",
  volume = "5",
  numbrer = "3(21)",
  pages = "27-50",
  comment = "(In Russian)",
  link = "\url{http://psta.psiras.ru/read/psta2014_3_27-50.pdf}",
  abstract =
    "It is discussed a practical possibility of a provable programming
    of mathematics basing of the approach of intuitionism, a language
    with dependent types, proof-carrying code. This approach is
    illustrated with examples. The discourse bases on the experience
    of implementing in the {\tt Agda} language of a certain small
    algebraic library including the arithmetic of a residue domain
    $R/(b)$ for an arbitrary Euclidean ring R. (In Russian)",
  paper = "Mesh14.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Mesh15,
  author = "Meshveliani, Sergei D.",
  title = {{Programming basic computer algebra in a language with
```

```
          dependent types}},
  journal = "Program systems: theory and applications",
  year = "2015",
  volume = "6",
  numbrer = "4(27)",
  pages = "313-340",
  comment = "(In Russian)",
  link = "\url{http://psta.psiras.ru/read/psta2015_4_313-340.pdf}",
  abstract =
    "It is described the experience in provable programming of certain
    classical categories of computational algebra (``group'', ``ring'',
    and so on) basing on the approach of intuitionism, a language with
    dependent types, forming of machine-checked proofs. There are detected
    the related problems, and are described certain additional possibilities
    given by the approach. The {\tt Agda} functional language is used as an
    instrument. This paper is a continuation for the introductory paper
    published in this journal in 2014. (In Russian)",
  paper = "Mesh15.pdf",
  keywords = "CAS-Proof"
}
```

———-

— **axiom.bib** —

```
@book{Mesh16,
  author = "Meshveliani, Sergei D.",
  title = {{DoCon -- A Provable Algebraic Domain Constructor}},
  link =
    "\url{http://www.botik.ru/pub/local/Mechveliani/docon-A/0.04/manual.pdf}",
  publisher = "User Manual, Version 0.04",
  year = "2016",
  abstract =
    "This book is about 1) a manual on the DoCon-A program library, 2) a book
    explaining how to program algebra in a purely functional language with
    {\sl dependent types} (specifially, in {\tt Agda}), with providing
    machine-checked proofs, and following constructive mathematics.

    The above point of proofs means that a program not only implements an
    algorithm, but explains to the compiler the needed mathematical notions
    and provides the needed proofs in the form of type expressions and
    functions. And the compiler (more precisely, type checker) is able to
    verify these proofs statically (before running), and to prepare the
    algorithm for running.",
  paper = "Mesh16,pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Mesh16a,
  author = "Meshveliani, Sergei D.",
  title = {{Provable programming of algebra: particular points, examples}},
  link = "\url{http://www.botik.ru/pub/local/Mechveliani/provProgExam.zip}",
  year = "2016",
  abstract =
    "It is discussed an experiance in provable programming of a computer
    algebra library with using a purely functional language with dependent
    tyhpes ({\tt Agda}). There are given several examples illustrating
    particular points of implementing the approach of constructive
    mathematics.",
  paper = "Mesh16a.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Mitc88,
  author = "Mitchell, John C. and Plotkin, Gordon D.",
  title = {{Abstract types have existential type}},
  journal = "ACM TOPLAS",
  volume = "10",
  number = "3",
  year = "1988",
  pages = "470-502",
  abstract =
    "Abstract data type declarations appear in typed programming languages
    like Ada, Alphard, CLU and ML. This form of declaration binds a list
    of identifiers to a type with associated operations, a composite
    value we call a data algebra. We use a second-order typed lambda
    calculus SOL to show how data algebras may be given types, passed as
    parameters, and returned as results of function calls. In the process,
    we discuss the semantics of abstract data type declarations and review
    a connection between typed programming languages and constructive
    logic.",
  paper = "Mitc88.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Miln72,
  author = "Milner, Robert",
  title = {{Logic for Computable Functions: Description of a Machine
           Implementation}},
  year = "1972",
  institution = "Stanford Artificial Intelligence Project",
  number = "STAN-CS-72-288",
  link =
```

```
 "\url{http://i.standford.edu/pub/cstr/reports/cs/tr/72/288/CS-TR-72-288.pdf}",
  abstract =
    "LCF is based on a logic by Dana Scott, proposed by him at Oxford in the
    fall of 1969, for reasoning about computable functions.",
  paper = "Miln72.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Miln72a,
  author = "Milner, Robin",
  title = {{Implementation and Applications of Scott's Logic for
            Computable Functions}},
  journal = "ACM SIGPLAN Notices",
  volume = "7",
  number = "1",
  year = "1972",
  pages = "1-6",
  abstract =
    "The basis for this paper is a logic designed by Dana Scott[1] in
    1969 for formalizing arguments about computable functions of
    higher type. This logic uses typed combinators, and we give a more
    or less direct translation into typed $\lambda$-calculus, which is
    an easier formalism to use, though not so easy for the metatheory
    because of the presence of bound variables. We then describe, by
    example only, a proof-checker program which has been implemented
    for this logic; the program is fully describe in [2]. We relate
    the induction rule which is central to the logic to two more
    familiar rules -- Recursion Induction and Structural Induction --
    showing that the former is a theorem of the logic, and that for
    recursively defined structures the latter is a derived rule of the
    logic. Finally we show how the syntax and semantics of a simple
    programming language may be described completely in the logic, and
    we give an example of a theorem which relates syntactic and
    semantic properties of programs and which can be stated and proved
    within the logic."
}
```

———-

— **axiom.bib** —

```
@article{Miln78,
  author = "Milner, Robin",
  title = {{A Theory of Type Polymorphism in Programming}},
  year = "1978",
  journal = "Journal of Computer and System Sciences",
  volume = "17",
  pages = "348-375",
```

```
  link = "\url{https://courses.engr.illinois.edu/cs421/sp2013/project/milner-polymorphism.pdf}",
  abstract =
    "The aim of this work is largely a practical one.  A widely employed
    style of programming, particularly in structure-processing languages
    which impose no discipline of types, entails defining procedures which
    work well on objects of a wide variety.  We present a formal type
    discipline for such polymorphic procedures in the context of a simple
    programming language, and a compile time type-checking algorithm $W$
    which enforces the discipline.  A Semantic Soundness Theorem (based on
    a formal semantics for the language) states that well-type programs
    cannot go wrong and a Syntactic Soundness Theorem states that if $W$
    accepts a program then it is well typed.  We also discuss extending
    these results to richer languages; a type-checking algorithm based on
    $W$ is in fact already implemented and working, for the metalanguage ML
    in the Edinburgh LCF system,",
  paper = "Miln78.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Miln84,
  author = "Milner, R.",
  title = {{The Use of Machines to Assist in Rigorous Proof}},
  journal = "Philosophical Transactions of the Royal Society",
  volume = "312",
  pages = "411-422",
  number = "1522",
  year = "1984",
  abstract =
    "A methodology for computer assisted proof is presented with an
    example. A central ingredient in the method is the presentation of
    tactics (or strategies) in an algorithmic metalanguage. Further, the
    same language is also used to express combinators, by which simple
    elementary tactics - which often correspond to the inference rules of
    the logic employed - are combined into more complex tactics, which may
    even be strategies complete for a class of problems. However, the
    emphasis is not upon completeness but upon providing a metalogical
    framework within which a user may express his insight into proof
    methods and may delegate routine (but error-prone) work to the
    computer. This method of tactic composition is presented at the start
    of the paper in the form of an elementary theory of goal-seeking. A
    second ingredient of the methodology is the stratification of
    machine-assisted proof by an ancestry graph of applied theories, and
    the example illustrates this stratification. In the final section,
    some recent developments and applications of the method are cited.",
  paper = "Miln84.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Mohr14,
  author = "Mohring-Paulin, Christine",
  title = {{Introduction to the Calculus of Inductive Constructions}},
  year = "2014",
  link = "\url{https://hal.inria.fr/hal-01094195/file/CIC.pdf}",
  paper = "Mohr14.pdf"
}
```

———-

— **axiom.bib** —

```
@mastersthesis{Mort10,
  author = {M\"ortbert, Anders},
  title = {{Constructive Algebra in Functional Programming and Type Theory}},
  school = "University of Gothenburg, Department of Computer Science",
  year = "2010",
  month = "5",
  file = "Mort10.pdf",
  abstract =
    "This thesis considers abstract algebra from a constructive point
    of view. The central concept of study is coherent rings -- algebraic
    structures in which it is possible to solve homogeneous systems of
    linear equations. Three different algebraic theories are considered;
    Bezout domains, Prufer domains and polynomial rings. The first two
    of these are non-Noetherian analogues of classical notions. The
    polynomial rings are presented from a constructive point of view with a
    treatment of Groebner bases. The goal of the thesis is to study the
    proofs that these theories are coherent and explore how the proofs can
    be implemented in functional programming and type theory."
}
```

———-

— **axiom.bib** —

```
@misc{Mour16,
  author = "de Moura, Leonardo and Avigad, Jeremy and Kong, Soonho and
            Roux, Cody",
  title = {{Elaboration in Dependent Type Theory}},
  year = "2016",
  link = "\url{http://leodemoura.github.io/files/elaboration.pdf}",
  abstract =
    "We describe the elaboration algorithm that is used in {\sl Lean}, a
    new interactive theorem prover based on dependent type theory. To be
    practical, interactive theorem provers must provide mechanisms to
    resolve ambiguities and infer implicit type information, thereby
    supporting convenient input of expressions and proofs. Lean's
    elaborator supports higher-order unification, ad-hoc overloading,
    insertion of coercions, type class inference, the use of tactics, and
```

```
    the computational reduction of terms.  The interactions between these
    components are subtle and complex, and Lean's elaborator has been
    carefully designed to balance efficiency and usability.",
  paper = "Mour16.pdf",
  keywords = "coercion"
}
```

———-

1.19.12 N

— **axiom.bib** —

```
@article{Naum06,
  author = "Naumowicz, Adam",
  title = {{An Example of Formalizing Recent Mathematical Results in Mizar}},
  journal = "Journal of Applied Logic",
  volume = "4",
  number = "4",
  year = "2006",
  pages = "396-413",
  abstract =
    "This paper describes an example of the successful formalization of
    quite advanced and new mathematics using the Mizar system. It shows
    that although much effort is required to formalize nontrivial facts in
    a formal computer deduction system, still it is possible to obtain the
    level of full logical correctness of all inference steps. We also
    discuss some problems encountered during the formalization, and try to
    point out some of the features of the Mizar system responsible for
    that. The formalization described in this paper allows also for
    contrasting the linguistic capability of the Mizar language and some
    of the phrases commonly used in informal mathematical papers that
    the Mizar system lacks, and consequently presents the methods of how
    to cope with it during the formalization. Yet, apart from the
    problems, this paper shows some definite benefits from using a formal
    computer system in the work of a mathematician.",
  paper = "Naum06.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Neup13,
  author = "Neuper, Walther",
  title = {{Computer Algebra implemented in Isabelle's Function Package
           under Lucas-Interpretation -- a Case Study}},
  year = "2013",
  link = "\url{http://ceur-ws.org/Vol-1010/paper-09.pdf}",
  paper = "Neup13.pdf",
  keywords = "CAS-Proof, printed, DONE"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Newc13,
  author = "Newcombe, Chris and  Rath, Tim and  Zhang, Fan and
            Munteanu, Bogdan and  Brooker, Marc and Deardeuff, Michael",
  title = {{Use of Formal Methods at Amazon Web Services}},
  link = "\url{http://research.microsoft.com/en-us/um/people/lamport/tla/formal-methods-amazon.pdf}",
  abstract =
    "In order to find subtle bugs in a system design, it is necessary to
    have a precise description of that design. There are at least two
    major benefits to writing a precise design; the author is forced to
    think more clearly, which helps eliminate ‘‘plausible hand-waving’’,
    and tools can be applied to check for errors in the design, even while
    it is being written. In contrast, conventional design documents
    consist of prose, static diagrams, and perhaps pseudo-code in an ad
    hoc untestable language. Such descriptions are far from precise; they
    are often ambiguous, or omit critical aspects such as partial failure
    or the granularity of concurrency (i.e. which constructs are assumed
    to be atomic). At the other end of the spectrum, the final executable
    code is unambiguous, but contains an overwhelming amount of detail. We
    needed to be able to capture the essence of a design in a few hundred
    lines of precise description. As our designs are unavoidably complex,
    we need a highly-expressive language, far above the level of code, but
    with precise semantics. That expressivity must cover real-world
    concurrency and fault-tolerance. And, as we wish to build services
    quickly, we wanted a language that is simple to learn and apply,
    avoiding esoteric concepts. We also very much wanted an existing
    ecosystem of tools.  We found what we were looking for in TLA+, a
    formal specification language."
}
```

———-

— **axiom.bib** —

```
@book{Nipk14,
  author = "Nipkow, Tobias and Klein, Gerwin",
  title = {{Concrete Semantics}},
  isbn = "978-3-10542-0",
  publisher = "Springer",
  year = "2014",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@article{Nguy16,
  author = "Nguyen, Phuc C. and Tobin-Hochstadt, Sam and van Horn, David",
  title = {{Higher-order symbolic execution for contract verification and
           refutation}},
  journal = "arXiv",
  link = "\url{http://arxiv.org/pdf/1507.04817v2.pdf}",
  year = "2016",
  month = "February",
  abstract =
    "We present a new approach to automated reasoning about higher-order
    programs by endowing symbolic execution with a notion of higher-order,
    symbolic values.

    To validate our approach, we use it to develop and evaluate a system
    for verifying and refuting behavioral software contracts of components
    in a functional language, which we call {\sl soft contract
    verification}. In doing so, we discover a mutually beneficial relation
    between behavioral contracts and higher-order symbolic
    execution. Contracts aid symbolic execution by providing a rich
    language of specifications that can serve as the basis of symbolic
    higher-order values; the theory of blame enables modular verification
    and leads to the theorem that {\sl verified compnents can't be
    blamed}; and the run-time monitoring of contracts enables {\sl soft}
    verification whereby verified and unverified components can safely
    interact and verification is not an all-or-nothing
    proposition. Conversely, symbolic execution aids contracts by
    providing compile-time verification which increases assurance and
    enables optimizations; automated test-case generation for contracts
    with counter-examples; and engendering a virtuous cycle between
    verification and the gradual spread of contracts.

    Our system uses higher-order symbolic execution, leveraging contracts
    as a source of symbolic values including unknown behavioral values,
    and employs an updatable heap of contract invariants to reason about
    flow-sensitive facts. Whenever a contract is refuted, it reports a
    concrete {\sl counterexample} reproducing the error, which may involve
    solving for an unknown function. The approach is able to analyze
    first-class contracts, recursive data structures, unknown functions,
    and control-flow-sensitive refinement of values, which are all
    idiomatic in dynamic languages. It makes effective use of an
    off-the-shelf solver to decide problems without heavy encodings. Our
    counterexample search is sound and relatively complete with respect to
    a first-order solver for base type values. Therefore, it can form the
    basis of automated verification and bug-finding tools for higher-order
    programs. The approach is competitive with a wide range of existing
    tools -- including type systems, flow analyzers, and model checkers --
    on their own benchmarks. We have built a tool which analyzes programs
    written in Racket, and report on its effectiveness in verifying and
    refuting contracts.",
  paper = "Nguy16.pdf"
}
```

— **axiom.bib** —

```
@book{Nord90,
  author = {Nordstr\"om, Bengt and Petersson, Kent and Smith, Jan M.},
  title = {{Programming in Martin-L\"of's Type Theory}},
  year = "1990",
  publisher = "Oxford University Press",
  paper = "Nord90.pdf",
  keywords = "printed"
}
```

———-

1.19.13 O

— **axiom.bib** —

```
@misc{OCon15,
  author = {O'Connor, Liam},
  title = {{Write Your Compiler by Proving It Correct}},
  year = "2015",
  link = "\url{http://liamoc.net/posts/2015-08-23-verified-compiler.html}",
  abstract =
    "Recently my research has been centered around the development of a
    self-certifying compiler for a functional language with linear types
    called Cogent (see O'Connor et al. [2016]). The compiler works by
    emitting, along with generated low-level code, a proof in Isabelle/HOL
    (see Nipkow et al. [2002]) that the generated code is a refinement of
    the original program, expressed via a simple functional semantics in HOL.

    As dependent types unify for us the language of code and proof, my
    current endeavour has been to explore how such a compiler would look
    if it were implemented and verified in a dependently typed programming
    language instead. In this post, I implement and verify a toy compiler
    for a language of arithmetic expressions and variables to an idealised
    assembly language for a virtual stack machine, and explain some of the
    useful features that dependent types give us for writing verified
    compilers."
}
```

———-

— **axiom.bib** —

```
@article{Owre92,
  author = "Owre, S. and Rushby, J.M. and Shankar, N.",
  title = {{PVS: A Prototype Verification System}},
  journal = "Lecture Notes in Computer Science",
  volume = "687",
  pages = "748-752",
  year = "1992",
```

```
  abstract =
    "This brief paper introduces the main ideas of PVS",
  paper = "Owre92.pdf"
}
```

———-

1.19.14 P

— **axiom.bib** —

```
@article{Pada80,
  author = "Padawitz, Peter",
  title = {{New results on completeness and consistency of abstract data
           types}},
  journal = "LNCS",
  volume = "88",
  pages = "460-473",
  year = "1980",
  abstract =
    "If an algebraic specification is designed in a structured way, a
    small specification is stepwise enriched by more complex operations
    and their defining equations. Based on normalization properties of
    term reductions we present sufficient ``local'' conditions for the
    completeness and consistency of enrichment steps, which can be
    efficiently verified in many cases where other attempts to prove the
    enrichment property ``syntactically'' have failed so far."
}
```

———-

— **axiom.bib** —

```
@book{Paol02,
  author = "Paoli, Francesco",
  title = {{Substructural Logics: A Primer}},
  publisher = "Springer",
  isbn = "978-90-481-6014-3",
  year = "2002",
  abstract =
    "Substructural logics are by now one of the most prominent branches of
    the research field usually labelled as ``nonclassical logics'' - and
    perhaps of logic tout court. Over the last few decades a vast amount
    of research papers and even some books have been devoted to this
    subject. The aim of the present book is to give a comprehensive
    account of the ``state of the art'' of substructural logics, focusing
    both on their proof theory (especially on sequent calculi and their
    generalizations) and on their semantics (both algebraic and relational).",
  paper = "Paol02.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pare93,
  author = "Parent, Catherine",
  title = {{Developing Certified Programs in the System Coq: The Program
           Tactic}},
  booktitle = "Proc. Int. Workshop on Types for Proofs and Programs",
  publisher = "Springer-Verlag",
  isbn = "3-540-58085-9",
  pages = "291-312",
  year = "1993",
  abstract =
    "The system {\sl Coq} is an environment for proof development based on
    the Calculus of Constructions extended by inductive definitions. The
    specification of a program can be represented by a logical formula and
    the program itself can be extracted from the constructive proof of the
    specification. In this paper, we look at the possibility of inverting
    the specification and a program, builds the logical condition to be
    verified in order to obtain a correctness proof of the program. We
    build a proof of the specification from the program from which the
    program can be extracted. Since some information cannot automatically
    be inferred, we show how to annotate the program by specifying some of
    its parts in order to guide the search for the proof.",
  paper = "Pare93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Pare94,
  author = "Parent, Catherine",
  title = {{Synthesizing proofs from programs in the Calculus of Inductive
           Constructions}},
  year = "1994",
  institution = {Ecole Normale Sup\'erieure de Lyon},
  abstract =
    "In type theory, a proof can be represented as a typed $\lambda$-term.
    There exist methods to mark logical parts in proofs and extract their
    algorithmic contents. The result is a correct program with respect to
    a specification. This paper focuses on the inverse problem: how to
    generate a proof from its specification. The framework is the Calculus
    of Inductive Constructions. A notion of coherence is introduced between
    a specification and a program containing types but no logical proofs.
    This notion is based on the definition of an extraction function called
    the weak extraction. Such a program can give a method to reconstruct a
    set of logical properties needed to have a proof of the initial
    specification. This can be seen either as a method of proving programs
    or as a method of synthetically describing proofs.",
  paper = "Pare94.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Pare96,
  author = "Parent-Vigouroux, Catherine",
  title = {{Natural proofs versus programs optimization in the
           Calculus of Inductive Constructions}},
  year = "1996",
  abstract =
    "This paper presents how to automatically prove that an 'optimized'
    program is correct with respect to a set of given properties that is a
    specification. Proofs of specifications contain logical and
    computational parts. Programs can be seen as computational parts of
    proofs. They can thus be extracted from proofs and be certified to be
    correct. The inverse problem can be solved: it is possible to
    reconstruct proof obligations from a program and its specification.
    The framework is a type theory where a proof can be represented as a
    typed $\lambda$-term and, particularly, the Calculus of Inductive
    Constructions. This paper shows how programs can be simplified in
    order to be written in a much closer way to the ML one's. Indeed,
    proofs structures are often much more heavy than program structures.
    The problem is consequently to consider natural programs (in a ML sense)
    and see how to retrieve natural structures of proofs from them.",
  paper = "Pare96.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Pare97,
  author = "Parent-Vigouroux, Catherine",
  title = {{Verifying programs in the Calculus of Inductive Constructions}},
  year = "1997",
  journal = "Formal Aspects of Computing",
  volume = "9",
  number = "5",
  pages = "484-517",
  abstract =
    "This paper deals with a particular approach to the verification of
    functional programs. A specification of a program can be represented
    by a logical formula. In a constructive framework, developing a program
    then corresponds to proving this formula. Given a specification and a
    program, we focus on reconstructing a proof of the specification whose
    algorithmic contents corresponds to the given program. The best we can
    hope is to generate proof obligations on atomic parts of the program
    corresponding to logical properties to be verified. First, this paper
    studies a weak extraction of a program from a proof that keeps track
    of intermediate specifications. From such a program, we prove the
```

```
    determinism of retrieving proof obligations. Then, heuristic methods
    are proposed for retrieving the proof from a natural program containing
    only partial annotations. Finally, the implementation of this methos as
    a tactic of the {\sl Coq} proof assistant is presented.",
  paper = "Pare97.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Paul93,
  author = "Paulin-Mohring, Christine",
  title = {{Inductive Definitions in the system Coq -- Rules and
           Properties}},
  booktitle = "Proc '93 Int. Conf. on Typed Lambda Calculi and
              Applications",
  year = "1993",
  pages = "328-345",
  isbn = "3-540-56517-5",
  abstract =
    "In the pure Calculus of Constructions, it is possible to represent
    data structures and predicates using higher-order
    quantification. However, this representation is not satisfactory, from
    the point of view of both the efficiency of the underlying programs
    and the power of the logical system. For these reasons, the calculus
    was extended with a primitive notion of inductive definitions
    [8]. This paper describes the rules for inductive definitions in the
    system Coq. They are general enough to be seen as one formulation of
    adding inductive definitions to a typed lambda-calculus. We prove
    strong normalization for a subsystem of Coq corresponding to the pure
    Calculus of Constructions plus Inductive Definitions with only weak
    eliminations"
}
```

———-

— **axiom.bib** —

```
@book{Paul94,
  author = "Paulson, Lawrence C.",
  title = {{ISABELLE: A Generic Theorem Prover}},
  year = "1994",
  publisher = "Springer-Verlag",
  isbn = "978-3-540-58244-1",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Paul98,
  author = "Paulson, Lawrence C.",
  title = {{Introduction to Isabelle}},
  publisher = "Computer Laboratory, Univ. of Cambridge",
  year = "1998"
}
```

— axiom.bib —

```
@article{Pela14,
  author = "Pelayo, Alvaro and Warren, Michael A.",
  title = {{Homotopy Type Theory and Voevodsky's Univalent Foundations}},
  journal = "Bulletin of the American Mathematical Society",
  volume = "51",
  number = "4",
  year = "2014",
  pages = "597-648",
  link = "\url{https://arxiv.org/pdf/1210.5658.pdf}",
  abstract =
    "Recent discoveries have been made connecting abstract homotopy
    theory and the field of type theory from logic and theoretical computer
    science. This has given rise to a new field, which has been christened
    {\sl homotopy type theory}. In this direction, Vladimir Voevodsky
    observed that it is possible to model type theory using simpical sets
    and that this model satisfies an additional property, called the
    {\sl Univalence Axiom}, which has a number of striking consequences.
    He has subsequently advocated a program, which he calls {\sl univalent
    foundations}, of developing mathematics in the setting of type theory
    with the Univalence Axiom and possibly other additional axioms motivated
    by the simplical set model. Because type theory posses good computational
    properties, this program can be carried out in a computer proof assistant.
    In this paper we give an introduction to homotopy type theory in
    Voevodsky's setting, paying attention to both theoretical and practical
    issues. In particular, the paper serves as an introduction to both the
    general ideas of homotopy type theory as well as to some of the concrete
    details of Voevodsky's work using the well-known proof assistant Coq.
    The paper is written for a general audience of mathematicians with basic
    knowledge of algebraic topology; the paper does not assume any
    preliminary knowledge of type theory, logic, or computer science. Because
    a defining characteristic of Voevodsky's program is that the Coq code has
    fundamental mathematical content, and many of the mathematical concepts
    which are efficiently captured in the code cannot be explained in
    standard mathematical English without a length detour through type theory,
    the later sections of this paper (beginning with Section 3) make use of
    code; however, all notions are introduced from the beginning and in a
    self-contained fashion.",
  paper = "Pela14.pdf"
}
```

— **axiom.bib** —

```
@inproceedings{Pell91,
  author = "Pelletier, Francis Jeffry",
  title = {{The Philosophy of Automated Theorem Proving}},
  booktitle = "Proc 12th IJCAI",
  pages = "538-543",
  year = "1991",
  publisher = "Morgan Kaufmann",
  paper = "Pell91.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pfen88,
  author = "Pfenning, Frank",
  title = {{Partial Polymorphic Type Inference and Higher-Order Unification}},
  booktitle = "Proc 1988 ACM Conf. on Lisp and Functional Programming",
  pages = "153-163",
  year = "1988",
  publisher = "ACM",
  isbn = "0-89791-273-X",
  abstract =
    "We show that the problem of partial type inference in the $n$-th
    order polymorphic $\lambda$-calculus is equivalent to $n$-th order
    unification. On the one hand, this means that partial type inference
    in polymorphic $\lambda$-calculi of order 2 or higher is
    undecidable. On the other hand, higher-order unification is often
    tractable in practice, and our translation entails a very useful
    algorithm for partial type inference in the $omega$-order polymorphic
    $\lambda$-calculus. We present an implementation in $\lambda$Prolog in
    full.",
  paper = "Pfen88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Pfen89,
  author = "Pfenning, Frank and Paulin-Mohring, Christine",
  title = {{Inductively Defined Types in the Calculus of Constructions}},
  institution = "Carnegie-Mellon University",
  year = "1989",
  number = "CMU-CS-89-209",
  link = "\url{http://repository.cmu.edu/cgi/viewcontent.cgi?article=2907&context=compsci}",
  abstract =
    "We define the notion of an {\sl inductively defined type} in the
```

```
    Calculus of Constructions and show how inductively defined types can
    be represented by closed types. We show that all primitive recursive
    functional over these inductively defined types are also
    representable. This generalizes work by Bohm and Berarducci on
    synthesis of functions on term algebras in the second-order
    polymorphic $\lambda$-calculus ($F_2$). We give several applications
    of this generalization, including a representation of $F_2$-programs
    in $F_3$, along with a definition of functions {\bf reify}, {\bf
    reflect}, and {\bf eval} for $F_2$ in $F_3$. We also show how to
    define induction over inductively defined types and sketch some
    results that show that the extension of the Calculus of Construction
    by induction principles does not alter the set of functions in its
    computational fragment, $F_\omega$. This is because a proof by
    induction can be {\bf realized} by primitive recursion, which is
    already definable in $F_\omega$.",
  paper = "Pfen89.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen17,
  author = "Pfenning, Frank",
  title = {{Logical Frameworks}},
  link = "\url{http://www.cs.cmu.edu/afs/cs.cmu.edu/user/fp/www/lfs.html}",
  year = "2017"
}
```

———-

— **axiom.bib** —

```
@incollection{Pfen92a,
  author = "Pfenning, Frank",
  title = {{Dependent Types in Logic Programming}},
  booktitle = "Types in Logic Programming",
  isbn = "9780262161312",
  publisher = "MIT Press",
  year = "1992",
  paper = "Pfen92a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Pier98,
  author = "Pierce, Benjamin C. and Turner, David N.",
  title = {{Local Type Inference}},
  year = "1998",
```

```
  link =
  "\url{http://www.cis.upenn.edu/~bcpierce/papers/lti-toplas.pdf}",
  abstract =
    "We study two partial type inference methods for a language
    combining subtyping and impredicative polymorphism. Both methods
    are local in the sense that missing annotations are recovered
    using only information from adjacent nodes in the syntax tree,
    without long-distance constraints such as unification
    variables. One method infers type arguments in polymorphic
    applications using a local constraint solver. The other infers
    annotations on bound variables in function abstractions by
    propagating type constraints downward from enclosing application
    nodes. We motivate our design choices by a statistical analysis of
    the uses of type inference in a sizable body of existing ML code.",
  paper = "Pier98.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Pier00,
  author = "Pierce, Benjamin C.",
  title = {{Type Systems for Programming Languages}},
  year = "2000",
  publisher = "MIT Press",
  link = "\url{http://ropas.snu.ac.kr/~kwang/S20/pierce\_book.pdf}",
  paper = "Pier00.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Pier15,
  author = {Pierce, Benjamin C. and Casinghino, Chris and Gaboardi, Marco and
     Greenberg, Michael and Hritcu, Catalin and Sj\"oberg, Vilhelm and
     Yorgey, Brent},
  title = {{Software Foundations}},
  year = "2015",
  file = "Pier15.tgz",
  abstract =
    "This electronic book is a course on Software Foundations, the
    mathematical underpinnings of reliable software. Topics include basic
    concepts of logic, computer-assisted theorem proving, the Coq proof
    assistant, functional programming, operational semantics, Hoare logic,
    and static type systems. The exposition is intended for a broad range
    of readers, from advanced undergraduates to PhD students and
    researchers. No specific background in logic or programming languages
    is assumed, though a degree of mathematical maturity will be helpful.
```

```
    The principal novelty of the course is that it is one hundred per cent
    formalized and machine-checked: the entire text is literally a script
    for Coq. It is intended to be read alongside an interactive session
    with Coq. All the details in the text are fully formalized in Coq, and
    the exercises are designed to be worked using Coq.

    The files are organized into a sequence of core chapters, covering
    about one semester's worth of material and organized into a coherent
    linear narrative, plus a number of appendices covering additional
    topics. All the core chapters are suitable for both upper-level
    undergraduate and graduate students."
}
```

———-

— **axiom.bib** —

```
@misc{Pier19a,
  author = {Pierce, Benjamin C. and Casinghino, Chris and Gaboardi, Marco and
     Greenberg, Michael and Hritcu, Catalin and Sj\"oberg, Vilhelm and
     Yorgey, Brent},
  title = {{Programming Language Foundations}},
  year = "2019",
  file = "Pier19a.tgz",
  abstract =
    "This electronic book is a course on Software Foundations, the
    mathematical underpinnings of reliable software. Topics include basic
    concepts of logic, computer-assisted theorem proving, the Coq proof
    assistant, functional programming, operational semantics, Hoare logic,
    and static type systems. The exposition is intended for a broad range
    of readers, from advanced undergraduates to PhD students and
    researchers. No specific background in logic or programming languages
    is assumed, though a degree of mathematical maturity will be helpful.

    The principal novelty of the course is that it is one hundred per cent
    formalized and machine-checked: the entire text is literally a script
    for Coq. It is intended to be read alongside an interactive session
    with Coq. All the details in the text are fully formalized in Coq, and
    the exercises are designed to be worked using Coq.

    The files are organized into a sequence of core chapters, covering
    about one semester's worth of material and organized into a coherent
    linear narrative, plus a number of appendices covering additional
    topics. All the core chapters are suitable for both upper-level
    undergraduate and graduate students."
}
```

———-

— **axiom.bib** —

```
@article{Piro05,
```

```
  author = "Piroi, Florina and Kutsiz, Temur",
  title = {{The Theorema Environment for Interactive Proof Development}},
  journal = "LNAI",
  volume = "3835",
  pages = "261-275",
  year = "2005",
  abstract =
    "We describe an environment that allows the users of the Theorema
    system to flexibly control aspects of computer-supported proof
    development. The environment supports the display and manipulation of
    proof trees and proof situations, logs the user activities (commands
    communicated with the system during the proving session), and presents
    (also unfinished) proofs in a human-oriented style. In particular, the
    user can navigate through the proof object, expand/remove proof
    branches, provide witness terms, develop several proofs concurrently,
    proceed step by step or automatically and so on. The environment
    enhances the effectiveness and flexibility of the reasoners of the
    Theorema system.",
  paper = "Piro05.pdf, printed"
}
```

———-

— axiom.bib —

```
@article{Plot77,
  author = "Plotkin, G.D.",
  title = {{LCF Considered as a Programming Language}},
  journal = "Theoretical Computer Science",
  volume = "5",
  year = "1977",
  pages = "223-255",
  link = "\url{http://homepages.inf.ed.ac.uk/gdp/publications/LCF.pdf}",
  abstract =
    "The paper studies connections between denotational and operational
    semantics for a simple programming language based on LCF. It begins
    with the connection between the behaviour of a program and its
    denotation. It turns out that a program denotes $\bot$ in any of
    several possible semantics iff it does not terminate. From this it
    follows that if two terms hae the same denotation in one of these
    semantics, they have the same behaviour in all contexts. The converse
    fails for all the semantics. If, however, the language is extended to
    allow certain parallel facilities behavioural equivalence does coincide
    with denotational equivalence in one of the semantics considered, which
    may therefore be called ``fully abstract''. Next a connection is given
    which actually determines the semantics up to isomorphism from the
    behaviour alone. Conversely, by allowing further parallel facilities,
    every r.e. element of the fully abstract semantics becomes definable,
    thus characterising the programming language, up to interdefinability,
    from the set of r.e. elements of the domains of the semantics.",
  paper = "Plot77.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Poll98,
  author = "Poll, Erik and Thompson, Simon",
  title = {{Adding the axioms to Axiom. Toward a system of automated
          reasoning in Aldor}},
  year = "1998",
  link =
    "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.7.1457}",
  abstract =
    "This paper examines the proposal of using the type system of Axiom to
    represent a logic, and thus to use the constructions of Axiom to
    handle the logic and represent proofs and propositions, in the same
    way as is done in theorem provers based on type theory such as Nuprl
    or Coq.

    The paper shows an interesting way to decorate Axiom with pre- and
    post-conditions.",
  paper = "Poll98.pdf",
  keywords = "axiomref, CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Poll99,
  author = "Poll, Erik",
  title = {{The Type System of Axiom}},
  year = "1999",
  link = "\url{http://www.cs.ru.nl/E.Poll/talks/axiom.pdf}",
  abstract =
   "This is a slide deck from a talk on the correspondence between
    Axiom/Aldor types and Logic.",
  paper = "Poll99.pdf",
  keywords = "axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Poll99a,
  author = "Poll, Erik and Thompson, Simon",
  title = {{The Type System of Aldor}},
  link = "\url{http://www.cs.kent.ac.uk/pubs/1999/874/content.ps}",
  abstract =
    "This paper gives a formal description of -- at least a part of --
```

```
    the type system of Aldor, the extension language of the Axiom.
    In the process of doing this a critique of the design of the system
    emerges.",
  paper = "Poll99a.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@article{Poll00,
  author = "Poll, Erik and Thompson, Simon",
  title = {{Integrating Computer Algebra and Reasoning through the Type
           System of Aldor}},
  journal = "Lecture Notes in Computer Science",
  volume = "1794",
  pages = "136-150",
  year = "2000",
  abstract =
    "A number of combinations of reasoning and computer algebra systems
    have been proposed; in this paper we describe another, namely a way to
    incorporate a logic in the computer algebra system Axiom. We examine
    the type system of Aldor -- the Axiom Library Compiler -- and show
    that with some modifications we can use the dependent types of the
    system to model a logic, under the Curry-Howard isomorphism. We give
    a number of example applications of the logic we construct and explain
    a prototype implementation of a modified type-checking system written
    in Haskell.",
  paper = "Poll00.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Poll00a,
  author = "Poll, Erik and Thompson, Simon",
  title = {{Integrating Computer Algebra and Reasoning through the Type
           System of Aldor}},
  booktitle = "Frontiers of Combining Systems",
  series = "Lecture Notes in Artificial Intelligence",
  year = "2000",
  isbn = "3-540-67281-8",
  location = "Nancy, France",
  pages = "136-150",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Poll97,
  author = "Pollack, Robert",
  title = {{How to Believe a Machine-Checked Proof}},
  type = "technical report",
  institution = "Univ. of Aarhus, Basic Research in Computer Science",
  number = "BRICS RS-97-18",
  year = "1997",
  abstract =
    "Suppose I say ``Here is a machine-checked proof of Fermat's last
    theorem (FLT)''. How can you use my putative machine-checked proof
    as evidence for belief in FLT? I start from the position that you
    must have some personal experience of understanding to attain
    belief, and to have this experience you must engage your intuition
    and other mental processes which are impossible to formalise.",
  paper = "Poll97.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Prev02,
  author = "Prevosto, Virgile and Doligez, Damien",
  title = {{Algorithms and proofs inheritance in the FOC language}},
  journal = "J. Autom. Reasoning",
  volume = "29",
  number = "3-4",
  year = "2002",
  pages = "337-363",
  abstract =
    "In this paper, we present the FOC language, dedicated to the
    development of certified computer algebra libraries (that is sets of
    programs). These libraries are based on a hierarchy of implementations
    of mathematical structures. After presenting the core set of features
    of our language, we describe the static analyses, which reject
    inconsistent programs. We then show how we translate FOC definitions
    into OCAML and COQ, our target languages for the computational part
    and the proof checking, respectively.",
  paper = "Prev02.pdf",
  keywords = "axiomref, printed"
}
```

———-

1.19.15 R

— **axiom.bib** —

```
#InCollection{Rect89,
  author = "Rector, D. L.",
  title = {{Semantics in Algebraic Computation}},
  booktitle = "Computers and Mathematics",
  publisher = "Springer-Verlag",
  year = "1989",
  pages = "299-307",
  isbn = "0-387-97019-3",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Reyn02,
  author = "Reynolds, John C.",
  title = {{Separation Logic: A Logic for Shared Mutable Data Structures}},
  booktitle = "Logic in Computer Science '02",
  year = "2002",
  pages = "55-74",
  isbn = "0-7695-1483-9",
  abstract =
    "In joint work with Peter O'Hearn and others, based on early ideas of
    Burstall, we have developed an extension of Hoare logic that permits
    reasoning about low-level imperativeprograms that use shared mutable
    data structure.The simple imperative programming language is extended
    with commands (not expressions) for accessing and modifying shared
    structures, and for explicit allocation and deallocation of
    storage. Assertions are extended by introducing a ‘‘separating
    conjunction'' that asserts that its sub-formulas hold for disjoint
    parts of the heap, and a closely related ‘‘separating
    implication''. Coupled with the inductive definition of predicates on
    abstract data structures, this extension permits the concise and
    flexible description of structures with controlled sharing.In this
    paper, we will survey the current development of this program logic,
    including extensions that permit unrestricted address arithmetic,
    dynamically allocated arrays, and recursive procedures. We will also
    discuss promising future directions.",
  paper = "Reyn02.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Reyn05,
  author = "Reynolds, John C.",
  title = {{An Overview of Separation Logic}},
  year = "2005",
  journal = "LNCS",
```

```
  volume = "4171",
  pages = "460-469",
  abstract =
    "After some general remarks about program verification, we introduce
    separation logic, a novel extension of Hoare logic that can strengthen
    the applicability and scalability of program verification for
    imperative programs that use shared mutable data structures or
    shared-memory concurrency",
  paper = "Reyn05.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Robe15,
 author = "Roberts, Siobhan",
 title = {{In Mathematics, Mistakes Aren't What They Used To Be}},
 year = 2015,
 link = "\url{http://nautil.us/issue/24/error/In-mathematics-mistakes-arent-what-they-used-to-be}"
}
```

———-

— **axiom.bib** —

```
@article{Rudn01,
  author = "Rudnicki, Piotr and Schwarzweller, Christoph and
            Trybulec, Andrzej",
  title = {{Commutative algebra in the Mizar system}},
  journal = "J. Symb. Comput.",
  volume = "32",
  number = "1-2",
  pages = "143-169",
  year = "2001",
  link = "\url{https://inf.ug.edu.pl/~schwarzw/papers/jsc01.pdf}",
  abstract =
    "We report on the development of algebra in the Mizar system. This
    includes the construction of formal multivariate power series and
    polynomials as well as the definition of ideals up to a proof of the
    Hilbert basis theorem. We present how the algebraic structures are
    handled and how we inherited the past developments from the Mizar
    Mathematical Library (MML). The MML evolves and past contributions are
    revised and generalized. Our work on formal power series caused a
    number of such revisions. It seems that revising past developments
    with an intent to generalize them is a necessity when building a
    database of formalized mathematics. This poses a question: how much
    generalization is best?",
  paper = "Rudn01.pdf",
  keywords = "axiomref"
}
```

———-

1.19.16 S

— axiom.bib —

```
@techreport{Sack87,
  author = "Sacks, Elisha",
  title = {{Hierarchical Reasoning about Inequalities}},
  institution = "MIT",
  year = "1987",
  abstract =
    "This paper describes a program called BOUNDER that proves
    inequalities between functions over finite sets of constraints.
    Previous inequality algorithms perform well on some subset of the
    elementary functions, but poorly elsewhere.  To overcome this problem,
    BOUNDER maintains a hierarchy of increasingly complex algorithms.
    When one fails to resolve an inequality, it tries the next.  This
    strategy resolves more inequalities than any single algorithm.  It
    also performs well on hard problems without wasting time on easy
    ones.  The current hierarchy consists of four algorithms: bounds
    propagation, substitution, derivative inspection, and iterative
    approximation.  Propagation is an extension of interval arithmetic
    that takes linear time, but ignores constraints between variables
    and multiple occurrences of variables.  The remaining algorithms
    consider these factors, but require exponential time.  Substitution
    is a new, provably correct, algorithm for utilizing constraints
    between variables.  The final two algorithms analyze constraints
    between variables.  Inspection examines the signs of partial
    derivatives.  Iteration is based on several earlier algorithms
    from interval arithmetic.",
  paper = "Sack87.pdf"
}
```

———-

— axiom.bib —

```
@phdthesis{Schw97,
  author = "Schwarzweller, Christoph",
  title = {{MIZAR verification of generic algebraic algorithms}},
  school = "University of Tubingen",
  year = "1997",
  abstract =
    "Although generic programming founds more and more attention
    nowadays generic programming languages as well as generic libraries
    exist  there are hardly approaches for the verification of generic
    algorithms or generic libraries. This thesis deals with generic
    algorithms in the field of computer algebra. We propose the Mizar
    system as a theorem prover capable of verifying generic algorithms on
```

```
    an appropriate abstract level. The main advantage of the MIZAR theorem
    prover is its special input language that enables textbook style
    presentation of proofs. For generic versions of Brown/Henrici addition
    and of Euclideans algorithm we give complete correctness proofs
    written in the MIZAR language.

    Moreover, we do not only prove algorithms correct in the usual
    sense. In addition we show how to check, using the MIZAR system, that
    a generic algebraic algorithm is correctly instantiated with a
    particular domain. Answering this question that especially arises if
    one wants to implement generic programming languages, in the field of
    computer algebra requires nontrivial mathematical knowledge.

    To build a verification system using the MIZAR theorem prover, we also
    implemented a generator which almost automatically computes for a
    given algorithm a set of theorems that imply the correctness of this
    algorithm.",
  paper = "Schw97.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Sege91,
  author = "Seger, C. and Joyce, J.J.",
  title = {{A Two-level Formal Verification Methodology using HOL and COSMOS}},
  type = "technical report",
  year = "1991",
  number = "91-10",
  institution = "Dept. of Comp. Sci. Univ. of British Columbia",
  abstract =
    "Theorem-proving and symbolic simulation are both described as methods
    for the formal verification of hardware. They are both used to achieve
    a common goal -- correctly designed hardware -- and both are intended
    to be an alternative to conventional methods based on non-exhaustive
    simulation. However, they have different strengths and weaknesses. The
    main significance of this paper -- and its most original contribution
    -- is the suggestion that symbolic simulation and theorem-proving can
    be combined in a complementary manner.  We also outline our plans for
    the development of a mathematical interface between the two approaches
    -- in particular, a semantic link between the formulation of
    higher-order logic used in the Cambridge HOL system and the
    specification language used in the COSMOS system. We believe that this
    combination offers great potential as a practical formal verification
    methodology which combines the ability to accurately model circuit
    level behavior with the ability to reason about digital hardware at
    higher levels of abstraction"
}
```

———-

— **axiom.bib** —

```
@misc{Seli13,
  author = "Selinger, Peter",
  title = {{Lecture Notes on the Lambda Calculus}},
  year = "2013",
  link = "\url{https://www.irif.fr/~mellies/mpri/mpri-ens/biblio/Selinger-Lambda-Calculus-Notes.pdf}",
  abstract =
    "This is a set of lecture notes that developed out of courses on the
    lambda calculus that I taught at the University of Ottawa in 2001 and
    at Dalhousie University in 2007 and 2013. Topcis covered in these
    notes include the untyped lambda calculus, the Church-Rosser theorem,
    combinatory algebras, the simply-typed lambda calculus, the
    Curry-Howard isomorphism, weak and strong normalization, polymorphism,
    type inference, denotational semantics, complete partial orders, and
    the language PCF.",
  paper = "Seli13.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Simm00,
  author = "Simmons, Harold",
  title = {{Derivation and Computation: Taking the Curry-Howard correspondence
           seriously}},
  year = "2000",
  publisher = "Cambridge University Press",
  isbn = "0-521-77173-0",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@misc{Smit60,
  author = "Smith, Brian Cantwell",
  title = {{The Limits of Correctness}},
  year = "1960",
  link = "\url{http://eliza.newhaven.edu/ethics/Resources/14.Reliability-Responsibility/LimitsOfCorrectness.pdf}
  paper = "Smit60.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Soze08,
```

```
  author = "Sozeau, Mattieu and Oury, Nicolas",
  title = {{First-Class Type Classes}},
  journal = "Lecture Notes in Computer Science",
  volume = "5170",
  publisher = "Springer",
  year = "2008",
  pages = "278-293",
  link = "\url{https://www.irif.fr/~sozeau/research/publications/First-Class\_Type\_Classes.pdf}",
  abstract =
    "Type Classes have met a large success in Haskell and Isabelle, as a
    solution for sharing notations by overloading and for specifying with
    abstract structures by quantificaiton on contexts. However, both
    systems are limited by second-class implementations of these
    constructs, and these limitations are only overcome by ad-hoc
    extensions to the respective systems. We propose an embedding of type
    classes into a dependent type theory that is first-class and supports
    some of the most popular extensions right away. The implementation is
    correspondingly cheap, general, and integrates well inside the system,
    as we have experimented in Coq. We show how it can be used to help
    structured programming and proving by way of examples.",
  paper = "Soze08.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Soze12,
  author = "Sozeau, Mattieu",
  title = {{Coq with Classes}},
  booktitle = "JFLA 2012",
  link = "\url{https://www.irif.fr/~sozeau/research/publications/Coq\_with\_Classes-JFLA-040212.pdf}",
  year = "2012",
  paper = "Soze12.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Spec18,
  author = "Spector-Zabusky, Antal and Breitner, Joachim and
            Rizkallah, Christine and Weirich, Stephanie",
  title = {{Total Haskell is Reasonable Coq}},
  booktitle = "ACM SIGPLAN Int. Conf. on Certified Programs and
               Proofs",
  publisher = "ACM",
  year = "2018",
  abstract =
    "We would like to use the Coq proof assistant to mechanically
    verify properites of Haskell programs. To that end, we present a
    tool, named {\tt hs-to-coq}, that translates total Haskell
```

```
    programs into Coq programs via a shallow embedding. We apply our
    tool in three case studies -- a lawful {\tt Monad} instance,
    ‘‘Hutton’s razor’’, and an existing data structure library -- and
    prove their correctness. These examples show that this approach is
    viable: both that {\tt hs-to-coq} applies to existing Haskell
    code, and that the output it produces is amenable to
    verification.",
  paper = "Spec18.pdf",
  keywords = "printed, reviewed"
}
```

———-

— **axiom.bib** —

```
@misc{Stac17a,
  author = "cstheory.stackexchange.com",
  title = {{Why does Coq have Prop?}},
  link = "\url{http://cstheory.stackexchange.com/questions/21836/why-does-coq-have-prop/21878\#21878}",
  year = "2017",
}
```

———-

— **axiom.bib** —

```
@misc{Ster17,
  author = "Sterling, Jonathan and Harper, Robert",
  title = {{Algebraic Foundations of Proof Refinement}},
  link = "\url{http://www.cs.cmu.edu/~rwh/papers/afpr/afpr.pdf}",
  year = "2017",
  abstract =
    "We contribute a general apparatus for {\sl dependent} tactic-based
    proof refinement in the LCF tradition, in which the statements of
    subgoals may express a dependency on the proofs of other subgoals;
    this form of dependency is extremely useful and can serve as an
    {\sl algorithmic} alternative to extensions of LCF based on non-local
    instantiation of schematic variables. Additionally, we introduce a
    novel behavioral distinction between {\sl refinement rules} and
    {\sl tactics} based on naturality. Our framework, called Dependent
    LCF, is already deployed in the nascent RedPRL proof assistant for
    computational cubical type theory.",
  paper = "Ster17.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Stur95,
  author = "Sturm, T.",
```

```
  title = {{A REDUCE package for first-order logic}},
  type = "technical report",
  institution = "Universitat Passau",
  year = "1995",
}
```

———-

1.19.17 T

— **axiom.bib** —

```
@misc{Theo17,
  author = "Unknown",
  title = {{Theorema Project}},
  link = "\url{http://www.theorema.org}",
  year = "2017"
}
```

———-

— **axiom.bib** —

```
@article{Ther01,
  author = "Th\'ery, Laurent",
  title = {{A Machine-Checked Implementation of Buchberger's Algorithm}},
  journal = "Journal of Automated Reasoning",
  volume = "26",
  year = "2001",
  pages = "107-137",
  abstract = "We present an implementation of Buchberger's algorithm that
    has been proved correct within the proof assistant Coq. The
    implementation contains the basic algorithm plus two standard
    optimizations.",
  paper = "Ther01.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Ther98,
  author = "Thery, Laurent",
  title = {{A Certified Version of Buchberger's Algorithm}},
  journal = "LNCS",
  volume = "1421",
  pages = "349-364",
  year = "1998",
  abstract =
    "We present a proof of Buchberger's algorithm that has been developed
```

```
    in the Coq proof assistant. The formulation of the algorithm in Coq
    can be efficiently compiled and used to do computation",
  paper = "Ther98.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Tomu98,
  author = "Tomuta, E.",
  title = {{Proof Control in the Theorema Project}},
  year = "1998",
  school = "RISC Linz"
}
```

———-

— **axiom.bib** —

```
@article{Tray11,
  author = "Traytel, Dmitriy and Berghofer, Stefan and Nipkow, Tobias",
  title = {{Extending Hindley-Milner Type Inference with Coercive
           Structural Subtyping}},
  journal = "LNCS",
  volume = "7078",
  pages = "89-104",
  year = "2011",
  abstract =
    "We investigate how to add coercive structural subtyping to a type
    system for simply-typed lambda calculus with Hindley-Milner
    polymorphism. Coercions allow to convert between different types, and
    their automatic insertion can greatly increase readability of
    terms. We present a type inference algorithm that, given a term
    without type information, computes a type assignment and determines at
    which positions in the term coercions have to be inserted to make it
    type-correct according to the standard Hindley-Milner system (without
    any subtypes). The algorithm is sound and, if the subtype relation
    on base types is a disjoint union of lattices, also complete. The
    algorithm has been implemented in the proof assistant Isabelle.",
  paper = "Tray11.pdf",
  keywords = "coercion"
}
```

———-

— **axiom.bib** —

```
@misc{Troe97,
  author = "Troelstra, A.S.",
  title = {{From constructivism to computer science}},
```

```
  volume = "211",
  year = "1999",
  pages = "232-252",
  abstract =
    "My field is mathematical logic, with a special interest in
    constructivism, and I would not dare to call myself a computer
    scientist. But some computer scientists regard my work as a
    contribution to their field; and in this text I shall try to explain
    how this is possible, by taking a look at the history of ideas.

    I want to describe how two interrelated ideas, connected with the
    constructivistic trend in the foundations of mathematics, developed
    within mathematical logic and ultimately diffused into computer
    science.

    It will be seen that this development has not been a quite
    straightforward one. In the history of ideas it often looks as if a
    certain idea has to be discovered several times, by different people,
    before it really enters inthe the ‘‘consciousness’’ of science",
  paper = "Troe99.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Tros13,
  author = "Trostle, Anne",
  title = {{An Algorithm for the Greatest Common Divisor}},
  link = "\url{http://www.nuprl.org/MathLibrary/gcd/}",
  year = "2013"
}
```

———-

— **axiom.bib** —

```
@article{Turn95,
  author = "Turner, D.A.",
  title = {{Elemntary Strong Functional Programming}},
  journal = "Lecture Notes in Computer Science",
  volume = "1022",
  pages = "1-13",
  year = "1995",
  abstract =
    "Functional programming is a good idea, but we havent got it quite
    right yet.  What we have been doing up to now is weak (or partial)
    functional programming.  What we should be doing is strong (or
    total) functional programming - in which all computations terminate.
    We propose an elementary discipline of strong functional programming.
    A key feature of the discipline is that we introduce a type
    distinction between data, which is known to be finite, and codata,
```

```
    which is (potentially) infinite.",
  paper = "Turn95.pdf"
}
```

———-

1.19.18 W

— **axiom.bib** —

```
@inproceedings{Wadl88,
  author = "Wadler, Philip and Blott, Stephen",
  title = {{How to Make Ad-hoc Polymorphism Less Ad hoc}},
  booktitle = "Proc 16th ACM SIGPLAN-SIGACT Symp. on Princ. of Prog. Lang",
  isbn = "0-89791-294-2",
  pages = "60-76",
  year = "1988",
  link = "\url{http://202.3.77.10/users/karkare/courses/2010/cs653/Papers/ad-hoc-polymorphism.pdf}",
  abstract =
    "This paper presents {\sl type classes}, a new approach to {\sl ad-hoc}
    polymorphism. Type classes permit overloading of arithmetic operators
    such as multiplication, and generalise the ``eqtype variables'' of
    Standard ML Type classes extend the Hindley/Milner polymorphic type
    system, and provide a new approach to issues that arise in object-oriented
    programming, bounded type quantification, and abstract data types. This
    paper provides an informal introduction to type classes, and defines them
    formally by means of type inference rules",
  paper = "Wadl88.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wadl89,
  author = "Wadler, Philip",
  title = {{Theorems for free!}},
  booktitle = "4th Intl Conf. on Functional Programming",
  pages = "347-359",
  year = "1989",
  abstract =
    "From the type of a polymorphic function we can derive a theorem that
    it satisfies. Every function of the same type satisfies the same
    theorem.  This provides a free source of useful theorems, courtesy of
    Reynolds' abstraction theorem for the polymorphic lambda calculus.",
  paper = "Wadl89.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Wadl14,
  author = "Wadler, Philip",
  title = {{Propositions as Types}},
  year = "2014",
  link = "\url{http://homepages.inf.ed.ac.uk/wadler/papers/propositions-as-types/propositions-as-types.pdf}",
  paper = "Wadl14.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wald74,
  author = "Waldinger, R.J. and Levitt, K.N.",
  title = {{Reasoning about Programs}},
  journal = "Artificial Intelligence",
  volume = "5",
  number = "3",
  year = "1974",
  pages = "235-316",
  abstract =
    "This paper describes a theorem prover that embodies knowledge about
    programming constructs, such as numbers, arrays, lists, and
    expressions. The program can reason about these concepts and is used
    as part of a program verification system that uses the Floyd-Naur
    explication of program semantics. It is implemented in the qa4
    language; the qa4 system allows many pieces of strategic knowledge,
    each expressed as a small program, to be coordinated so that a program
    stands forward when it is relevant to the problem at hand. The
    language allows clear, concise representation of this sort of
    knowledge. The qa4 system also has special facilities for dealing with
    commutative functions, ordering relations, and equivalence relations;
    these features are heavily used in this deductive system. The program
    interrogates the user and asks his advice in the course of a
    proof. Verifications have been found for Hoare's FIND program, a
    real-number division algorithm, and some sort programs, as well as for
    many simpler algorithms. Additional theorems have been proved about a
    pattern matcher and a version of Robinson's unification algorithm. The
    appendix contains a complete, annotated listing of the deductive
    system and annotated traces of several of the deductions performed by
    the system.",
  paper = "Wald74.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Wals92,
  author = "Walsh, Toby and Nunes, Alex and Bundy, Alan",
```

```
  title = {{The Use of Proof Plans to Sum Series}},
  booktitle = "Proc. of 11th Int. Conf. on Automated Deduction",
  year = "1992",
  abstract =
    "We describe a program for finding closed form solutions to finite
    sums. The program was built to test the applicability of the proof
    planning search control technique in a domain of mathematics outwith
    induction. This experiment was successful. The series summing program
    extends previous work in this area and was built in a short time just
    by providing new series summing methods to our existing inductive
    theorem proving system CLAM.

    One surprising discovery was the
    usefulness of the ripple tactic in summing series. Rippling is the key
    tactic for controlling inductive proofs, and was previously thought to
    be specialised to such proofs. However, it turns out to be the key
    sub-tactic used by all the main tactics for summing series. The only
    change required was that it had to be supplemented by a difference
    matching algorithm to set up some initial meta-level annotations to
    guide the rippling process. In inductive proofs these annotations are
    provided by the application of mathematical induction. This evidence
    suggests that rippling, supplemented by difference matching, will find
    wide application in controlling mathematical proofs. The research
    reported in this paper was supported by SERC grant GR/F/71799, a SERC
    PostDoctoral Fellowship to the first author and a SERC Senior
    Fellowship to the third author. We would like to thank the other
    members of the mathematical reasoning group for their feedback on this
    project.",
  paper = "Wals92.pdf"
}
```

———-

— axiom.bib —

```
@inproceedings{Wals93,
  author = "Walsh, Toby",
  title = {{General Purpose Proof Plans}},
  booktitle = "DISCO 1993",
  year = "1993",
  pages = "319-330",
  publisher = "Springer",
  abstract =
    "One of the key problems in the design and implementation of
    automated reasoning systems is the specification of heuristics to
    control search. ‘‘Proof planning’’ has been developed as a
    powerful technique for declaratively specifying high-level proof
    strategies. This paper describes an extension to proof planning to
    allow for the specification of more general purpose plans.",
  paper = "Wals93.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Warn17,
  author = "Warner, Evan",
  title = {{Splash Talk: The Foundational Crisis of Mathematics}},
  year = "2017",
  link = "\url{web.stanford.edu/~ebwarner/SplashTalk.pdf}",
  abstract =
    "This class will cover some of the mathematics, history, and
    philosophy of the co-called {\sl foundational crisis in mathematics}.
    Broadly speaking, mathematics in the late nineteenth and early
    twentieth centuries was marked by an increased awareness of
    ``foundational issues,'' prompted by a number of problems in the
    practice of mathematics that had accumulated over the years. We will
    discuss a few examples of some of these problems, and then discusssthe
    three major schools of thought that emerged to deal with them and
    provide a coherent philosophical and methodologicial underpinning for
    mathematics.",
  paper = "Warn17.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Wied07,
  author = "Wiedijk, Freek",
  title = {{The Seventeen Provers of the World}},
  link = "\url{http://www.cs.kun.nl/~freek/comparison/comparison.pdf}",
  year = "2007",
  abstract =
    "We compare the styles of several proof assistants for mathematics.
     We present Pythagoras' proof of the irrationality of $\sqrt{2}$ both
     informal and formalized in (1) HOL, (2) Mizar, (3) PVS, (4) Coq,
     (5) Otter/Ivy, (6) Isabelle/Isar, (7) Alfa/Agda, (8) ACL2, (9) PhoX,
     (10) IMPS, (11) Metamath, (12) Theorema, (13) Lego, (14) Nuprl,
     (15) $\Omega$mega, (16) B method, (17) Minlog",
  paper = "Wied07.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Wijn68,
  author = "Wijngarrden, A. van and Mailloux, B.J. and Peck, J.E.L. and
            Koster, C.H.A. and Sintzoff, M. and Lindsey, C.H. and
            Meertens, L.G.T. and Fisker, R.G.",
  title = {{Revised Report on the Algorithmic Language ALGOL 68}},
  link = "\url{http://www.eah-jena.de/~kleine/history/languages/algol68-revisedreport.pdf}",
  year = "1968",
  paper = "Wijn68.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Wiki17,
  author = "Wikipedia",
  title = {{Calculus of constructions}},
  year = "2017",
  link = "\url{https://en.wikipedia.org/wiki/Calculus\_of\_constructions}"
}
```

———-

— **axiom.bib** —

```
@misc{Wiki19,
  author = "Wikipedia",
  title = {{Algebraic data type}},
  year = "2019",
  link = "\url{https://en.wikipedia.org/wiki/Algebraic_data_type}"
}
```

———-

— **axiom.bib** —

```
@misc{WikiED,
  author = "Wikipedia",
  title = {{Euclidean Domain}},
  year = "2017",
  link = "\url{https://en.wikipedia.org/wiki/Euclidean\_domain}"
}
```

———-

— **axiom.bib** —

```
@article{Wind14,
  author = "Windsteiger, Wolfgang",
  title = {{Theorema 2.0: A System for Mathematical Theory
           Exploration}},
  journal = "LNCS",
  volume = "8592",
  pages = "49-52",
  year = "2014",
  abstract =
    "Theorema 2.0 stands for a re-design including a complete
    re-implementation of the Theorema system, which was originally designed,
    developed, and implemented by Bruno Buchberger and his Theorema group
```

```
    at RISC. In this talk, we want to present the current status of the
    new implementation, in particular the new user interface of the
    system.",
  paper = "Wind14.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Wind06,
  author = "Windsteiger, Wolfgang",
  title = {{An automated prover for Zermelo-Fraenkel set theory in
           Theorema}},
  journal = "J. of Symbolic Computation",
  volume = "41",
  pages = "435-470",
  year = "2006",
  abstract =
    "This paper presents some fundamental aspects of the design and
    implementation of an automated prover for Zermelo-Fraenkel set theory
    within the Theorema system. The method applies the
    ‘‘Prove-Compute-Solve’’ paradigm as its major strategy for generating
    proofs in a natural style for statements involving constructs from set
    theory.",
  paper = "Wind06.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Wins93,
  author = "Winskel, Glynn",
  title = {{The Formal Semantics of Programming Languages}},
  isbn = "978-0262731034",
  year = "1993",
  publisher = "MIT",
  keywords = "shelf"
}
```

———-

1.19.19 Y

— **axiom.bib** —

```
@book{Yasu71,
  author = "Yasuhara, Ann",
```

```
  title = {{Recursive Function Theory and Logic}},
  year = "1971",
  publisher = "Academic Press",
  isbn = "0-12-768950-8",
  keywords = "shelf"
}
```

———-

1.20 Interval Arithmetic

— **axiom.bib** —

```
@misc{Boeh86,
  author = "Boehm, Hans-J. and Cartwright, Robert and Riggle, Mark and
            O'Donnell, Michael J.",
  title = {{Exact Real Arithmetic: A Case Study in Higher Order Programming}},
  year = "1986",
  link = "\url{http://dev.acm.org/pubs/citations/proceedings/lfp/319838/p162-boehm}",
  abstract =
    "Two methods for implementing {\sl exact} real arithmetic are explored
    One method is based on formulating real numbers as functions that map
    rational tolerances to rational approximations.  This approach, which
    was developed by constructive mathematicians as a concrete
    formalization of the real numbers, has lead to a surprisingly
    successful implementation.  The second method formulates real numbers
    as potentially infinite sequences of digits, evaluated on demand.
    This approach has frequently been advocated by proponents of lazy
    functional languages in the computer science community.  Ironically,
    it leads to much less satisfactory implementations.  We discuss the
    theoretical problems involved m both methods, give algortthms for the
    basic arithmetic operations, and give an empirical comparison of the
    two techniques.  We conclude wtth some general observations about the
    lazy evaluation paradigm and its implementation.",
  paper = "Boeh86.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Brig04,
  author = "Briggs, Keith",
  title = {{Exact real arithmetic}},
  link = "\url{http://keithbriggs.info/documents/xr-kent-talk-pp.pdf}",
  year = "2004",
  paper = "Bri04.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate94a,
  author = "Fateman, Richard J.; Yan, Tak W.",
  title ={{Computation with the Extended Rational Numbers and an
           Application to Interval Arithmetic}},
  link = "\url{http://www.cs.berkeley.edu/~fateman/papers/extrat.pdf}",
  abstract = "
    Programming languages such as Common Lisp, and virtually every
    computer algebra system (CAS), support exact arbitrary-precision
    integer arithmetic as well as exect rational number computation.
    Several CAS include interval arithmetic directly, but not in the
    extended form indicated here. We explain why changes to the usual
    rational number system to include infinity and ``not-a-number'' may be
    useful, especially to support robust interval computation. We describe
    techniques for implementing these changes.",
  paper = "Fate94a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Gust16,
  author = "Gustafson, John",
  title ={{A Radical Approach to Computation with Real Numbers}},
  link = "\url{http://www.johngustafson.net/presentations/Multicore2016-JLG.pdf}",
  ppt = "Gust16.pptx",
  abstract =
    "If we are willing to give up compatibility with IEEE 754 floats and
    design a number format with goals appropriate to 2016, we can achieve
    several goals simultaneously: Extremely high energy efficiency and
    information-per-bit, no penalty for decimal operations instead of
    binary, rigorous bounds on answers without the overly pessimistic
    bounds produced by interval methods, and unprecedented high speed up
    to some precision. This approach extends the ideas of unum arithmetic
    introduced two years ago by breaking completely from the IEEE
    float-type format, resulting in fixed bit size values, fixed execution
    time, no exception values or 'gradual underflow' issues, no wasted bit
    patterns, and no redundant representations (like 'negative zero'). As
    an example of the power of this format, a difficult 12-dimensional
    nonlinear robotic kinematics problem that has defied solvers to date
    is quickly solvable with absolute bounds. Also unlike interval
    methods, it becomes possible to operate on arbitrary disconnected
    subsets of the real number line with the same speed as operating on a
    simple bound.",
  paper = "Gust16.pdf",
}
```

———-

— **axiom.bib** —

```
@book{Gust16a,
  author = "Gustafson, John",
  title ={{The End of Error: Unum Computing}},
  publisher = "Chapman and Hall / CRC Computational Series",
  year = "2016",
  isbn = "978-1482239867"
}
```

———-

— **axiom.bib** —

```
@article{Gust16b,
  author = "Gustafson, John",
  title = {{Unums 2.0 An Interview with John L. Gustafson}},
  publisher = "ACM",
  journal = "Ubiquity",
  year = "2016",
  paper = "Gust16b.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Lamb06,
  author = "Lambov, Branimir",
  title = {{Interval Arithmetic Using SSE-2}},
  booktitle = "Lecture Notes in Computer Science",
  publisher = "Springer-Verlag",
  year = "2006",
  isbn = "978-3-540-85520-0",
  pages = "102-113"
}
```

———-

1.21 Numerics

— **axiom.bib** —

```
@misc{Atki09,
  author = "Atkinson, Kendall and Han, Welmin and Stewear, David",
  title = {{Numerical Solution of Ordinary Differential Equations}},
  link =
     "\url{http://homepage.math.uiowa.edu/~atkinson/papers/NAODE_Book.pdf}",
  abstract = "
    This book is an expanded version of supplementary notes that we used
```

```
    for a course on ordinary differential equations for upper-division
    undergraduate students and beginning graduate students in mathematics,
    engineering, and sciences. The book introduces the numerical analysis
    of differential equations, describing the mathematical background for
    understanding numerical methods and giving information on what to
    expect when using them. As a reason for studying numerical methods as
    a part of a more general course on differential equations, many of the
    basic ideas of the numerical analysis of differential equations are
    tied closely to theoretical behavior associated with the problem being
    solved. For example, the criteria for the stability of a numerical
    method is closely connected to the stability of the differential
    equation problem being solved.",
  paper = "Atki09.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Crank 96]{Cran96} Crank, J.; Nicolson, P.
  title = {{A practical method for numerical evaluations of solutions
            of partial differential equations of heat-conduction type}},
Advances in Computational Mathematics Vol 6 pp207-226 (1996)
  link = "\url{http://www.acms.arizona.edu/FemtoTheory/MK_personal/opti547/literature/CNMethod-original.pdf}",
  paper = "Cran96.pdf",
```

———-

— **ignore** —

```
\bibitem[Lef\'evre 06]{Lef06} Lef\'evre, Vincent; Stehl\'e, Damien;
Zimmermann, Paul
  title = {{Worst Cases for the Exponential Function in the IEEE-754r
           decimal64 Format}},
in Lecture Notes in Computer Science, Springer ISBN 978-3-540-85520-0
(2006) pp114-125
  abstract = "
    We searched for the worst cases for correct rounding of the
    exponential function in the IEEE 754r decimal64 format, and computed
    all the bad cases whose distance from a breakpoint (for all rounding
    modes) is less than $10^{-15}$ ulp, and we give the worst ones. In
    particular, the worst case for
    $\vert{}x\vert{} \ge 3 x 10^{-11}$ is
    \[
    exp(9.407822313572878x10^{-2} =
    1.0986456820663385000000000000000278\ldots
    \]
    This work can be extended to other elementary functions in the decimal64
    format and allows the design of reasonably fast routines that will
    evaluate these functions with correct rounding, at least in some
    situations."
```

— **axiom.bib** —

```
@book{Hamm62,
  author = "Hamming, R W.",
  title = {{Numerical Methods for Scientists and Engineers}},
  publisher = "Dover",
  year = "1973",
  isbn = "0-486-65241-6"
}
```

— **axiom.bib** —

```
@article{Nord62,
  author = "Nordsieck, Arnold",
  title = {{On Numerical Integration of Ordinary Differential Equations}},
  journal = "Mathematics of Computations",
  volume = "XVI",
  year = "1962",
  pages = "22-49",
  abstract =
    "A reliable efficient general-purpose method for automatic digital
    computer integration of systems of ordinary differential equations is
    described. The method operates with the current values of the higher
    derivatives of a polynomial approximating the solution. It is
    thoroughly stable under all circumstances, incorporates automatic
    starting and automatic choice and revision of elementary interval
    size, approximately minimizes the amount of computation for a
    specified accuracy of solution, and applies to any system of
    differential equations with derivatives continuous or piecewise
    continuous with finite jumps. ILLIAC library subroutine F7, University
    of Illinois Digital Computer Laboratory, is a digital computer program
    applying this method."
}
```

— **axiom.bib** —

```
@misc{Walt71,
  author = "Walther, J.S.",
  title = {{A Unified Algorithm for Elementary Functions}},
  link = "\url{}",
  year = "1971",
  abstract =
    "This paper describes a single unified algorithm for the calculation
    of elementary functions including multipli- cation, division, sin,
    cos, tan, arctan, sinh, cosh, tanh, arctanh, In, exp and square-root.
    The basis for the algorithm is coordinate rotation in a linear,
```

```
    circular, or hyperbolic coordinate system depending on which function
    is to be calculated.  The only operations re- quired are shifting,
    adding, subtracting and the recall of prestored constants.  The
    limited domain of con- vergence of the algorithm is calculated,
    leading to a discussion of the modifications required to extend the
    domain for floating point calculations.

    A hardware floating point processor using the algo- rithm was built at
    Hewlett-Packard Laboratories.  The block diagram of the processor, the
    microprogram control used for the algorithm, and measures of actual
    performance are shown.",
  paper = "Walt71.pdf"
}
```

———-

1.22 Advanced Documentation

— **ignore** —

```
\bibitem [Bostock 14]{Bos14} Bostock, Mike
  title = {{Visualizing Algorithms}},
  link = "\url{http://bost.ocks.org/mike/algorithms}",
  abstract = "
    This website hosts various ways of visualizing algorithms. The hope is
    that these kind of techniques can be applied to Axiom."
```

———-

— **axiom.bib** —

```
@misc{Kama15,
  author = "Kamareddine, Fairouz and Wells, Joe and Zengler, Christoph and
            Barendregt, Henk",
  title = {{Computerising Mathematical Text}},
  year = "2015",
  abstract =
    "Mathematical texts can be computerised in many ways that capture
    differing amounts of the mathematical meaning. At one end, there is
    document imaging, which captures the arrangement of black marks on
    paper, while at the other end there are proof assistants (e.g. Mizar,
    Isabelle, Coq, etc.), which capture the full mathematical meaning and
    have proofs expressed in a formal foundation of mathematics. In
    between, there are computer typesetting systems (e.g. Latex and
    Presentation MathML) and semantically oriented systems (e.g. Content
    MathML, OpenMath, OMDoc, etc.). In this paper we advocate a style of
    computerisation of mathematical texts which is flexible enough to
    connect the diferent approaches to computerisation, which allows
    various degrees of formalsation, and which is compatible with
    different logical frameworks (e.g. set theory, category theory, type
```

```
    theory, etc.) and proof systems. The basic idea is to allow a
    man-machine collaboration which weaves human input with machine
    computation at every step in the way. We propose that the huge step from
    informal mathematics to fully formalised mathematics be divided into
    smaller steps, each of which is a fully developed method in which
    human input is minimal."
}
```

———-

— **axiom.bib** —

```
@misc{Leeu94a,
  author = {van Leeuwen, Andr\'e M.A.},
  title = {{Representation of mathematical object in interactive books}},
  abstract = "
    We present a model for the representation of mathematical objects in
    structured electronic documents, in a way that allows for interaction
    with applications such as computer algebra systems and proof checkers.
    Using a representation that reflects only the intrinsic information of
    an object, and storing application-dependent information in so-called
    {\sl application descriptions}, it is shown how the translation from
    the internal to an external representation and {\sl vice versa} can be
    achieved. Hereby a formalisation of the concept of {\sl context} is
    introduced. The proposed scheme allows for a high degree of
    application integration, e.g., parallel evaluation of subexpressions
    (by different computer algebra systems), or a proof checker using a
    computer algebra system to verify an equation involving a symbolic
    computation.",
  paper = "Leeu94a.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Soiffer 91]{Soif91} Soiffer, Neil Morrell
  title = {{The Design of a User Interface for Computer Algebra Systems}},
  link = "\url{http://www.eecs.berkeley.edu/Pubs/TechRpts/1991/CSD-91-626.pdf}",
  abstract = "
    This thesis discusses the design and implementation of natural user
    interfaces for Computer Algebra Systems. Such an interface must not
    only display expressions generated by the Computer Algebra System in
    standard mathematical notation, but must also allow easy manipulation
    and entry of expressions in that notation. The user interface should
    also assist in understanding of large expressions that are generated
    by Computer Algebra Systems and should be able to accommodate new
    notational forms.",
  paper = "Soif91.pdf"
```

———-

— **ignore** —

```
\bibitem[Victor 11]{Vict11} Victor, Bret
  title = {{Up and Down the Ladder of Abstraction}},
  link = "\url{http://worrydream.com/LadderOfAbstraction}",
  abstract = "
    This interactive essay presents the ladder of abstraction, a technique for
    thinking explicitly about these levels, so a designer can move among
    them consciously and confidently. "
```

———-

— **ignore** —

```
\bibitem[Victor 12]{Vict12} Victor, Bret
  title = {{Inventing on Principle}},
  link = "\url{http://www.youtube.com/watch?v=PUv66718DII}",
  abstract = "
    This video raises the level of discussion about human-computer
    interaction from a technical question to a question of effectively
    capturing ideas.  In particular, this applies well to Axiom's focus on
    literate programming."
```

———-

1.23 Differential Equations

— **ignore** —

```
\bibitem[Abramov 95]{Abra95} Abramov, Sergei A.; Bronstein, Manuel;
Petkovsek, Marko
  title = {{On Polynomial Solutions of Linear Operator Equations}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  paper = "Abra95.pdf"
```

———-

— **axiom.bib** —

```
@misc{Abra01,
  author = "Abramov, Sergei and Bronstein, Manuel",
  title = {{On Solutions of Linear Functional Systems}},
  year = "2001",
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  algebra =
  "\newline\refto{category OREPCAT UnivariateSkewPolynomialCategory}
   \newline\refto{category LODOCAT LinearOrdinaryDifferentialOperatorCategory}
   \newline\refto{domain AUTOMOR Automorphism}
   \newline\refto{domain ORESUP SparseUnivariateSkewPolynomial}
```

```
   \newline\refto{domain OREUP UnivariateSkewPolynomial}
   \newline\refto{domain LODO LinearOrdinaryDifferentialOperator}
   \newline\refto{domain LODO1 LinearOrdinaryDifferentialOperator1}
   \newline\refto{domain LODO2 LinearOrdinaryDifferentialOperator2}
   \newline\refto{package APPLYORE ApplyUnivariateSkewPolynomial}
   \newline\refto{package OREPCTO UnivariateSkewPolynomialCategoryOps}
   \newline\refto{package LODOF LinearOrdinaryDifferentialOperatorFactorizer}
   \newline\refto{package LODOOPS LinearOrdinaryDifferentialOperatorsOps}",
  abstract = "
    We describe a new direct algorithm for transforming a linear system of
    recurrences into an equivalent one with nonsingular leading or
    trailing matrix. Our algorithm, which is an improvement to the EG
    elimination method, uses only elementary linear algebra operations
    (ranks, kernels, and determinants) to produce an equation satisfied by
    the degress of the solutions with finite support. As a consequence, we
    can boudn and compute the polynomial and rational solutions of very
    general linear functional systems such as systems of differential or
    ($q$)-difference equations.",
  paper = "Abra01.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bron96b,
  author = "Bronstein, Manuel",
  title = {{On the Factorization of Linear Ordinary Differential Operators}},
  booktitle = "Mathematics and Computers in Simulation",
  volume = "42",
  pages = "387-389",
  year = "1996",
  abstract =
    "After reviewing the arithmetic of linear ordinary differential
    operators, we describe the current status of the factorisation
    algorithm, specially with respect to factoring over non-algebraically
    closed constant fields. We also describe recent results from Singer
    and Ulmer that reduce determining the differential Galois group of an
    operator to factoring.",
  paper = "Bro96b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bron96a,
  author = "Bronstein, Manuel and Petkovsek, Marko",
  title = {{An introduction to pseudo-linear algebra}},
  journal = "Theoretical Computer Science",
  volume = "157",
  pages = "3-33",
```

```
  year = "1966",
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  algebra = "\newline\refto{category LORER LeftOreRing}",
  abstract =
    "Pseudo-linear algebra is the study of common properties of linear
    differential and difference operators. We introduce in this paper its
    basic objects (pseudo-derivations, skew polynomials, and pseudo-linear
    operators) and describe several recent algorithms on them, which, when
    applied in the differential and difference cases, yield algorithms for
    uncoupling and solving systems of linear differential and difference
    equations in closed form.",
  paper = "Bron96a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bron01,
  author = "Bronstein, Manuel",
  title = {{Computer Algebra Algorithms for Linear Ordinary Differential
           and Difference equations}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/ecm3.pdf}",
  booktitle = "European Congress of Mathematics",
  series = "Progress in Mathematics",
  volume = "202",
  year = "2001",
  pages = "105-119",
  abstract = "
    Galois theory has now produced algorithms for solving linear ordinary
    differential and difference equations in closed form. In addition,
    recent algorithmic advances have made those algorithms effective and
    implementable in computer algebra systems. After introducing the
    relevant parts of the theory, we describe the latest algorithms for
    solving such equations.",
  paper = "Bron01.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Bronstein 94]{Bro94} Bronstein, Manuel
  title = {{An improved algorithm for factoring linear ordinary
           differential operators}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  abstract = "
    We describe an efficient algorithm for computing the associated
    equations appearing in the Beke-Schlesinger factorisation method for
    linear ordinary differential operators. This algorithm, which is based
    on elementary operations with sets of integers, can be easily
    implemented for operators of any order, produces several possible
```

```
    associated equations, of which only the simplest can be selected for
    solving, and often avoids the degenerate case, where the order of the
    associated equation is less than in the generic case. We conclude with
    some fast heuristics that can produce some factorizations while using
    only linear computations."
```

———-

— **ignore** —

```
\bibitem[Bronstein 90]{Bro90} Bronstein, Manuel
  title = {{On Solutions of Linear Ordinary Differential Equations in
            their Coefficient Field}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  abstract = "
    We describe a rational algorithm for finding the denominator of any
    solution of a linear ordinary differential equation in its coefficient
    field. As a consequence, there is now a rational algorithm for finding
    all such solutions when the coefficients can be built up from the
    rational functions by finitely many algebraic and primitive
    adjunctions. This also eliminates one of the computational bottlenecks
    in algorithms that either factor or search for Liouvillian solutions
    of such equations with Liouvillian coefficients.",
  paper = "Bro90.pdf"
```

———-

— **axiom.bib** —

```
@misc{Bron96,
  author = "Bronstein, Manuel",
  title =
   {{$\sum^{IT}$ -- A strongly-typed embeddable computer algebra library}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  abstract =
    "We describe the new computer algebra library $\sum^{IT}$ and its
    underlying design. The development of $\sum^{IT}$ is motivated by the
    need to provide highly efficient implementations of key algorithms for
    linear ordinary differential and ($q$)-difference equations to
    scientific programmers and to computer algebra users, regardless of
    the programming language or interactive system they use. As such,
    $\sum^{IT}$ is not a computer algebra system per se, but a library (or
    substrate) which is designed to be ``plugged'' with minimal efforts
    into different types of client applications.",
  paper = "Bron96.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Bronstein 99a]{Bro99a} Bronstein, Manuel
  title = {{Solving linear ordinary differential equations over
            $C(x,e^{\int{f(x)dx}})$}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  abstract = "
    We describe a new algorithm for computing the solutions in
    \[F=C(x,e^{\int{f(x)dx}})\] of linear ordinary differential equations
    with coefficients in $F$. Compared to the general algorithm, our
    algorithm avoids the computation of exponential solutions of equations
    with coefficients in $C(x)$, as well as the solving of linear
    differential systems over $C(x)$. Our method is effective and has been
    implemented.",
  paper = "Bro99a.pdf"
```

———-

— **ignore** —

```
\bibitem[Bronstein 00]{Bro00} Bronstein, Manuel
  title = {{On Solutions of Linear Ordinary Differential Equations in
            their Coefficient Field}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  abstract = "
    We extend the notion of monomial extensions of differential fields,
    i.e. simple transcendental extensions in which the polynomials are
    closed under differentiation, to difference fields. The structure of
    such extensions provides an algebraic framework for solving
    generalized linear difference equations with coefficients in such
    fields. We then describe algorithms for finding the denominator of any
    solution of those equations in an important subclass of monomial
    extensions that includes transcendental indefinite sums and
    products. This reduces the general problem of finding the solutions of
    such equations in their coefficient fields to bounding their
    degrees. In the base case, this yields in particular a new algorithm
    for computing the rational solutions of $q$-difference equations with
    polynomial coefficients.",
  paper = "Bro00.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bron02,
  author = "Bronstein, Manuel and Lafaille, S\'ebastien",
  title = {{Solutions of linear ordinary differential equations in terms
           of special functions}},
  booktitle = "Proc. ISSAC '02",
  publisher = "ACM Press",
  pages = "23-28",
  year = "2002",
  isbn = "1-58113-484-3",
```

```
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/issac2002.pdf}",
  url2 = "http://xena.hunter.cuny.edu/ksda/papers/bronstein2.pdf",
  paper2 = "Bron02x.pdf",
  abstract =
    "We describe a new algorithm for computing special function solutions
    of the form $y(x) = m(x)F(\eta(x))$ of second order linear ordinary
    differential equations, where $m(x)$ is an arbitrary Liouvillian
    function, $\eta(x)$ is an arbitrary rational function, and $F$
    satisfies a given second order linear ordinary differential
    equations. Our algorithm, which is base on finding an appropriate
    point transformation between the equation defining $F$ and the one to
    solve, is able to find all rational transformations for a large class
    of functions $F$, in particular (but not only) the $_0F_1$ and $_1F_1$
    special functions of mathematical physics, such as Airy, Bessel,
    Kummer and Whittaker functions. It is also able to identify the values
    of the parameters entering those special functions, and can be
    generalized to equations of higher order.",
  paper = "Bron02.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Bronstein 03]{Bro03} Bronstein, Manuel; Trager, Barry M.
  title = {{A Reduction for Regular Differential Systems}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mega2003.pdf}",
  abstract = "
    We propose a definition of regularity of a linear differential system
    with coefficients in a monomial extension of a differential field, as
    well as a global and truly rational (i.e. factorisation-free)
    iteration that transforms a system with regular finite singularites
    into an equivalent one with simple finite poles. We then apply our
    iteration to systems satisfied by bases of algebraic function fields,
    obtaining algorithms for computing the number of irreducible
    components and the genus of algebraic curves.",
  paper = "Bro03.pdf"
```

———-

— **ignore** —

```
\bibitem[Bronstein 03a]{Bro03a} Bronstein, Manuel; Sol\'e, Patrick
  title = {{Linear recurrences with polynomial coefficients}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  abstract = "
    We relate sequences generated by recurrences with polynomial
    coefficients to interleaving and multiplexing of sequences generated
    by recurrences with constant coefficients. In the special case of
    finite fields, we show that such sequences are periodic and provide
    linear complexity estimates for all three constructions.",
  paper = "Bro03a.pdf"
```

———-

— **axiom.bib** —

```
@misc{Bron02a,
  author = "Bronstein, Manuel",
  title = {{$\Sigma^{it}$ User Guide and Reference Manual}},
  year = "2002",
  paper = "Bron02a.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Bronstein 05]{Bro05} Bronstein, Manuel; Li, Ziming; Wu, Min
  title = {{Picard-Vessiot Extensions for Linear Functional Systems}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/issac2005.pdf}",
  abstract = "
    Picard-Vessiot extensions for ordinary differential and difference
    equations are well known and are at the core of the associated Galois
    theories. In this paper, we construct fundamental matrices and
    Picard-Vessiot extensions for systems of linear partial functional
    equations having finite linear dimension. We then use those extensions
    to show that all the solutions of a factor of such a system can be
    completed to solutions of the original system.",
  paper = "Bro05.pdf"
```

———-

— **axiom.bib** —

```
@article{Dave86,
  author = "Davenport, James H.",
  title = {{The Risch Differential Equation Problem}},
  year = "1986",
  journal = "SIAM J. COMPUT.",
  volume = "15",
  number = "4",
  comment = "Technical Report 83-4, Dept. Comp. Sci, Univ. Delaware",
  abstract = "
    We propose a new algorithm, similar to Hermite's method for the
    integration of rational functions, for the resolution of Risch
    differential equations in closed form, or proving that they have no
    resolution. By requiring more of the presentation of our differential
    fields (in particular that the exponentials be weakly normalized), we
    can avoid the introduction of arbitrary constants which have to be
    solved for later.

    We also define a class of fields known as exponentially reduced, and
```

```
    show that solutions of Risch differential equations which arise from
    integrating in these fields satisfy the ‘‘natural’’ degree constraints
    in their main variables, and we conjecture (after Risch and Norman)
    that this is true in all variables.",
  paper = "Dave86.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Dave86a,
  author = "Davenport, James H.",
  title = {{SCRATCHPAD II Programming Language Reference}},
  link = "\url{https://hopl.info/Showsource.prx?source=15034}",
  comment = "IBM T.J. Watson",
  year = "1986"
}
```

———-

— **ignore** —

```
\bibitem[Singer 9]{Sing91.pdf} Singer, Michael F.
  title = {{Liouvillian Solutions of Linear Differential Equations with
           Liouvillian Coefficients}},
J. Symbolic Computation V11 No 3 pp251-273
  year = "1991",
  link = "\url{http://www.sciencedirect.com/science/article/pii/S074771710880048X}",
  abstract = "
    Let $L(y)=b$ be a linear differential equation with coefficients in a
    differential field $K$. We discuss the problem of deciding if such an
    equation has a non-zero solution in $K$ and give a decision procedure
    in case $K$ is an elementary extension of the field of rational
    functions or is an algebraic extension of a transcendental liouvillian
    extension of the field of rational functions We show how one can use
    this result to give a procedure to find a basis for the space of
    solutions, liouvillian over $K$, of $L(y)=0$ where $K$ is such a field
    and $L(y)$ has coefficients in $K$.",
  paper = "Sing91.pdf"
```

———-

— **ignore** —

```
\bibitem[Von Mohrenschildt 94]{Mohr94} {von Mohrenschildt}, Martin
  title = {{Symbolic Solutions of Discontinuous Differential Equations}},
  link = "\url{http://e-collection.library.ethz.ch/eserv/eth:39463/eth-39463-01.pdf}",
  paper = "Mohr94.pdf"
```

———-

— **ignore** —

```
\bibitem[Von Mohrenschildt 98]{Mohr98} von Mohrenschildt, Martin
  title = {{A Normal Form for Function Rings of Piecewise Functions}},
J. Symbolic Computation (1998) Vol 26 pp607-619
  link = "\url{http://www.cas.mcmaster.ca/~mohrens/JSC.pdf}",
  abstract = "
    Computer algebra systems often have to deal with piecewise continuous
    functions. These are, for example, the absolute value function,
    signum, piecewise defined functions but also functions that are the
    supremum or infimum of two functions. We present a new algebraic
    approach to these types of problems. This paper presents a normal form
    for a function ring containing piecewise polynomial functions of an
    expression. The main result is that this normal form can be used to
    decide extensional equality of two piecewise functions. Also we define
    supremum and infimum for piecewise functions; in fact, we show that
    the function ring forms a lattice. Additionally, a method to solve
    equalities and inequalities in this function ring is
    presented. Finally, we give a ``user interface'' to the algebraic
    representation of the piecewise functions.",
  paper = "Mohr98.pdf"
```

———-

— **ignore** —

```
\bibitem[Weber 06]{Webe06} Weber, Andreas
  title = {{Quantifier Elimination on Real Closed Fields and Differential
           Equations}},
  link = "\url{http://cg.cs.uni-bonn.de/personal-pages/weber/publications/pdf/WeberA/Weber2006a.pdf}",
  keywords = "survey",
  abstract = "
    This paper surveys some recent applications of quantifier elimination
    on real closed fields in the context of differential
    equations. Although polynomial vector fields give rise to solutions
    involving the exponential and other transcendental functions in
    general, many questions can be settled within the real closed field
    without referring to the real exponential field. The technique of
    quantifier elimination on real closed fields is not only of
    theoretical interest, but due to recent advances on the algorithmic
    side including algorithms for the simplification of quantifier-free
    formulae the method has gained practical applications, e.g. in the
    context of computing threshold conditions in epidemic modeling.",
  paper = "Webe06.pdf"
```

———-

— **ignore** —

```
\bibitem[Ulmer 03]{Ulm03} Ulmer, Felix
  title = {{Liouvillian solutions of third order differential equations}},
J. Symbolic COmputations 36 pp 855-889
```

```
  year = "2003",
  link = "\url{http://www.sciencedirect.com/science/article/pii/S0747717103000658}",
  abstract = "
    The Kovacic algorithm and its improvements give explicit formulae for
    the Liouvillian solutions of second order linear differential
    equations. Algorithms for third order differential equations also
    exist, but the tools they use are more sophisticated and the
    computations more involved. In this paper we refine parts of the
    algorithm to find Liouvillian solutions of third order equations. We
    show that,except for four finite groups and a reduction to the second
    order case, it is possible to give a formula in the imprimitve
    case. We also give necessary conditions and several simplifications
    for the computation of the minimal polynomial for the remaining finite
    set of finite groups (or any known finite group) by extracting
    ramification information from the character table. Several examples
    have been constructed, illustrating the possibilities and limitations.",
  paper = "Ulm03.pdf"
```

———-

1.24 Expression Simplification

— **axiom.bib** —

```
@article{Bund94,
  author = "Bundgen, Reinhard",
  title = {{Combining Computer Algebra and Rule Based Reasoning}},
  journal = "LNCS",
  volume = "958",
  pages = "209-223",
  year = "1994",
  abstract =
    "We present extended term rewriting systems as a means to describe a
    simplification relation for an equational specification with a
    built-ln domain of external objects.  Even if the extended term
    rewriting system is canonical, the combined relation including
    built-in computations of 'ground terms' needs neither be terminating
    nor confluent.  We investigate restrictions on the extended term
    rewriting systems and the built-in domains under which these
    properties hold.  A very important property of extended term rewriting
    systems is decomposition freedom.  Among others decomposition free
    extended term rewriting systems allow for efficient simplifications.
    Some interesting algebraic applications of canonical simplification
    relations are presented.",
  paper = "Bund94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Fort87,
  author = "Fortenbacher, Albrecht",
  title = {{An Algebraic Approach to Unification Under Associativity and
           Commutativity}},
  journal = "J. Symbolic Computation",
  volume = "3",
  pages = "217-229",
  year = "1987",
  abstract =
    "From the work of Siekmann and Livesey, and Stickel it is known how to
    unify two terms in an associative and commutative theory: transfer the
    terms into Abelian strings, look for mappings which solve the problem
    in the Abelian monoid, and decide whether a mapping can be regarded as
    a unifier.  Very often most of the mappings are thus eliminated, and
    so it is crucial for efficiency either to not create these unnecessary
    solutions or to remove them as soon as possible.  The following work
    formalises the transformations between the free algebra and this
    monoid.  This leads to an algorithm which uses maximal information for
    its search for solutions in the monoid.  It is both very efficient and
    easily verifiable.  Some applications of this algorithm are shown in
    the appendix.",
  paper = "Fort87.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Carette 04]{Car04} Carette, Jacques
  title = {{Understanding Expression Simplification}},
  link = "\url{http://www.cas.mcmaster.ca/~carette/publications/simplification.pdf}",
  abstract = "
    We give the first formal definition of the concept of {\sl
    simplification} for general expressions in the context of Computer
    Algebra Systems. The main mathematical tool is an adaptation of the
    theory of Minimum Description Length, which is closely related to
    various theories of complexity, such as Kolmogorov Complexity and
    Algorithmic Information Theory. In particular, we show how this theory
    can justify the use of various ``magic constants'' for deciding
    between some equivalent representations of an expression, as found in
    implementations of simplification routines.",
  paper = "Car04.pdf"
```

———-

— **axiom.bib** —

```
@inproceedings{Cavi76,
  author = "Caviness, Bob F. and Fateman, Richard J.",
  title = {{Simplification of Radical Expressions}},
  booktitle = "Proc. 1976 SYMSAC",
```

```
  pages = "329-338",
  year = "1976",
  abstract =
    "In this paper we discuss the problem of simplifying unnested radical
    expressions. We describe an algorithm implemented in MACSYMA that
    simplifies radical expressions and then follow this description with
    a formal treatment of the problem. Theoretical computing times for some
    of the algorithms are briefly discussed as is related work of other
    authors",
  paper = "Cavi76.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Land93,
  author = "Landau, Susan",
  title = {{How to Tangle with a Nested Radical}},
  institution = "University of Massachusetts",
  journal = "The Mathematical Intelligencer",
  year = "1993",
  paper = "Land93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Shac90,
  author = "Shackell, John",
  title = {{Growth Estimates for Exp-Log Functions}},
  journal = "J. Symbolic Computation",
  volume = "10",
  year = "1990",
  pages = "611-632",
  abstract =
    "Exp-log functions are those obtained from the constant 1 and the
    variable X by means of arithmetic operations and the function symbols
    exp() and log(). This paper gives an explicit algorithm for
    determining eventual dominance of these functions modulo an oracle for
    deciding zero equivalence of constant terms. This also provides
    another proof that the dominance problem for exp-log functions is
    Turing-reducible to the identity problem for constant terms."
}
```

———-

— **axiom.bib** —

```
@article{Stou76,
  author = "Stoutemyer, David R.",
  title = {{Automatic Simplification for the Absolute-value Function and its
           Relatives}},
  journal = "ACM SIGSAM",
  volume = "10",
  number = "4",
  year = "1976",
  pages = "48-49",
  abstract =
    "Computer symbolic mathematics has made impressive progress for the
    automatic simplification of rational expressions, algebraic
    expressions, and elementary transcendental expressions. However,
    existing computer-algebra systems tend to provide little or no
    simplification for the absolute-value function or for its relatives
    such as the signum, unit ramp, unit step, max, min, modulo, and Dirac
    delta functions. Although these functions lack certain desireable
    properties that are helpful for canonical simplification, there are
    opportunities for some ad hoc simplification. Moreover, a perusal of
    most mathematics, engineering, and scientific journals or texts
    reveals that these functions are too prevalent to be ignored.This
    article describes specific simplification rules implemented in a
    program that supplements the built-in rules for the MACSYMA ABS and
    SIGNUM functions.",
  paper = "Stou76.pdf"
}
```

———-

1.25 Integration

— **ignore** —

```
\bibitem[Adamchik xx]{Adamxx} Adamchik, Victor
  title = {{Definite Integration}},
  link = "\url{http://www.cs.cmu.edu/~adamchik/articles/integr/mj.pdf}",
  paper = "Adamxx.pdf"
```

———-

— **ignore** —

```
\bibitem[Adamchik 97]{Adam97} Adamchik, Victor
  title = {{A Class of Logarithmic Integrals}},
  link = "\url{http://www.cs.cmu.edu/~adamchik/articles/issac/issac97.pdf}",
  abstract = "
    A class of definite integrals involving cyclotomic polynomials and
    nested logarithms is considered. The results are given in terms of
    derivatives of the Hurwitz Zeta function. Some special cases for which
    such derivatives can be expressed in closed form are also considered.",
```

```
  paper = "Adam97.pdf"
```

———-

— ignore —

```
\bibitem[Avgoustis 77]{Avgo77} Avgoustis, Ioannis Dimitrios
  title =
    {{Definite Integration using the Generalized Hypergeometric Functions}},
  link = "\url{http://dspace.mit.edu/handle/1721.1/16269}",
  abstract = "
    A design for the definite integration of approximately fifty Special
    Functions is described. The Generalized Hypergeometric Functions are
    utilized as a basis for the representation of the members of the above
    set of Special Functions. Only a relatively small number of formulas
    that generally involve Generalized Hypergeometric Functions are
    utilized for the integration stage.  A last and crucial stage is
    required for the integration process: the reduction of the Generalized
    Hypergeometric Function to Elementary and/or Special Functions.

    The result of an early implementation which involves Laplace
    transforms are given and some actual examples with their corresponding
    timing are provided.",
  paper = "Avgo77.pdf"
```

———-

— ignore —

```
\bibitem[Baddoura 89]{Bad89} Baddoura, Jamil
  title = {{A Dilogarithmic Extension of Liouville's Theorem on Integration in
           Finite Terms}},
  link = "\url{http://www.dtic.mil/dtic/tr/fulltext/u2/a206681.pdf}",
  abstract = "
    The result obtained generalizes Liouville's Theorem by allowing, in
    addition to the elementary functions, dilogarithms to appear in the
    integral of an elementary function. The basic conclusion is that an
    associated function to the dilogarihm, if dilogarithms appear in the
    integral, appears linearly, with logarithms appearing in a non-linear
    way.",
  paper = "Bad89.pdf"
```

———-

— ignore —

```
\bibitem[Baddoura 94]{Bad94} Baddoura, Mohamed Jamil
  title = {{Integration in Finite Terms with Elementary Functions and
           Dilogarithms}},
  link = "\url{http://dspace.mit.edu/bitstream/handle/1721.1/26864/30757785.pdf}",
  abstract = "
```

```
    In this thesis, we report on a new theorem that generalizes
    Liouville's theorem on integration in finite terms. The new theorem
    allows dilogarithms to occur in the integral in addition to elementary
    functions. The proof is base on two identities for the dilogarithm,
    that characterize all the possible algebraic relations among
    dilogarithms of functions that are built up from the rational
    functions by taking transcendental exponentials, dilogarithms, and
    logarithms.",
  paper = "Bad94.pdf"
```

———-

— **ignore** —

```
\bibitem[Baddoura 10]{Bad10} Baddoura, Jamil
  title = {{A Note on Symbolic Integration with Polylogarithms}},
  year = "2011",
J. Math Vol 8 pp229-241 (2011)
  abstract = "
    We generalize partially Liouville's theorem on integration in finite
    terms to allow polylogarithms of any order to occur in the integral in
    addition to elementary functions. The result is a partial
    generalization of a theorem proved by the author for the
    dilogarithm. It is also a partial proof of a conjecture postulated by
    the author in 1994. The basic conclusion is that an associated
    function to the nth polylogarithm appears linearly with logarithms
    appearing possibly in a polynomial way with non-constant coefficients.",
  paper = "Bad10.pdf"
```

———-

— **ignore** —

```
\bibitem[Bajpai 70]{Bajp70} Bajpai, S.D.
  title = {{A contour integral involving legendre polynomial and
           Meijer's G-function}},
  link = "\url{http://link.springer.com/article/10.1007/BF03049565}",
  abstract = "
    In this paper a countour integral involving Legendre polynomial and
    Meijer's G-function is evaluated. the integral is of general character
    and it is a generalization of results recently given by Meijer,
    MacRobert and others.  An integral involving regular radial Coulomb
    wave function is also obtained as a particular case.",
  paper = "Bajp70.pdf"
```

———-

— **axiom.bib** —

```
@misc{Bronxxa,
  author = "Bronstein, Manuel",
```

```
  title = {{Symbolic Integration: towards Practical Algorithms}},
  abstract =
    "After reviewing the Risch algorithm for the integration of elementary
    functions and the underlying theory, we descrbe the successive
    improvements in the field, and the current ‘‘rational’’ approach to
    symbolic integration. We describe how a technique discovered by
    Hermite a century ago can be efficiently applied to rational,
    algebraic, elementary transcendental and mixed elementary functions."
}
```

———-

— **axiom.bib** —

```
@article{Bron88,
  author = "Bronstein, Manuel",
  title = {{The Transcendental Risch Differential Equation}},
  journal = "J. Symbolic Computation",
  volume = "9",
  year = "1988",
  pages = "49-60",
  abstract =
    "We present a new rational algorithm for solving Risch differential
    equations in towers of transcendental elementary extensions.  In
    contrast to a recent algorithm of Davenport we do not require a
    progressive reduction of the denominators involved, but use weak
    normality to obtain a formula for the denominator of a possible
    solution.  Implementation timings show this approach to be faster than
    a Hermite-like reduction.",
  paper = "Bron88.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Bronstein 89]{Bro89a} Bronstein, M.
  title = {{An Algorithm for the Integration of Elementary Functions}},
Lecture Notes in Computer Science Vol 378 pp491-497
  year = "1989",
  abstract = "
    Trager (1984) recently gave a new algorithm for the indefinite
    integration of algebraic functions. His approach was ‘‘rational’’ in
    the sense that the only algebraic extension computed in the smallest
    one necessary to express the answer. We outline a generalization of
    this approach that allows us to integrate mixed elementary
    functions. Using only rational techniques, we are able to normalize
    the integrand, and to check a necessary condition for elementary
    integrability.",
  paper = "Bro89a.pdf"
```

———-

— **axiom.bib** —

```
@article{Bron90a,
  author = "Bronstein, Manuel",
  title = {{Integration of Elementary Functions}},
  journal = "J. Symbolic Computation",
  volume = "9",
  pages = "117-173",
  year = "1990",
  abstract =
    "We extend a recent algorithm of Trager to a decision procedure for the
    indefinite integration of elementary functions. We can express the
    integral as an elementary function or prove that it is not
    elementary. We show that if the problem of integration in finite terms
    is solvable on a given elementary function field $k$, then it is
    solvable in any algebraic extension of $k(\theta)$, where $\theta$ is
    a logarithm or exponential of an element of $k$. Our proof considers
    an element of such an extension field to be an algebraic function of
    one variable over $k$.

    In his algorithm for the integration of algebraic functions, Trager
    describes a Hermite-type reduction to reduce the problem to an
    integrand with only simple finite poles on the associated Riemann
    surface. We generalize that technique to curves over liouvillian
    ground fields, and use it to simplify our integrands.  Once the
    multipe finite poles have been removed, we use the Puiseux expansions
    of the integrand at infinity and a generalization of the residues to
    compute the integral. We also generalize a result of Rothstein that
    gives us a necessary condition for elementary integrability, and
    provide examples of its use.",
  paper = "Bron90a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bron90c,
  author = "Bronstein, Manuel",
  title = {{On the integration of elementary functions}},
  journal = "Journal of Symbolic Computation",
  volume = "9",
  number = "2",
  pages = "117-173",
  year = "1990",
  month = "February"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Bron93,
  author = "Bronstein, Manuel and Salvy, Bruno",
  title = {{Full partial fraction decomposition of rational functions}},
  booktitle = "Proc. ISSAC 1993",
  year = "1993",
  pages = "157-160",
  isbn = "0-89791-604-2",
  link = "\url{http://www.acm.org/pubs/citations/proceedings/issac/164081/}",
  algebra = "\newline\refto{domain FPARFRAC FullPartialFractionExpansion}",
  abstract =
    "We describe a rational algorithm that computes the full partial
    fraction expansion of a rational function over the algebraic closure
    of its field of definition. The algorithm uses only gcd operations
    over the initial field but the resulting decomposition is expressed
    with linear denominators. We give examples from its Axiom and Maple
    implementations.",
  paper = "Bron93.pdf",
  keywords = "axiomref",
  beebe = "Bronstein:1993:FPF"
}
```

———-

— **Bronstein:1993:FPF** —

```
@InProceedings{Bronstein:1993:FPF,
  author = "Manuel Bronstein and Bruno Salvy",
  title = {{Full Partial Fraction Decomposition of Rational Functions}},
  crossref = "Bronstein:1993:IPI",
  pages = "157--160",
  year = "1993",
  bibdate = "Thu Mar 12 08:40:26 MST 1998",
  bibsource = "http://www.acm.org/pubs/toc/;
               http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link = "\url{http://www.acm.org:80/pubs/citations/proceedings/issac/164081/p157-bronstein/}",
  abstract =     "We describe a rational algorithm that computes the
                 full partial fraction expansion of a rational function
                 over the algebraic closure of its field of definition.
                 The algorithm uses only gcd operations over the initial
                 field but the resulting decomposition is expressed with
                 linear denominators. We give examples from its Axiom
                 and Maple implementations.",
  acknowledgement = ack-nhfb,
  affiliation =  "Wissenschaftliches Rechnen, Eidgenossische Tech.
                 Hochschule, Zurich, Switzerland",
  classification = "B0290D (Functional analysis); B0290H (Linear
                 algebra); B0290M (Numerical integration and
                 differentiation); C4120 (Functional analysis); C4140
                 (Linear algebra); C4160 (Numerical integration and
                 differentiation); C7310 (Mathematics computing)",
  keywords =     "ACM; Algebraic closure; algebraic computation; Axiom;
```

```
                Decomposition; Full partial fraction decomposition; Gcd
                operations; Maple; Polynomial; Rational functions;
                SIGSAM; symbolic computation; Symbolic integration,
                ISSAC; theory; verification",
  subject =     "{\bf I.1.0} Computing Methodologies, SYMBOLIC AND
                ALGEBRAIC MANIPULATION, General. {\bf G.1.0}
                Mathematics of Computing, NUMERICAL ANALYSIS, General.
                {\bf I.1.2} Computing Methodologies, SYMBOLIC AND
                ALGEBRAIC MANIPULATION, Algorithms.",
  thesaurus =   "Function evaluation; Integration; Matrix
                decomposition; Polynomial matrices; Symbol
                manipulation",
}
```

———-

— **ignore** —

```
\bibitem[Bronstein 90]{Bro90b} Bronstein, Manuel
  title = {{A Unification of Liouvillian Extensions}},
  abstract = "
    We generalize Liouville's theory of elementary functions to a larger
    class of differential extensions. Elementary, Liouvillian and
    trigonometric extensions are all special cases of our extensions. In
    the transcendental case, we show how the rational techniques of
    integration theory can be applied to our extensions, and we give a
    unified presentation which does not require separate cases for
    different monomials.",
  paper = "Bro90b.pdf"
```

———-

— **axiom.bib** —

```
@book{Bron97,
  author = "Bronstein, Manuel",
  title = {{Symbolic Integration I--Transcendental Functions}},
  publisher = "Springer, Heidelberg",
  year = "1997",
  isbn = "3-540-21493-3",
  link = "\url{http://evil-wire.org/arrrXiv/Mathematics/Bronstein,_Symbolic_Integration_I,1997.pdf}",
  paper = "Bron97.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Bronstein 05a]{Bro05a} Bronstein, Manuel
  title = {{The Poor Man's Integrator, a parallel integration heuristic}},
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/pmint/pmint.txt}",
```

```
  url2 = "http://www-sop.inria.fr/cafe/Manuel.Bronstein/pmint/examples",
  paper = "Bro05a.txt"
```

———-

— **axiom.bib** —

```
@article{Bron06,
  author = "Bronstein, M.",
  title = {{Parallel integration}},
  journal = "Programming and Computer Software",
  year = "2006",
  issn = "0361-7688",
  volume = "32",
  number = "1",
  doi = "10.1134/S0361768806010075",
  link = "\url{http://dx.doi.org/10.1134/S0361768806010075}",
  publisher = "Nauka/Interperiodica",
  pages = "59-60",
  abstract = "
    Parallel integration is an alternative method for symbolic
    integration.  While also based on Liouville's theorem, it handles all
    the generators of the differential field containing the integrand ``in
    parallel'', i.e.  all at once rather than considering only the topmost
    one in a recursive fasion. Although it still contains heuristic
    aspects, its ease of implementation, speed, high rate of success, and
    ability to integrate functions that cannot be handled by the Risch
    algorithm make it an attractive alternative.",
  paper = "Bron06.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Bron07,
  author = "Bronstein, Manuel",
  title = {{Structure theorems for parallel integration}},
  journal = "Journal of Symbolic Computation",
  volume = "42",
  number = "7",
  pages = "757-769",
  year = "2007",
  month = "July",
  abstract = "
    We introduce structure theorems that refine Liouville's Theorem on
    integration in closed form for general derivations on multivariate
    rational function fields. By predicting the arguments of the new
    logarithms that an appear in integrals, as well as the denominator of
    the rational part, those theorems provide theoretical backing for the
    Risch-Norman integration method.  They also generalize its applicability
    to non-monomial extensions, for example the Lambert W function.",
```

```
  paper = "Bron07.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Charlwood 07]{Charl07} Charlwood, Kevin
  title = {{Integration on Computer Algebra Systems}},
The Electronic J of Math. and Tech. Vol 2, No 3, ISSN 1933-2823
  link = "\url{http://12000.org/my_notes/ten_hard_integrals/paper.pdf}",
  abstract = "
    In this article, we consider ten indefinite integrals and the ability
    of three computer algebra systems (CAS) to evaluate them in
    closed-form, appealing only to the class of real, elementary
    functions. Although these systems have been widely available for many
    years and have undergone major enhancements in new versions, it is
    interesting to note that there are still indefinite integrals that
    escape the capacity of these systems to provide antiderivatves. When
    this occurs, we consider what a user may do to find a solution with
    the aid of a CAS.",
  paper = "Charl07.pdf"
```

———-

— **ignore** —

```
\bibitem[Charlwood 08]{Charl08} Charlwood, Kevin
  title = {{Symbolic Integration Problems}},
  link = "\url{http://www.apmaths.uwo.ca/~arich/IndependentTestResults/CharlwoodIntegrationProblems.pdf}",
  abstract = "
    A list of the 50 example integration problems from Kevin Charlwood's 2008
    article ``Integration on Computer Algebra Systems''. Each integral along
    with its optimal antiderivative (that is, the best antiderivative found
    so far) is shown.",
  paper = "Charl08.pdf"
```

———-

— **axiom.bib** —

```
@inproceedings{Cher84,
  author = "Cherry, G.W. and Caviness, B.F.",
  title = {{Integration in Finite Terms With Special Functions:
            A Progress Report}},
  booktitle = "International Sympoium on Symbolic and Algebraic
               Manipulation",
  pages = "351-358",
  publisher = "Springer",
  year = "1984",
  comment = "LNCS 174",
```

```
  abstract =
    "Since R. Risch published an algorithm for calculating symbolic
    integrals of elementary functions in 1969, there has been an
    interest in extending his methods to include nonelementary
    functions. We report here on the recent development of two
    decision procedures for calculating integrals of transcendental
    elementary functions in terms of logarithmic integrals and error
    functions Both of these algorithms are based on the Singer,
    Saunders, Caviness extension of Liouville's theorem on integration
    in finite terms. Parts of the logarithmic integral algorithm have
    been implemented in Macsyma and a brief demonstraction is given.",
  paper = "Cher84.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Cher85,
  author = "Cherry, G.W.",
  title = {{Integration in Finite Terms with Special Functions:
           The Error Function}},
  journal = "J. Symbolic Computation",
  year = "1985",
  volume = "1",
  pages = "283-302",
  abstract =
    "A decision procedure for integrating a class of transcendental
    elementary functions in terms of elementary functions and error
    functions is described. The procedure consists of three mutually
    exclusive cases. In the first two cases a generalised procedure for
    completing squares is used to limit the error functions which can
    appear in the integral of a finite number.  This reduces the problem
    to the solution of a differential equation and we use a result of
    Risch (1969) to solve it.  The third case can be reduced to the
    determination of what we have termed $\sum$-decompositions. The resutl
    presented here is the key procuedure to a more general algorithm which
    is described fully in Cherry (1983).",
  paper = "Cher85.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Cherry 86]{Che86} Cherry, G.W.
  title = {{Integration in Finite Terms with Special Functions:
           The Logarithmic Integral}},
SIAM J. Comput. Vol 15 pp1-21 February 1986
```

———-

— **ignore** —

```
\bibitem[Cherry 89]{Che89} Cherry, G.W.
  title = {{An Analysis of the Rational Exponential Integral}},
SIAM J. Computing Vol 18 pp 893-905 (1989)
  abstract = "
    In this paper an algorithm is presented for integrating expressions of
    the form $\int{ge^f~dx}$, where $f$ and $g$ are rational functions of
    $x$, in terms of a class of special functions called the special
    incomplete $\Gamma$ functions. This class of special functions
    includes the exponential integral, the error functions, the sine and
    cosing integrals, and the Fresnel integrals. The algorithm presented
    here is an improvement over those published previously for integrating
    with special functions in the following ways: (i) This algorithm
    combines all the above special functions into one algorithm, whereas
    previously they were treated separately, (ii) Previous algorithms
    require that the underlying field of constants be algebraically
    closed. This algorithm, however, works over any field of
    characteristic zero in which the basic field operations can be carried
    out. (iii) This algorithm does not rely on Risch's solution of the
    differential equation $y^\prime + fy = g$. Instead, a more direct
    method of undetermined coefficients is used.",
  paper = "Che89.pdf"
```

———-

— **ignore** —

```
\bibitem[Churchill 06]{Chur06} Churchill, R.C.
  title = {{Liouville's Theorem on Integration Terms of Elementary Functions}},
  link = "\url{http://www.sci.ccny.cuny.edu/~ksda/PostedPapers/liouv06.pdf}",
  abstract = "
    This talk should be regarded as an elementary introduction to
    differential algebra. It culminates in a purely algebraic proof, due
    to M. Rosenlicht, of an 1835 theorem of Liouville on the existence of
    ``elementary'' integrals of ``elementary'' functions. The precise
    meaning of elementary will be specified. As an application of that
    theorem we prove that the indefinite integral $\int{e^{x^2}}~dx$
    cannot be expressed in terms of elementary functions.
    \begin{itemize}
    \item Preliminaries on Meromorphic Functions
    \item Basic (Ordinary) Differential Algebra
    \item Differential Ring Extensions with No New Constants
    \item Extending Derivations
    \item Integration in Finite Terms
    \end{itemize}",
  paper = "Chur06.pdf"
```

———-

— **axiom.bib** —

```
@article{Coll67,
  author = "Collins, George E.",
  title = {{Subresultants and Reduced Polynomial Remainder
           Sequences}},
  journal = "J. ACM",
  volume = "14",
  number = "1",
  pages = "128-142",
  year = "1967",
  paper = "Coll67.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Coll69,
  author = "Collins, George E.",
  title =
    {{Algorithmic Approaches to Symbolic Integration and SImplification}},
  journal = "ACM SIGSAM",
  volume = "12",
  year = "1969",
  pages = "5016",
  abstract =
    "This panel session followed the format announced by SIGSAM Chairman
    Carl Engelman in the announcement published in SIGSAM Bulletin No. 10
    (October 1968). Carl gave a brief (five or ten minutes) introduction
    to the subject and introduced Professor Joel Moses (M. I. T.). Joel
    presented an excellent exposition of the recent research
    accomplishments of the other panel members, synthesizing their work
    into a single large comprehensible picture. His presentation was
    greatly enhanced by a series of 27 carefully prepared slides
    containing critical examples and basic formulas, and was certainly the
    feature of the show. A panel discussion followed, with some audience
    participation. Panel members were Dr. W. S. Brown (Bell Telephone
    Laboratories), Professor B. F. Caviness (Duke University), Dr. Daniel
    Richardson and Dr. R. H. Risch (IBM).",
  paper = "Coll69.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Dave81,
  author = "Davenport, James H.",
  title = {{On the Integration of Algebraic Functions}},
  publisher = "Springer-Verlag",
  series = "Lecture Notes in Computer Science 102",
  isbn = "0-387-10290-6",
  year = "1981",
```

```
  abstract =
    "This work is concerned with the following question: ``{\sl When is an
    algebraic function integrable?}''. We can state this question in
    another form which makes clearer our interpretation of integration:
    ``If we are given an algebraic function, when can we find an
    expression in terms of algebraics, logarithms and exponentials whose
    derivative is the given function, and what is that expression?''.

    This question can be looked at purely mathematically, as a question in
    decidablility theory, but our interest in this question is more
    practical and springs from the requirements of computer algebra.  Thus
    our goal is ``{\sl Write a program which, when given an algebraic
    function, will produce an expression for its integral in terms of
    algebraics, exponentials and logarithms, or will prove that there is
    no such expression}''.",
  paper = "Dave81.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave81c,
  author = "Davenport, James H.",
  title = {{Algebraic Computations}},
  publisher = "Springer-Verlag",
  journal = "Lecture Notes in Computer Science 102",
  pages = "14-29",
  isbn = "0-387-10290-6",
  year = "1981",
  abstract =
    "Algebraic relationships between variables and expressions are very
    common in computer algebra. Not only do they often occur explicitly,
    in forms like $sqrt(x^2+1)$, but well known difficulties such as
    $sin(x)^2+cos(x)^2=1$ (Stoutemyer, 1977) can be expressed in this
    form. Nevertheless it is difficult to compute with regard to these
    relationships. This chapter discusses the problem of such computig,
    and then enters the area of algebraic geometry, which is a natural
    outgrowth of attempts to perform such computations as readily as one
    computes without them.",
  paper = "Dave81c.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave81d,
  author = "Davenport, James H.",
  title = {{Coates' Algorithm}},
  publisher = "Springer-Verlag",
  journal = "Lecture Notes in Computer Science 102",
```

```
  pages = "30-48",
  isbn = "0-387-10290-6",
  year = "1981",
  abstract =
    "In this chapter, we consider the problem of finding a function with a
    certain set of poles. That this problem is non-trivial in the case of
    algebraic functions (although it is trivial in the case of rational
    functions) can be seen from the fact that such functions need not
    always exist. For example, on the curve defined by $\sqrt{x^3+1}$,
    there is no function with a zero of order 1 at one place lying over
    the point $X=0$ and a pole of order 1 at infinity and no other poles
    or zeros, but there is one with divisor 3 times that (ie.e the divisor
    has order 3). On the curve defined by $Y^2=x^3-3X^2+X+1$, there are no
    functions with a zero on one place lying over $X=0$ and a pole at the
    other, both having the same order, and no other zeros or poles.",
  paper = "Dave81d.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave81e,
  author = "Davenport, James H.",
  title = {{Risch's Theorem}},
  publisher = "Springer-Verlag",
  journal = "Lecture Notes in Computer Science 102",
  pages = "49-63",
  isbn = "0-387-10290-6",
  year = "1981",
  abstract =
    "This chapter describes an underlying body of theory to the area of
    finding (or proving non-existent) the elementary integrals of
    algebraic functions, where a function is {\sl algebraic} if it can be
    generated from the variable of integration and constants by the
    arithmetic operations and the taking of roots of equations (the theory
    does not require that these roots should be expressible in terms of
    radicals), possibly with nesting. By {\sl elementary} we mean
    denerated from the variable of integration and constants by the
    arithmetic operations and the taking of roots, exponentials and
    logarithms, possibly with nesting.",
  paper = "Dave81e.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave81f,
  author = "Davenport, James H.",
  title = {{The Problem of Torsion Divisors}},
  publisher = "Springer-Verlag",
```

```
  journal = "Lecture Notes in Computer Science 102",
  pages = "64-75",
  isbn = "0-387-10290-6",
  year = "1981",
  abstract =
    "This chapter and the next three are concerned with the theory and
    practice of the FIND-ORDER procedure, which, as we saw in the last
    chapter, is a necessary part of our integration algorithm, and which
    turns out to be the mathematically most difficult. This chapter will
    outline the general nature of the problem, with special reference to
    the simplest non-trivial case, viz. problems involving the square root
    of one cubic or quartic and involving no constants other than the
    rationals.",
  paper = "Dave81f.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave81g,
  author = "Davenport, James H.",
  title = {{Gauss-Manin Operators}},
  publisher = "Springer-Verlag",
  journal = "Lecture Notes in Computer Science 102",
  pages = "76-91",
  isbn = "0-387-10290-6",
  year = "1981",
  abstract =
    "This chapter is devoted to the case of integrands which contain a
    transcendental parameter apart from the variable of integration, so
    that we can consider our problem to be the integration of a function
    in $\{K(x,y) | F(u,x,y) = 0\}$, where $K$ is an algebraic extension of
    $k(u)$ for some field $k$ and $u$ transcendental over it.  We shall
    use this notation, with $u$ being the independent transcendental, as
    we shall use the prefix operator $D$ to denote differentiation with
    respect to $u$, and the suffix $\prime$ to denote differentiation with
    respect to $x$. THis case is often more tractable than the case when
    there is no such transcendental, for integration with respect to $x$
    and differentiation with respect to $u$ commute, so that if $G(u,x,y0$
    is integrable, then so is $DG(u,x,y)$, $D^2G(u,x,y)$ and so on.",
  paper = "Dave81g.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave81h,
  author = "Davenport, James H.",
  title = {{Elliptic Integrals Concluded}},
  publisher = "Springer-Verlag",
```

```
  journal = "Lecture Notes in Computer Science 102",
  pages = "92-105",
  isbn = "0-387-10290-6",
  year = "1981",
  abstract =
    "The previous chapter (including the algorithm FIND\_ORDER\_MANIN)
    completely solved the problem of torsion divisors over ground fields
    containing a transcendental. We are therefore left with the case of
    ground fields all of whose elements are algebraic over the rationals,
    and this is the problem we will consider in this chapter (for elliptic
    curves) and the next. Furthermore, any particular definition of a
    curve and of a divisor can only involve a finite number of algebraics,
    so we can restrict our attention to fields which are generated from
    the rationals by extending with a finite number of algebraics, i.e.
    {\sl algebraic number fields}. Before we can explore the torsion
    divisor problem over them, we first need to know more about their
    structure and possible computer representations, and this we discuss
    in the next section, amplifying the discussion of general algebraic
    expression in Chapter 2.",
  paper = "Dave81h.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave81i,
  author = "Davenport, James H.",
  title = {{Curves over Algebraic Number Fields}},
  publisher = "Springer-Verlag",
  journal = "Lecture Notes in Computer Science 102",
  pages = "106-118",
  isbn = "0-387-10290-6",
  year = "1981",
  abstract =
    "The case of curves of arbitrary genus is much more difficult than the
    case of curves of genus 1, and there are no well-developed algorithms
    for this case. I have not been able to code any significant program to
    deal with this case because of the large number of subsidiary
    algorithms for which I do not have programs, though such programs have
    been written elsewhere, or can readily be written. Presented here,
    therefore, are the outlines of techniques which will enable one to
    bound the torsion of curves of arbitrary genus over algebraic number
    fields",
  paper = "Dave81i.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave79,
```

```
  author = "Davenport, J.H.",
  title = {{The Computerisation of Algebraic Geometry}},
  journal = "LNCS",
  volume = "72",
  pages = "119-133",
  year = "1979",
  abstract =
    "This paper is concerned with the problems of performing computer
    algebra when the variables involved are related by some algebraic
    dependencies. It is shown that heuristic or ad hoc treatment of
    such cases leads rapidly to problems, and the proper mathematical
    foundations for the treatment of algebraic functions is
    presented. The formalism leads directly to the requirement for
    algorithms to find the genus of an algebraic curve, and to
    discover what function, if any, is associated with a given
    divisor. These algorithms and the relevant computational
    techniques are briefly described. In a concluding section the
    areas where these techniques are required in an integration scheme
    for algebraic functions are explained.",
  paper = "Dave79.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave79c,
  author = "Davenport, James H.",
  title = {{Algorithms for the Integration of Algebraic Functions}},
  journal = "Lecture Notes in Computer Science",
  volume = "72",
  pages = "415-425",
  year = "1979",
  abstract = "
    The problem of finding elementary integrals of algebraic functions has
    long been recognized as difficult, and has sometimes been thought
    insoluble. Risch stated a theorem characterising the integrands with
    elementary integrals, and we can use the language of algebraic
    geometry and the techniques of Davenport to yield an algorithm that will
    always produce the integral if it exists. We explain the difficulty in
    the way of extending this algorithm, and outline some ways of solving
    it. Using work of Manin we are able to solve the problem in all cases
    where the algebraic expressions depend on a parameter as well as on
    the variable of integration.",
  paper = "Dave79c.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave82a,
```

```
  author = "Davenport, Jamess H.",
  title = {{The Parallel Risch Algorithm (I)}},
  journal = "Lecture Notes in Computer Science",
  volume = "144",
  pages = "144-157",
  year = "1982",
  abstract =
    "In this paper we review the so-called ``parallel Risch'' algorithm for
    the integration of transcendental functions, and explain what the
    problems with it are. We prove a positive result in the case of
    logarithmic integrands.",
  paper = "Dave82a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave85b,
  author = "Davenport, Jamess H. and Trager, Barry M.",
  title = {{The Parallel Risch Algorithm (II)}},
  journal = "ACM TOMS",
  volume = "11",
  number = "4",
  pages = "356-362",
  year = "1985",
  abstract =
    "It is proved that, under the usual restrictions, the denominator of
    the integral of a purely logarithmic function is the expected one,
    that is, all factors of the denominator of the integrand have their
    multiplicity decreased by one. Furthermore, it is determined which new
    logarithms may appear in the integration.",
  paper = "Dave85b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave82b,
  author = "Davenport, Jamess H.",
  title = {{The Parallel Risch Algorithm (III): use of tangents}},
  journal = "ACM SIGSAM",
  volume = "16",
  number = "3",
  pages = "3-6",
  year = "1982",
  abstract =
    "In this note, we look at the extension to the parallel Risch
    algorithm (see, e.g., the papers by Norman and Moore [1977], Norman and
    Davenport [1979], Fitch [1981] or Davenport [1982] for a description
    of the basic algorithm) which represents trigonometric functions in
```

```
    terms of tangents, rather than instead of complex exponentials.",
  paper = "Dave82b.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Davenport 03]{Dav03} Davenport, James H.
  title = {{The Difficulties of Definite Integration}},
  link = "\url{http://www.researchgate.net/publication/247837653_The_Diculties_of_Definite_Integration/file/72e7
  abstract = "
    Indefinite integration is the inverse operation to differentiation,
    and, before we can understand what we mean by indefinite integration,
    we need to understand what we mean by differentiation.",
  paper = "Dav03.pdf"
```

———-

— **axiom.bib** —

```
@article{Dave16,
  author = "Davenport, James H.",
  title =
     {{Complexity of Integration, Special Values, and Recent Developments}},
  journal = "LNCS",
  volume = "9725",
  pages = "485-491",
  year = "2016",
  abstract =
    "Two questions often come up when the author discusses integration:
    what is the complexity of the integration process, and for what
    special values of parameters is an unintegrable function actually
    integrable. These questions have not been much considered in the
    formal literature, and where they have been, there is one recent
    development indicating that the question is more delicate than had
    been supposed.",
  paper = "Dave16.pdf",
  keywords = "printed, DONE"
}
```

———-

— **ignore** —

```
\bibitem[Fateman 02]{Fat02} Fateman, Richard
  title = {{Symbolic Integration}},
  link = "\url{http://inst.eecs.berkeley.edu/~cs282/sp02/lects/14.pdf}",
  paper = "Fat02.pdf"
```

———-

— **axiom.bib** —

```
@inproceedings{Gedd89,
  author = "Geddes, K. O. and Stefanus, L. Y.",
  title = {{On the Risch-norman Integration Method and Its Implementation
           in MAPLE}},
  booktitle = "Proc. of the ACM-SIGSAM 1989 Int. Symp. on Symbolic and
               Algebraic Computation",
  series = "ISSAC '89",
  year = "1989",
  isbn = "0-89791-325-6",
  location = "Portland, Oregon, USA",
  pages = "212--217",
  numpages = "6",
  link = "\url{http://doi.acm.org/10.1145/74540.74567}",
  doi = "10.1145/74540.74567",
  acmid = "74567",
  publisher = "ACM",
  address = "New York, NY, USA",
  abstract = "
    Unlike the Recursive Risch Algorithm for the integration of
    transcendental elementary functions, the Risch-Norman Method processes
    the tower of field extensions directly in one step. In addition to
    logarithmic and exponential field extensions, this method can handle
    extentions in terms of tangents. Consequently, it allows trigonometric
    functions to be treated without converting them to complex exponential
    form. We review this method and describe its implementation in
    MAPLE. A heuristic enhancement to this method is also presented.",
  paper = "Gedd89.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Geddes 92a]{GCL92a} Geddes, K.O.; Czapor, S.R.; Labahn, G.
  title = {{The Risch Integration Algorithm}},
Algorithms for Computer Algebra, Ch 12 pp511-573 (1992)
  paper = "GCL92a.pdf"
```

———-

— **axiom.bib** —

```
@incollection{Grad80,
  author = "Gradshteyn, I.S. and Ryzhik, I.M.",
  title = {{Definite Integrals of Elementary Functions}},
  booktitle = "Table of Integrals, Series, and Products",
  publisher = "Academic Press",
  year = "1980",
  comment = "Chapter 3-4"
}
```

———-

— **ignore** —

```
\bibitem[Hardy 1916]{Hard16} Hardy, G.H.
  title = {{The Integration of Functions of a Single Variable}},
Cambridge Unversity Press, Cambridge, 1916
% REF:00002
```

———-

— **ignore** —

```
\bibitem[Harrington 78]{Harr87} Harrington, S.J.
  title = {{A new symbolic integration system in reduce}},
  link = "\url{http://comjnl.oxfordjournals.or/content/22/2/127.full.pdf}",
  abstract = "
    A new integration system, employing both algorithmic and pattern match
    integration schemes is presented. The organization of the system
    differs from that of earlier programs in its emphasis on the
    algorithmic approach to integration, its modularity and its ease of
    revision. The new Norman-Rish algorithm and its implementation at the
    University of Cambridge are employed, supplemented by a powerful
    collection of simplification and transformation rules. The facility
    for user defined integrals and functions is also included. The program
    is both fast and powerful, and can be easily modified to incorporate
    anticipated developments in symbolic integration.",
  paper = "Harr87.pdf"
```

———-

— **axiom.bib** —

```
@article{Hebi15,
  author = "Hebisch, Waldemer",
  title = {{Integration in terms of exponential integrals and incomplete
           gamma functions}},
  year = "2015",
  journal = "ACM Communications in Computer Algebra",
  volume = "49",
  Issue = "3",
  pages = "98-100",
  abstract =
    "Indefinite integration means that given $f$ in some set we want to
    find $g$ from possibly larger set such that $f = g^\prime$. When $f$
    and $g$ are required to be elementary functions due to work of among
    others Risch, Rothstein, Trager, Bronstein (see [1] for references)
    integration problem is now solved at least in theory. In his thesis
    Cherry gave algorithm to integrate transcendental elementary functions
    in terms of exponential integrals. In [2] he gave algorithm to
```

```
    integrate transcendental elementary functions in so called reduced
    fields in terms of error functions. Knowles [3] and [4] extended this
    allowing also liovillian integrands and weakened restrictions on the
    field containing integrands. We extend previous results allowing
    incomplete gamma function $\Gamma(a, x)$ with rational $a$. Also, our
    theory can handle algebraic extensions and is complete jointly (and
    not only separately for Ei and erf). In purely transcendental case our
    method should be more efficient and easier to implement than [2]. In
    fact, it seems that no system currently implements algorithm from [2],
    while partial implementation of our method in FriCAS works well enough
    to be turned on by default. With our approach non-reduced case from
    [2] can be handled easily. We hope that other classes of special
    functions can be handled in a similar way, in particular irrational
    case of incomplete gamma function and polylogarithms (however
    polylogarithms raise tricky theoretical questions).",
  paper = "Hebi15.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Herm1872,
  author = "Hermite, E.",
  title = {{Sur l'int\'{e}gration des fractions rationelles}},
  journal = "Nouvelles Annales de Math\'{e}matiques",
  volume = "11",
  pages = "145-148",
  year = "1872"
}
```

———-

— **ignore** —

```
\bibitem[Horowitz 71]{Horo71} Horowitz, Ellis
  title = {{Algorithms for Partial Fraction Decomposition and
           Rational Function Integration}},
SYMSAC '71 Proc. ACM Symp. on Symbolic and Algebraic Manipulation (1971)
pp441-457
  ref = "00018",
  abstract = "
    Algorithms for symbolic partial fraction decomposition and indefinite
    integration of rational functions are described. Two types of
    partial fraction decomposition are investigated, square-free and
    complete square-free. A method is derived, based on the solution of
    a linear system, which produces the square-free decomposition of any
    rational function, say A/B. The computing time is show to be
    $O(n^4(ln nf)^2)$ where ${\rm deg}(A) < {\rm\ deg}(B) = n$ and $f$
    is a number which is closely related to the size of the coefficients
    which occur in A and B. The complete square-free partical fraction
```

```
    decomposition can then be directly obtained and it is shown that the
    computing time for this process is also bounded by $O(n^4(ln nf)^2)$.",
  paper = "Horo71.pdf"
```

———-

— **ignore** —

```
\bibitem[Jeffrey 97]{Jeff97} Jeffrey, D.J.; Rich, A.D.
  title = {{Recursive integration of piecewise-continuous functions}},
  link = "\url{http://www.cybertester.com/data/recint.pdf}",
  abstract = "
    An algorithm is given for the integration of a class of
    piecewise-continuous functions. The integration is with respect to a
    real variable, because the functions considered do not in general
    allow integration in the complex plane to be defined. The class of
    integrands includes commonly occurring waveforms, such as square
    waves, triangular waves, and the floor function; it also includes the
    signum function.  The algorithm can be implemented recursively, and it
    has the property of ensuring that integrals are continuous on domains
    of maximum extent.",
  paper = "Jeff97.pdf"
```

———-

— **ignore** —

```
\bibitem[Jeffrey 99]{Jeff99} Jeffrey, D.J.; Labahn, G.; Mohrenschildt, M.v.;
Rich, A.D.
  title = {{Integration of the signum, piecewise and related functions}},
  link = "\url{http://cs.uwaterloo.ca/~glabahn/Papers/issac99-2.pdf}",
  abstract = "
    When a computer algebra system has an assumption facility, it is
    possible to distinguish between integration problems with respect to a
    real variable, and those with respect to a complex variable.  Here, a
    class of integration problems is defined in which the integrand
    consists of compositions of continuous functions and signum functions,
    and integration is with respect to a real variable.  Algorithms are
    given for evaluating such integrals.",
  paper = "Jeff99.pdf"
```

———-

— **axiom.bib** —

```
@misc{Kaha90,
  author = "Kahan, William",
  title = {{The Persistence of Irrationals in Some Integrals}},
  year = "1990",
  abstract =
    "Computer algebra systems are expected to simplify formulas they
```

```
    obtain for symbolic integrals whenever they can, and often they
    succeed. However, the formulas so obtained may then produce incorrect
    results for symblic definite integrals"
}
```

———-

— **axiom.bib** —

```
@TechReport{Kalt84b,
  author = "Kaltofen, E.",
  title = {{The Algebraic Theory of Integration}},
  institution = "RPI",
  address = "Dept. Comput. Sci., Troy, New York",
  year = "1984",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/84/Ka84_integration.pdf}",
  paper = "Kalt84b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kano76,
  author = "Kanoui, Henry",
  title = {{Some Aspects of Symbolic Integration via Predicate Logic
           Programming}},
  journal = "ACM SIGSAM",
  volume = "10",
  number = "4",
  year = "1976",
  pages = "29-42",
  abstract =
    "During the past years, various algebraic manipulations systems have
    been described in the literature. Most of them are implemented via
    ``classic'' programming languages like Fortran, Lisp, PL1 ... We propose
    an alternative approach: the use of Predicate Logic as a programming
    language.",
  paper = "Kano76.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kasp80,
  author = "Kasper, Toni",
  title = {{Integration in Finite Terms: The Liouville Theory}},
  journal = "ACM SIGSAM",
  volume = "14",
  number = "4",
```

```
  year = "1980",
  pages = "2-8",
  abstract =
    "The search for elementary antiderivatives leads from classical
    analysis through modern algebra to contemporary research in computer
    algorithms.",
  paper = "Kasp80.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Kiymaz 04]{Kiym04} Kiymaz, Onur; Mirasyedioglu, Seref
  title = {{A new symbolic computation for formal integration with exact
           power series}},
  abstract = "
    This paper describes a new symbolic algorithm for formal integration
    of a class of functions in the context of exact power series by using
    generalized hypergeometric series and computer algebraic technique.",
  paper = "Kiym04.pdf"
```

———-

— **axiom.bib** —

```
@article{Know93,
  author = "Knowles, Paul",
  title = {{Integration of a Class of Transcendental Liouvillian Functions
           with Error-Functions, Part I}},
  journal = "Journal of Symbolic Computation",
  volume = "13",
  number = "5",
  pages = "525-543",
  year = "1993",
  abstract =
    "This paper gives a decision-procedure for the symbolic integration of
    a certain class of transcendental Liouvillian functions in terms of
    elementary functions and error-functions. An example illustrating the
    use of the decision-procedure is given.",
  paper = "Know93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Know93a,
  author = "Knowles, Paul",
  title = {{Integration of a Class of Transcendental Liouvillian Functions
           with Error-Functions, Part II}},
```

```
  journal = "Journal of Symbolic Computation",
  volume = "16",
  number = "3",
  year = "1993",
  pages = "227-239",
  abstract =
    "This paper extends the decision procedure for the symbolic
    integration of a certain class of transcendental Liouvillian functions
    in terms of elementary functions and error-functions given in Knowles
    (1992) to allow a much larger class of integrands. Examples
    illustrating the use of the decision procedure are given.",
  paper = "Know93a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Krag09,
  author = "Kragler, R.",
  title = {{On Mathematica Program for Poor Man's Integrator Algorithm}},
  journal = "Programming and Computer Software",
  volume = "35",
  number = "2",
  pages = "63-78",
  year = "2009",
  issn = "0361-7688",
  abstract = "
    In this paper by means of computer experiment we study advantages and
    disadvantages of the heuristical method of ‘‘parallel integrator’’. For
    this purpose we describe and use implementation of the method in
    Mathematica. In some cases we compare this implementation with the original
    one in Maple.",
  paper = "Krag09.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Lang 93]{Lang93} Lang, S.
  title = {{Algebra}},
Addison-Wesly, New York, 3rd edition 1993
```

———-

— **ignore** —

```
\bibitem[Leerawat 02]{Leer02} Leerawat, Utsanee; Laohakosol, Vichian
  title = {{A Generalization of Liouville's Theorem on Integration in
           Finite Terms}},
```

```
  link = "\url{http://www.mathnet.or.kr/mathnet/kms_tex/113666.pdf}",
  abstract = "
    A generalization of Liouville's theorem on integration in finite
    terms, by enlarging the class of fields to an extension called
    Ei-Gamma extension is established. This extension includes the
    $\mathcal{E}\mathcal{L}$-elementary extensions of Singer, Saunders and
    Caviness and contains the Gamma function.",
  paper = "Leer02.pdf"
```

———-

— **ignore** —

```
\bibitem[Leslie 09]{Lesl09} Leslie, Martin
  title = {{Why you can't integrate exp($x^2$)}},
  link = "\url{http://math.arizona.edu/~mleslie/files/integrationtalk.pdf}",
  paper = "Lesl09.pdf"
```

———-

— **axiom.bib** —

```
@article{Lich11,
  author = "Lichtblau, Daniel",
  title = {{Symbolic Definite (and Indefinite) Integration: Methods and
           Open Issues}},
  journal = "ACM Comm. in Computer Algebra",
  volume = "45",
  number = "1",
  year = "2011",
  link = "\url{http://www.sigsam.org/bulletin/articles/175/issue175.pdf}",
  abstract = "
    The computation of definite integrals presents one with a variety of
    choices. There are various methods such as Newton-Leibniz or Slater's
    convolution method. There are questions such as whether to split or
    merge sums, how to search for singularities on the path of
    integration, when to issue conditional results, how to assess
    (possibly conditional) convergence, and more. These various
    considerations moreover interact with one another in a multitude of
    ways. Herein we discuss these various issues and illustrate
    with examples.",
  paper = "Lich11.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Liou1833a,
  author = "Liouville, Joseph",
  title = {{Premier m\'{e}moire sur la d\'{e}termination des int\'{e}grales
```

```
          dont la valeur est alg\'{e}brique}},
  journal = "Journal de l'Ecole Polytechnique",
  volume = "14",
  pages = "124-128",
  year = "1833"
}
```

———-

— **axiom.bib** —

```
@article{Liou1833b,
  author = "Liouville, Joseph",
  title = {{Second m\'{e}moire sur la d\'{e}termination des int\'{e}grales
          dont la valeur est alg\'{e}brique}},
  journal = "Journal de l'Ecole Polytechnique",
  volume = "14",
  pages = "149-193",
  year = "1833"
}
```

———-

— **ignore** —

```
\bibitem[Liouville 1833c]{Lio1833c} Liouville, Joseph
  title = {{Note sur la determination des int\'egrales dont la
           valeur est alg\'ebrique}},
Journal f\"ur die Reine und Angewandte Mathematik,
Vol 10 pp 247-259, (1833)
```

———-

— **ignore** —

```
\bibitem[Liouville 1833d]{Lio1833d} Liouville, Joseph
  title = {{Sur la determination des int\'egrales dont la
           valeur est alg\'ebrique}},
{\sl Journal de l'Ecole Polytechnique}, 14:124-193, 1833
```

———-

— **ignore** —

```
\bibitem[Liouville 1835]{Lio1835} Liouville, Joseph
  title = {{M\'emoire sur l'int\'gration d'une classe de fonctions
           transcendentes}},
Journal f\"ur die Reine und Angewandte Mathematik,
Vol 13(2) pp 93-118, (1835)
```

———-

— **axiom.bib** —

```
@article{Lope99,
  author = {L\'opez, Jos\'e L.},
  title = {{Asymptotic expansions of integrals: The term-by-term integration
           method}},
  year = "1999",
  journal = "Journal of Computational and Applied Mathematics",
  volume = "102",
  pages = "181-194",
  abstract =
    "The classical term-by-term integration technique used for obtaining
    asymptotic expansions of integrals requires the integrand to have an
    uniform asymptotic expansion in the integration variable. A
    modification of this method is presented in which the uniformity
    conditions provides the term-by-term integration technique a large
    range of applicability. As a consequence of this generality, Watson's
    lemma and the integration by parts technique applied to Laplace's and
    a special family of Fourier's transforms become corollaries of the
    term-by-term integration method.",
  paper = "Lope99.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Marc 94]{Marc94} Marchisotto, Elena Anne; Zakeri, Gholem-All
  title = {{An Invitation to Integration in Finite Terms}},
College Mathematics Journal Vol 25 No 4 (1994) pp295-308
  link = "\url{http://www.rangevoting.org/MarchisottoZint.pdf}",
  paper = "Marc94.pdf"
```

———-

— **ignore** —

```
\bibitem[Marik 91]{Mari91} Marik, Jan
  title = {{A note on integration of rational functions}},
  link = "\url{http://dml.cz/bitstream/handle/10338.dmlcz/126024/MathBohem_116-1991-4_9.pdf}",
  abstract = "
    Let $P$ and $Q$ be polynomials in one variable with complex coefficients
    and let $n$ be a natural number. Suppose that $Q$ is not constant and
    has only simple roots. Then there is a rational function $\varphi$
    with $\varphi^\prime=P/Q^{n+1}$ if and only if the Wronskian of the
    functions $Q^\prime$, $(Q^2)^\prime,\ldots\,(Q^n)^\prime$,$P$ is
    divisible by $Q$.",
  paper = "Mari91.pdf"
```

———-

— **axiom.bib** —

```
@book{Munr53,
  author = "Munroe, M.E.",
  title = {{Introduction to Measure and Integration}},
  publisher = "Addison-Wesley",
  year = "1953"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mose76,
  author = "Moses, Joel",
  title = {{An introduction to the Risch Integration Algorithm}},
  booktitle = "Proc. 1976 annual conference",
  publisher = "ACM",
  pages = "425-428",
  year = "1976",
  abstract =
    "Risch's decision procedure for determining the integrability in closed
    form of the elementary functions of the calculus is presented via
    examples.  The exponential and logarithmic cases of the algorithsm had
    been implemented for the MACSYMA system several years ago. The
    implementation of the algebraic case of the algorithm is the subject
    of current research.",
  paper = "Mose76.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mose71,
  author = "Moses, Joel",
  title = {{Symbolic Integration: The Stormy Decade}},
  journal = "CACM",
  year = "1971",
  volume = "14",
  number = "8",
  pages = "548-560",
  link =
  "\url{http://www-inst.eecs.berkeley.edu/~cs282/sp02/readings/moses-int.pdf}",
  abstract =
    "Three approaches to symbolic integration in the 1960's are
    described. The first, from artificial intelligence, led to Slagle's
    SAINT and to a large degree to Moses' SIN. The second, from algebraic
    manipulation, led to Monove's implementation and to Horowitz' and
    Tobey's reexamination of the Hermite algorithm for integrating
```

```
    rational functions. The third, from mathematics, led to Richardson's
    proof of the unsolvability of the problem for a class of functions and
    for Risch's decision procedure for the elementary functions.
    Generalizations of Risch's algorithm to a class of special
    functions and programs for solving differential equations and for
    finding the definite integral are also described.",
  paper = "Mos71a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Ngxx74,
  author = "Ng, Edward W.",
  title = {{Symbolic Integration of a Class of Algebraic Functions}},
  journal = "ACM SIGSAM",
  volume = "8",
  number = "3",
  year = "1974",
  pages = "99-102",
  abstract =
    "In this presentation we describe the outline of an algorithmic
    approach to handle a class of algebraic integrands. (It is important
    to stress that for an extended abstract of the present form, we can at
    best convey the flavor of the approach, with numerous details
    missing.) We shall label this approach Carlson's algorithm because it
    is based on a series of analyses rendered by Carlson and his
    associates in the last ten years (Refs. 2, 3, 4, 8, and 12). The class
    of integrands is of the form $r(x,y)$, where $y^2$ is a polynomial in $x$,
    and $r$ a rational function in $x$ and $y$. This is the type of integrand
    that classically led to the study of elliptic integrals. At first
    glance this is a rather restricted class of algebraic functions. But
    in fact many trigonometric and hyperbolic integrands reduce to this
    form. The richness of this class of integrands is exemplified by a
    recently published handbook of 3000 integral formulas (Ref. 1). Our
    proposed approach will cover fifty to seventy percent of the items in
    the handbook. Furthermore the non-classical approach we shall describe
    holds great promise of developing to the case where definite integrals
    can be evaluated in terms of a host of other well-known functions
    (e.g., Bessel and Legendre).",
  paper = "Ngxx74.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Ngxx71,
  author = "Ng, Edward W.",
  title = {{Observations on Approximate Integrations}},
```

```
  institution = "Jet Propulsion Lab",
  year = "1971",
  paper = "Ngxx71.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Norm79,
  author = "Norman, Arthur C. and Davenport, James H.",
  title = {{Symbolic Integration -- The Dust Settles?}},
  journal = "LNCS",
  volume = "72",
  pages = "398-407",
  year = "1979",
  abstract = "
    By the end of the 1960s it had been shown that a computer could find
    indefinite integrals with a competence exceeding that of typical
    undergraduates. This practical advance was backed up by algorithmic
    interpretations of a number of clasical results on integration, and by
    some significant mathematical extensions to these same results. At
    that time it would have been possible to claim that all the major
    barriers in the way of a complete system for automated analysis had
    been breached. In this paper we survey the work that has grown out of
    the above-mentioned early results, showing where the development has
    been smooth and where it has spurred work in seemingly unrelated fields.",
  paper = "Norm79.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Norm90,
  author = "Norman, Arthur C.",
  title = {{A Critical-Pair/Completion based Integration ALgorithm}},
  booktitle = "ISSAC 90",
  pages = "201-205",
  year = "1990",
  isbn = "0-201-54892-5",
  abstract =
    "In 1976 Risch [1] proposed a scheme for finding the integrals of
    forms built up out of transcendental functions that viewed general
    functions as rational forms in a suitable differential field and
    represented the polynomial parts of those forms in a distributed
    rather than recursive way. By using a data representation where all
    variables were (more or less) equally important this new method seemed
    to side-step some of the complications that had appeared in his
    previous scheme [2] where various side-constraints had to be
    propagated between the levels present in a tower of separate
    extensions of differential fields, otherwise seen as levels in
```

```
    recursive data structures.

    An initial implementation of the method was
    prepared in the context of the SCRATCHPAD/1 algebra system and
    demonstrated at the 1976 SYMSAC meeting at Yorktown Heights, a
    subsequent version for Reduce [3][5] came after that, and made it
    possible to try the method on a large range of integrals. These
    practical studies showed up some problems with the method and its
    implementation.

    The presentation given here re-expresses the 1976
    Risch method in terms of rewrite rules, and thus exposes the major
    problem it suffers from as a manifestation of the fact that in certain
    circumstances the set of rewrites generated is not confluent. This
    difficulty is then attacked using a critical-pair/completion (CPC)
    approach. For very many integrands it is then easy to see that the
    initial set of rewrites used in the early implementations [1] and [3]
    do not need any extension, and this fact explains the high level of
    competence of the programs involved despite their shaky theoretical
    foundations. For a further large collection of problems even a simple
    CPC scheme converges rapidly; when the techniques presented here are
    applied to the REDUCE integration test suite in all applicable cases a
    short computation succeeds in completing the set of rewrites and hence
    gives a secure basis for testing for integrability.

    This paper describes the implementation of the CPC process and
    discusses current limitations to and possible future extended
    applications of it.",
  paper = "Norm90.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Ostrowski 46]{Ost46} Ostrowski, A.
  title = {{Sur l'int\'egrabilit\'e \'el\'ementaire de quelques
           classes d'expressions}},
Comm. Math. Helv., Vol 18 pp 283-308, (1946)
% REF:00008
```

———-

— **ignore** —

```
\bibitem[Raab 12]{Raab12} Raab, Clemens G.
  title = {{Definite Integration in Differential Fields}},
  link = "\url{http://www.risc.jku.at/publications/download/risc_4583/PhD_CGR.pdf}",
  abstract = "
    The general goal of this thesis is to investigate and develop computer
    algebra tools for the simplification resp. evaluation of definite
```

```
   integrals. One way of finding the value of a definite integral is
   via the evaluation of an antiderivative of the integrand. In the
   nineteenth century Joseph Liouville was among the first who analyzed
   the structure of elementary antiderivatives of elementary functions
   systematically. In the early twentieth century the algebraic structure
   of differential fields was introduced for modeling the differential
   properties of functions. Using this framework Robert H. Risch
   published a complete algorithm for transcendental elementary
   integrands in 1969. Since then this result has been extended to
   certain other classes of integrands as well by Michael F. Singer,
   Manuel Bronstein, and several others. On the other hand, if no
   antiderivative of suitable form is available, then linear relations
   that are satisfied by the parameter integral of interest may be found
   based on the principle of parametric integration (often called
   differentiating under the integral sign or creative telescoping).

   The main result of this thesis extends the results mentioned above to
   a complete algorithm for parametric elementary integration for a
   certain class of integrands covering a majority of the special
   functions appearing in practice such as orthogonal polynomials,
   polylogarithms, Bessel functions, etc. A general framework is provided
   to model those functions in terms of suitable differential fields. If
   the integrand is Liouvillian, then the present algorithm considerably
   improves the efficiency of the corresponding algorithm given by Singer
   et al. in 1985. Additionally, a generalization of Czichowskis
   algorithm for computing the logarithmic part of the integral is
   presented. Moreover, also partial generalizations to include other
   types of integrands are treated.

   As subproblems of the integration algorithm one also has to find
   solutions of linear ordinary differential equations of a certain
   type. Some contributions are also made to solve those problems in our
   setting, where the results directly dealing with systems of
   differential equations have been joint work with Moulay A. Barkatou.

   For the case of Liouvillian integrands we implemented the algorithm in
   form of our Mathematica package Integrator. Parts of the
   implementation also deal with more general functions. Our procedures
   can be applied to a significant amount of the entries in integral
   tables, both indefinite and definite integrals. In addition, our
   procedures have been successfully applied to interesting examples of
   integrals that do not appear in these tables or for which current
   standard computer algebra systems like Mathematica or Maple do not
   succeed. We also give examples of how parameter integrals coming from
   the work of other researchers can be solved with the software, e.g.,
   an integral arising in analyzing the entropy of certain processes.",
  paper = "Raab12.pdf"
```

———-

— ignore —

```
\bibitem[Raab 13]{Raab13} Raab, Clemens G.
```

```
  title = {{Generalization of Risch's Algorithm to Special Functions}},
  link = "\url{http://arxiv.org/pdf/1305.1481.pdf}",
  abstract = "
    Symbolic integration deals with the evaluation of integrals in closed
    form. We present an overview of Risch's algorithm including recent
    developments. The algorithms discussed are suited for both indefinite
    and definite integration. They can also be used to compute linear
    relations among integrals and to find identities for special functions
    given by parameter integrals. The aim of this presentation is twofold:
    to introduce the reader to some basic idea of differential algebra in
    the context of integration and to raise awareness in the physics
    community of computer algebra algorithms for indefinite and definite
    integration.",
  paper = "Raab13.pdf"
```

———-

— **ignore** —

```
\bibitem[Raab xx]{Raabxx} Raab, Clemens G.
  title = {{Integration in finite terms for Liouvillian functions}},
  link = "\url{http://www.mmrc.iss.ac.cn/~dart4/posters/Raab.pdf}",
  abstract = "
    Computing integrals is a common task in many areas of science,
    antiderivatives are one way to accomplish this.  The problem of
    integration in finite terms can be states as follows. Given a
    differential field $(F,D)$ and $f \in F$, compute $g$ in some
    elementary extension of $(F,D)$ such that $Dg = f$ if such a $g$
    exists.

    This problem has been solved for various classes of fields $F$.  For
    rational functions $(C(x), \frac{d}{dx})$ such a $g$ always exists and
    algorithms to compute it are known already for a long time. In 1969
    Risch published an algorithm that solves this problem when $(F,D)$ is
    a transcendental elementary extension of $(C(x),\frac{d}{dx})$. Later
    this has been extended towards integrands being Liouvillian functions
    by Singer et. al. via the use of regular log-explicit extensions of
    $(C(x),\frac{d}{dx})$. Our algorithm extends this to handling
    transcendental Liouvillian extensions $(F,D)$ of $(C,0)$ directly
    without the need to embed them into log-explicit extensions. For
    example, this means that
    \[\int{(z-x)x^{z-1}e^{-x}dx} = x^ze^{-x}\]
    can be computed without including log(x) in the differential field.",
  paper = "Raabxx.pdf"
```

———-

— **axiom.bib** —

```
@article{Renb82,
  author = "Renbao, Zhong",
  title = {{An Algorithm for Avoiding Complex Numbers in Rational Function
```

```
          Integration}},
  journal = "ACM SIGSAM",
  volume = "16",
  number = "3",
  pages = "30-32",
  year = "1982",
  abstract =
    "Given a proper rational function $A(x)/B(x)$ where $A(x)$ and $B(x)$
    both are in $R[x]$ with $gcd(A(x), B(x))= 1$, $B(x)$ monic and
    $deg(A(x)) < deg(B(x))$, from the Hermite algorithm for rational
    function integration in [3], we obtain
    \[\int{frac{A(x)}{B(x)}~dx} = S(x)+\int{\frac{T(x)}{B^*(x)}~dx}\]
    where $S(x)$ is a rational function
    which is called the rational part of the integral of $A(x)/B(x)$ in
    eq. (1), $B^*(x)$ is the greatest square-free factor of $B(x)$, and
    $T(x)$ is in $R[x]$ with $deg(T(x)) < deg(B^*(x))$.  The integral of
    $T(x)/B^*(x)$ is called the transcendental part of the integral of
    $A(x)/B(x)$ in eq. (1).",
  paper = "Renb82.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Rich09,
  author = "Rich, Albert D. and Jeffrey, David J.",
  title = {{A Knowledge Repository for Indefinite Integration
          Based on Transformation Rules}},
  journal = "LNCS",
  volume = "5625",
  year = "2009",
  abstract = "
    Taking the specific problem domain of indefinite integration, we
    describe the on-going development of a repository of mathematical
    knowledge based on transformation rules. It is important that the
    repository be not confused with a look-up table. The database of
    transformation rules is at present encoded in Mathematica, but this is
    only one convenient form of the repository, and it could be readily
    translated into other formats. The principles upon which the set of
    rules is compiled is described. One important principle is
    minimality. The benefits of the approach are illustrated with
    examples, and with the results of comparisons with other approaches.",
  paper = "Rich09.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Risc68,
  author = "Risch, Robert",
```

```
  title = {{On the integration of elementary functions which are built up
           using algebraic operations}},
  type = "Research Report",
  number = "SP-2801/002/00",
  institution = "System Development Corporation, Santa Monica, CA, USA",
  year = "1968",
  abstract =
    "This paper advances the study of the problem of integration of
    elementary functions in finite terms to within one step of a
    complete solution. A previous paper gave an algorithm for
    integrating those elementary functions which are built up using
    rational operations, exponentials and logarithms, under the
    condition that the exponentials and logarithms could not be
    replaced by adjoining constants and performing algebraic
    operations. Now it is show that with algebraic operations allowed,
    the problem reduces to a problem in the theory of algebraci
    functions which is believed to be decidable."
}
```

———-

— **axiom.bib** —

```
@techreport{Risc69a,
  author = "Risch, Robert",
  title = {{Further results on elementary functions}},
  type = "Research Report",
  number = "RC-2042",
  institution = "IBM Research, Yorktown Heights, NY, USA",
  year = "1969"
}
```

———-

— **axiom.bib** —

```
@article{Risc69b,
  author = "Risch, Robert",
  title = {{The problem of integration in finite terms}},
  journal = "Transactions of the American Mathematical Society",
  volume = "139",
  year = "1969",
  pages = "167-189",
  abstract = "This paper deals with the problem of telling whether a
    given elementary function, in the sense of analysis, has an elementary
    indefinite integral.",
  paper = "Ris69b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Risc70,
  author = "Risch, Robert",
  title = {{The Solution of the Problem of Integration in Finite Terms}},
  journal = "Bull. AMS",
  year = "1970",
  issn = "0002-9904",
  volume = "76",
  number = "3",
  pages = "605-609",
  abstract = "
    The problem of integration in finite terms asks for an algorithm for
    deciding whether an elementary function has an elementary indefinite
    integral and for finding the integral if it does.  ``Elementary'' is
    used here to denote those functions build up from the rational
    functions using only exponentiation, logarithms, trigonometric,
    inverse trigonometric and algebraic operations.  This vaguely worded
    question has several precise, but inequivalent formulations. The
    writer has devised an algorithm which solves the classical problem of
    Liouville. A complete account is planned for a future publication. The
    present note is intended to indiciate some of the ideas and techniques
    involved.",
  paper = "Risc70.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Ritt48,
  author = "Ritt, Joseph Fels",
  title = {{Integration in Finite Terms}},
  publisher = "Columbia University Press, New York",
  year = "1948",
  keywords = "shelf"
}
```

———-

— **ignore** —

```
\bibitem[Rosenlicht 68]{Ro68} Rosenlicht, Maxwell
  title = {{Liouville's Theorem on Functions with Elementary Integrals}},
Pacific Journal of Mathematics Vol 24 No 1 (1968)
  link = "\url{http://msp.org/pjm/1968/24-1/pjm-v24-n1-p16-p.pdf}",
  ref = "00047",
  abstract = "
    Defining a function with one variable to be elemetary if it has an
    explicit representation in terms of a finite number of algebraic
    operations, logarithms, and exponentials. Liouville's theorem in its
    simplest case says that if an algebraic function has an elementary
    integral then the latter is itself an algebraic function plus a sum of
```

```
    constant multiples of logarithms of algebraic functions. Ostrowski has
    generalized Liouville's results to wider classes of meromorphic
    functions on regions of the complex plane and J.F. Ritt has given the
    classical account of the entire subject in his Integraion in Finite
    Terms, Columbia University Press, 1948. In spite of the essentially
    algebraic nature of the problem, all proofs so far have been analytic.
    This paper gives a self contained purely algebraic exposition of the
    probelm, making a few new points in addition to the resulting
    simplicity and generalization.",
  paper = "Ro68.pdf"
```

———-

— **axiom.bib** —

```
@article{Rose72,
  author = "Rosenlicht, Maxwell",
  title = {{Integration in finite terms}},
  journal = "American Mathematical Monthly",
  year = "1972",
  volume = "79",
  pages = "963-972",
  paper = "Rose72.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Roth76,
  author = "Rothstein, Michael",
  title = {{Aspects of symbolic integration and simplifcation of
           exponential and primitive functions}},
  school = "University of Wisconsin-Madison",
  year = "1976",
  link = "\url{http://www.cs.kent.edu/~rothstei/dis.pdf}",
  abstract =
    "In this thesis we cover some aspects of the theory necessary to obtain
    a canonical form for functions obtained by integration and
    exponentiation from the set of rational functions.

    These aspects include a new algorithm for symbolic integration of
    functions involving logarithms and exponentials which avoids
    factorization of polynomials in those cases where algebraic extension
    of the constant field is not required, avoids partial fraction
    decompositions, and only solves linear systems with a small number of
    unknowns.

    We have also found a theorem which states, roughly speaking, that if
    integrals which can be represented as logarithms are represented as
    such, the only algebraic dependence that a new exponential or
    logarithm can satify is given by the law of exponents or the law of
```

```
    logarithms.",
  paper = "Ro76.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Ro76a,
  author = "Rothstein, Michael and  Caviness, Bob F.",
  title = {{A structure theorem for exponential and primitive functions:
           a preliminary report}},
  journal = "ACM Sigsam Bulletin",
  volume = "10",
  number = "4",
  year = "1976",
  abstract =
    "In this paper a generalization of the Risch Structure Theorem is reported.
    The generalization applies to fields $F(t_1,\ldots,t_n)$ where $F$
    is a differential field (in our applications $F$ will be a finitely
    generated extension of $Q$, the field of rational numbers) and each $t_i$
    is either algebraic over $F_{i-1}=F(t_1,\ldots,t_{i-1})$, is an
    exponential of an element in $F_{i-1}$, or is an integral of an element
    in $F_{i-1}$. If $t_i$ is an integral and can be expressed using
    logarithms, it must be so expressed for the generalized structure
    theorem to apply.",
  paper = "Ro76a.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Rothstein 76b]{Ro76b} Rothstein, Michael; Caviness, B.F.
  title = {{A structure theorem for exponential and primitive functions}},
SIAM J. Computing Vol 8 No 3 (1979)
  ref = "00104",
  abstract = "
    In this paper a new theorem is proved that generalizes a result of
    Risch.  The new theorem gives all the possible algebraic relationships
    among functions that can be built up from the rational functions by
    algebraic operations, by taking exponentials, and by integration. The
    functions so generated are called exponential and primitive functions.
    From the theorem an algorithm for determining algebraic dependence
    among a given set of exponential and primitive functions is derived.
    The algorithm is then applied to a problem in computer algebra.",
  paper = "Ro76b.pdf"
```

———-

— **axiom.bib** —

```
@article{Roth77,
  author = "Rothstein, Michael",
  title = {{A new algorithm for the integration of exponential and
           logarithmic functions}},
  journal = "Proceedings of the 1977 MACSYMA Users Conference",
  year = "1977",
  pages = "263-274",
  publisher = "NASA Pub CP-2012"
}
```

———-

— **axiom.bib** —

```
@article{Scho89,
  author = "Schou, Wayne C. and Broughan, Kevin A.",
  title = {{The Risch Algorithms of MACSYMA and SENAC}},
  journal = "ACM SIGSAM",
  volume = "23",
  number = "3",
  year = "1989",
  abstract =
    "The purpose of this paper is to report on a computer implementation
    of the Risch algorithm for the symbolic integration of rational
    functions containing nested exponential and logarithms. For the class
    of transcendental functions, the Risch algorithm [4] represents a
    practical method for symbolic integration. Because the Risch algorithm
    describes a decision procedure for transcendental integration it is an
    ideal final step in an integration package. Although the decision
    characteristic cannot be fully realised in a computer system, because
    of major algebraic problems such as factorisation, zero-equivalence
    and simplification, the potential advantages are considerable.",
  paper = "Scho89.pdf",
}
```

———-

— **ignore** —

```
\bibitem[Seidenberg 58]{Sei58} Seidenberg, Abraham
  title = {{Abstract differential algebra and the analytic case}},
Proc. Amer. Math. Soc. Vol 9 pp159-164 (1958)
```

———-

— **ignore** —

```
\bibitem[Seidenberg 69]{Sei69} Seidenberg, Abraham
  title = {{Abstract differential algebra and the analytic case. II}},
Proc. Amer. Math. Soc. Vol 23 pp689-691 (1969)
```

———-

— **axiom.bib** —

```
@article{Sing85,
  author = "Singer, Michael F. and  Saunders, B. David and Caviness, Bob F.",
  title =
    {{An extension of Liouville's theorem on integration in finite terms}},
  journal = "SIAM J. of Comp.",
  volume = "14",
  pages = "965-990",
  year = "1985",
  link = "\url{http://www4.ncsu.edu/~singer/papers/singer_saunders_caviness.pdf}",
  abstract =
    "In Part 1 of this paper, we give an extension of Liouville's Theorem
    and give a number of examples which show that integration with special
    functions involves some phenomena that do not occur in integration
    with the elementary functions alone. Our main result generalizes
    Liouville's Theorem by allowing, in addition to the elementary
    functions, special functions such as the error function, Fresnel
    integrals and the logarithmic integral (but not the dilogarithm or
    exponential integral) to appear in the integral of an elementary
    function. The basic conclusion is that these functions, if they
    appear, appear linearly. We give an algorithm which decides if an
    elementary function, built up using only exponential functions and
    rational operations has an integral which can be expressed in terms of
    elementary functions and error functions.",
  paper = "Sing85.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Slagle 61]{Slag61} Slagle, J.
  title = {{A heuristic program that solves symbolic integration problems in
           freshman calculus}},
Ph.D Diss. MIT, May 1961; also Computers and Thought, Feigenbaum and Feldman.
% REF:00014
```

———-

— **axiom.bib** —

```
@article{Smit83,
  author = "Smith, Paul and Sterling, Leon",
  title = {{Of Integration by Man and Machine}},
  journal = "ACM SIGSAM",
  volume = "17",
  number = "3-4",
  year = "1983",
  abstract =
```

```
    "We describe a symbolic integration problem arising from an
    application in engineering. A solution is given and compared with the
    solution generated by the REDUCE integration package running at
    Cambridge. Nontrivial symbol manipulation, particularly
    simplification, is necessary to reconcile the answers.",
  paper = "Smit83.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Temmxx,
  author = "Temme, N.M.",
  title = {{Uniform Asymptotic Expansions of Integrals}},
  abstract =
    "The purpose of the paper is to give an account of several aspects of
    uniform asymptotic expansions of integrals. We give examples of
    standard forms, the role of critical points and methods to construct
    the experiences."
}
```

———-

— **axiom.bib** —

```
@article{Temm95,
  author = "Temme, N.M.",
  title = {{Uniform asymptotic expansions of integrals: a selection of
           problems}},
  journal = "Journal of Computational and Applied Mathematics",
  volume = "65",
  number = "1-3",
  year = "1995",
  pages = "395-417",
  abstract =
    "On the occasion of the conference we mention examples of Stieltjes'
    work on asymptotics of special functions. The remaining part of the
    paper gives a selection of asymptotic methods for integrals, in
    particular on uniform approximations. We discuss several standard
    problems and examples, in which known special functions (error
    functions, Airy functions, Bessel functions, etc.) are needed to
    construct uniform approximations. Finally, we discuss the recent
    interest and new insights in the Stokes phenomenon. An extensive
    bibliography on uniform asymptotic methods for integrals is given,
    together with references to recent papers on the Stokes phenomenon for
    integrals and related topics.",
  paper = "Temm95.pdf"

}
```

———-

— **axiom.bib** —

```
@mastersthesis{Tere09,
  author = "Terelius, Bjorn",
  title = {{Symbolic Integration}},
  school = "Royal Institute of Technology",
  address = "Stockholm, Sweden",
  year = "2009",
  abstract =
    "Symbolic integration is the problem of expressing an indefinite integral
    $\int{f}$ of a given function $f$ as a finite combination $g$ of elementary
    functions, or more generally, to determine whether a certain class of
    functions contains an element $g$ such that $gˆ\prime = f$.

    In the first part of this thesis, we compare different algorithms for
    symbolic integration. Specifically, we review the integration rules
    taught in calculus courses and how they can be used systematically to
    create a reasonable, but somewhat limited, integration method. Then we
    present the differential algebra required to prove the transcendental
    cases of Risch's algorithm. Risch's algorithm decides if the integral
    of an elementary function is elementary and if so computes it. The
    presentation is mostly self-contained and, we hope, simpler than
    previous descriptions of the algorithm. Finally, we describe
    Risch-Norman's algorithm which, although it is not a decision
    procedure, works well in practice and is considerably simpler than the
    full Risch algorithm.

    In the second part of this thesis, we briefly discuss an
    implementation of a computer algebra system and some of the
    experiences it has given us.  We also demonstrate an implementation of
    the rule-based approach and how it can be used, not only to compute
    integrals, but also to generate readable derivations of the results.",
  paper = "Tere09.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Trag76,
  author = "Trager, Barry",
  title = {{Algebraic factoring and rational function integration}},
  journal = "Proceedings of SYMSAC'76",
  year = "1976",
  pages = "219-226",
  abstract = "
    This paper presents a new, simple, and efficient algorithm for
    factoring polynomials in several variables over an algebraic number
    field. The algorithm is then used interatively to construct the
    splitting field of a polynomial over the integers. Finally the
    factorization and splitting field algorithms are applied to the
    problem of determining the transcendental part of the integral of a
    rational function. In particular, a constructive procedure is given
```

```
    for finding a least degree extension field in which the integral can
    be expressed.",
  paper = "Trag76.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Trager 76a]{Tr76a} Trager, Barry Marshall
  title = {{Algorithms for Manipulating Algebraic Functions}},
MIT Master's Thesis.
  link = "\url{http://www.dm.unipi.it/pages/gianni/public_html/Alg-Comp/fattorizzazione-EA.pdf}",
  ref = "00050",
  abstract = "
    Given a base field $k$, of characteristic zero, with effective
    procedures for performing arithmetic and factoring polynomials, this
    thesis presents algorithms for extending those capabilities to
    elements of a finite algebraic symbolic manipulation system. An
    algebraic factorization algorithm along with a constructive version of
    the primitive element theorem is used to construct splitting fields of
    polynomials. These fields provide a context in which we can operate
    symbolically with all the roots of a set of polynomials. One
    application for this capability is rational function integrations.
    Previously presented symbolic algorithms concentrated on finding the
    rational part and were only able to compute the complete
    integral in special cases. This thesis presents an algorithm for
    finding an algebraic extension field of least degreee in which the
    integral can be expressed, and then constructs the integral in that
    field.  The problem of algebraic function integration is also
    examined, and a highly efficient procedure is presented for generating
    the algebraic part of integrals whose function fields are defined by a
    single radical extension of the rational functions.",
  paper = "Tr76a.pdf"
```

———-

— **axiom.bib** —

```
@phdthesis{Trag84,
  author = "Trager, Barry",
  title = {{Integration of Algebraic Functions}},
  school = "MIT",
  year = "1984",
  link = "\url{http://www.dm.unipi.it/pages/gianni/public_html/Alg-Comp/thesis.pdf}",
  abstract = "
    We show how the ``rational'' approach for integrating algebraic
    functions can be extended to handle elementary functions. The
    resulting algorithm is a practical decision procedure for determining
    whether a given elementary function has an elementary antiderivative,
    and for computing it if it exists.",
  paper = "Trag76.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@phdthesis{Wang71,
  author = "Wang, Paul S.",
  title = {{Evaluation of Definite Integrals by Symbolic Manipulation}},
  school = "MIT",
  year = "1971",
  link =
    "\url{http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-092.pdf}",
  comment = "MIT/LCS/TR-92",
  abstract =
    "A heuristic computer program for the evaluation of real definite
    integrals of elementary functions is described This program, called
    WANDERER, (WANg's DEfinite integRal EvaluatoR), evaluates many proper
    and improper integrals. The improper integrals may have a finite or
    infinite range of integration. Evaluation by contour integration and
    residue theory is among the methods used. A program called DELIMITER
    (DEfinitive LIMIT EvaluatoR) is used for the limit computations needed
    in evaluating some definite integrals. DELIMITER is a heuristic
    program written for computing limits of real or complex analytic
    functions. For real functions of a real variable, one-sided as well
    been implmented in the MACSYMA system, a symbolic and algebraic
    manipulation system being developed at Project MAC, MIT. A typical
    problem in applied mathematics, namely asymptotic analysis of a
    definite integral, is solved using MACSYMA to demonstrate the
    usefulness of such a system and the facilities provided by WANDERER.",
  paper = "Wang71.pdf"
}
```

———-

— **ignore** —

```
\bibitem[W\"urfl 07]{Wurf07} W\"urfl, Andreas
  title = {{Basic Concepts of Differential Algebra}},
  link = "\url{http://www14.in.tum.de/konferenzen/Jass07/courses/1/Wuerfl/wuerfl_paper.pdf}",
  abstract = "
    Modern computer algebra systems symbolically integrate a vast variety
    of functions. To reveal the underlying structure it is necessary to
    understand infinite integration not only as an analytical problem but
    as an algebraic one. Introducing the differential field of elementary
    functions we sketch the mathematical tools like Liouville's Principle
    used in modern algorithms. We present Hermite's method for integration
    of rational functions as well as the Rothstein/Trager method for
    rational and for elementary functions. Further applications of the
    mentioned algorithms in the field of ODE's conclude this paper.",
  paper = "Wurf07.pdf"
```

———-

1.26 Partial Fraction Decomposition

— **ignore** —

```
\bibitem[Angell]{Angell} Angell, Tom
  title = {{Guidelines for Partial Fraction Decomposition}},
  link = "\url{http://www.math.udel.edu/~angell/partfrac_I.pdf}",
  paper = "Angell.pdf"
```

———-

— **ignore** —

```
\bibitem[Laval 08]{Lava08} Laval, Philippe B.
  title = {{Partial Fractions Decomposition}},
  link = "\url{http://www.math.wisc.edu/~park/Fall2011/integration/Partial%20Fraction.pdf}",
  paper = "Lava08.pdf"
```

———-

— **ignore** —

```
\bibitem[Mudd 14]{Mudd14} Harvey Mudd College
  title = {{Partial Fractions}},
  link = "\url{http://www.math.hmc.edu/calculus/tutorials/partial_fractions/partial_fractions.pdf}",
  paper = "Mudd14.pdf"
```

———-

— **ignore** —

```
\bibitem[Rajasekaran 14]{Raja14} Rajasekaran, Raja
  title = {{Partial Fraction Expansion}},
  link = "\url{http://www.utdallas.edu/~raja1/EE4361%20Spring%2014/Lecture%20Notes/Partial%20Fractions.pdf}",
  paper = "Raja14.pdf"
```

———-

— **ignore** —

```
\bibitem[Wootton 14]{Woot14} Wootton, Aaron
  title = {{Integration of Rational Functions by Partial Fractions}},
  link = "\url{http://faculty.up.edu/wootton/calc2/section7.4.pdf}",
  paper = "Woot14.pdf"
```

———-

1.27 Ore Rings

This is used as a reference for the LeftOreRing category, in particular, the least left common multiple (lcmCoef) function.

— **ignore** —

```
\bibitem[Abramov 97]{Abra97} Abramov, Sergei A.; van Hoeij, Mark
  title = {{A method for the Integration of Solutions of Ore Equations}},
Proc ISSAC 97 pp172-175 (1997)
  abstract = "
    We introduce the notion of the adjoint Ore ring and give a definition
    of adjoint polynomial, operator and equation. We apply this for
    integrating solutions of Ore equations.",
  paper = "Abra97.pdf"
```

———-

— **axiom.bib** —

```
@misc{Dele06,
  author = "Delenclos, Jonathon and Leroy, Andr\'e",
  title = {{Noncommutative Symmetric functions and $W$-polynomials}},
  link = "\url{http://arxiv.org/pdf/math/0606614.pdf}",
  algebra = "\newline\refto{category LORER LeftOreRing}",
  abstract = "
    Let $K$, $S$, $D$ be a division ring an endomorphism and a
    $S$-derivation of $K$, respectively. In this setting we introduce
    generalized noncommutative symmetric functions and obtain Vi\'ete
    formula and decompositions of different operators. $W$-polynomials
    show up naturally, their connetions with $P$-independency. Vandermonde
    and Wronskian matrices are briefly studied. The different linear
    factorizations of $W$-polynomials are analysed. Connections between
    the existence of LLCM (least left common multiples) of monic linear
    polynomials with coefficients in a ring and the left duo property are
    established at the end of the paper.",
  paper = "Dele06.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Abramov 05]{Abra05} Abramov, S.A.; Le, H.Q.; Li, Z.
``Univariate Ore Polynomial Rings in Computer Algebra''
  link = "\url{http://www.mmrc.iss.ac.cn/~zmli/papers/oretools.pdf}",
  abstract = "
    We present some algorithms related to rings of Ore polynomials (or,
    briefly, Ore rings) and describe a computer algebra library for basic
    operations in an arbitrary Ore ring. The library can be used as a
    basis for various algorithms in Ore rings, in particular, in
    differential, shift, and $q$-shift rings.",
```

```
  paper = "Abra05.pdf"
```

———-

1.28 Number Theory

— **axiom.bib** —

```
@mastersthesis{Bohl08,
  author = "Bohler, Per Reidar",
  title = {{Special number field sieve}},
  school = "Norwegian University of Science and Technology",
  year = "2008",
  link = "\url{http://www.diva-portal.org/smash/get/diva2:348611/FULLTEXT01.pdf}",
  abstract =
    "Integer factorization is a problem not yet solved for arbitrary integers.
    Huge integers are therefore widely used for encrypting, e.g. in the RSA
    encryption scheme. The special number field sieve holds the current
    factorization record for factoring the number $2^{1039}+1$. The
    algorithms depends on arithmetic in an algebraic number fields and is
    a further development from the quadratic sieve factoring algorithm.
    We therefor present the quadratic sieve first. Then the special number
    field is described. The key concepts is evaluated one bye one.
    Everything is illustrated with the corresponding parts of an example
    factorization. The running time of the special number field sieve is
    then evaluated and compared against that of the quadratic sieve. The
    special number field sieve only applies to integers of a special form,
    but a generalization has been made, the general number field sieve. It
    is slower but all estimates suggests it is asymptotically faster than
    all other existing general purpose algorithms.",
  paper = "Bohl08.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Case16,
  author = "Case, Michael",
  title = {{A Beginner's Guide to the General Number Field Sieve}},
  year = "2016",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.219.2389}",
  paper = "Case16.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Hill11,
```

```
  author = "Hill, Joshua E.",
  title = {{The Number Field Sieve: An Extended Abstract}},
  year = "2011",
  link = "\url{http://www.untruth.org/~josh/math/NFS.pdf}",
  paper = "Hill11.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt89d,
  author = "Kaltofen, E. and Valente, T. and Yui, N.",
  title = {{An improved {Las Vegas} primality test}},
  booktitle = "Proc. 1989 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "26--33",
  year = "1989",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/89/KVY89.pdf}",
  paper = "Kalt89d.pdf"
}
```

———-

— **axiom.bib** —

```
@InCollection{Kalt91b,
  author = "Kaltofen, E. and Yui, N.",
  editor = "D. V. Chudnovsky and G. V. Chudnovsky and H. Cohn and
            M. B. Nathanson",
  title = {{Explicit construction of {Hilbert} class fields of imaginary
           quadratic fields by integer lattice reduction}},
  booktitle = "Number Theory New York Seminar 1989--1990",
  pages = "150--202",
  publisher = "Springer-Verlag",
  year = "1991",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/91/KaYui91.pdf}",
  paper = "Kalt91b.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt84a,
  author = "Kaltofen, E. and Yui, N.",
  title = {{Explicit construction of the {Hilbert} class field of imaginary
           quadratic fields with class number 7 and 11}},
  booktitle = "Proc. EUROSAM '84",
  pages = "310--320",
  year = "1984",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/84/KaYui84_eurosam.ps.gz}",
```

```
  paper = "Kalt84a.ps"
}
```

———-

— **axiom.bib** —

```
@article{Pome94,
  author = "Pomerance, Carl",
  title = {{The Number Field Sieve}},
  journal = "Proc. Symposia in Applied Mathematics",
  volume = "48",
  year = "1994",
  abstract =
    "The most exciting recent development in the integer factorization
    problem is the number field sieve. It has had some spectacular successes
    with integers in certain special forms, most notably the factorization in
    1990 of the 155 decimal digit number $2^{512}+1$. For arbitrary hard
    integers, it now appears to threaten the quadratic sieve as the algorithm
    of choice. In this paper the number field sieve, and the ideas behind it,
    are described",
  paper = "Pome94.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Sho08,
  author = "Shoup, Victor",
  title = {{A Computational Introduction to Number Theory}},
  link = "\url{http://shoup.net/ntb/ntb-v2.pdf}",
  paper = "Sho08.pdf"
}
```

———-

1.29 Sparse Polynomial Interpolation

— **axiom.bib** —

```
@InProceedings{Kalt07a,
  author = "Kaltofen, Erich and Yang, Zhengfeng and Zhi, Lihong",
  title = {{On probabilistic analysis of randomization in hybrid
           symbolic-numeric algorithms}},
  year = "2007",
  booktitle = "Proc. 2007 Internat. Workshop on Symbolic-Numeric Comput.",
  pages = "11--17",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/07/KYZ07.pdf}",
  paper = "Kalt07a.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt07b,
  author = "Kaltofen, Erich and Yang, Zhengfeng",
  title = {{On Exact and Approximate Interpolation of Sparse
           Rational Functions}},
  year = "2007",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'07",
  pages = "203--210",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/07/KaYa07.pdf}",
  paper = "Kalt07b.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Gies03,
  author = "Giesbrecht, Mark and Kaltofen, Erich and Lee, Wen-shin",
  title = {{Algorithms for Computing Sparsest Shifts of Polynomials in
           Power, {Chebychev}, and {Pochhammer} Bases}},
  year = "2003",
  journal = "Journal of Symbolic Computation",
  volume = "36",
  number = "3--4",
  pages = "401--424",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/03/GKL03.pdf}",
  paper = "Gies03.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Gies02,
  author = "Giesbrecht, Mark and Kaltofen, Erich and Lee, Wen-shin",
  title = {{Algorithms for Computing the Sparsest Shifts for Polynomials via
           the Berlekamp/Massey Algorithm}},
  booktitle = "Proc. 2002 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "101--108",
  year = "2002",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/02/GKL02.pdf}",
  paper = "Gies02.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt03b,
  author = "Kaltofen, Erich and Lee, Wen-shin",
  title = {{Early Termination in Sparse Interpolation Algorithms}},
  year = "2003",
  journal = "Journal of Symbolic Computation",
  volume = "36",
  number = "3--4",
  pages = "365--400",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/03/KL03.pdf}",
  paper = "Kalt03b.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt00a,
  author = "Kaltofen, E. and Lee, W.-s. and Lobo, A.A.",
  title = {{Early termination in Ben-Or/Tiwari sparse interpolation
           and a hybrid of Zippel's algorithm}},
  booktitle = "Proc. 2000 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "192--201",
  year = "2000",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/2K/KLL2K.pdf}",
  paper = "Kalt00a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt10b,
  author = "Kaltofen, Erich L.",
  title = {{Fifteen years after {DSC} and {WLSS2} {What} parallel
           computations {I} do today [{Invited} Lecture at {PASCO} 2010]}},
  year = "2010",
  booktitle = "Proc. 2010 Internat. Workshop on Parallel Symbolic Comput.",
  pages = "10--17",
  month = "July",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/10/Ka10_pasco.pdf}",
  paper = "Kalt10b.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt90,
  author = "Kaltofen, E. and Lakshman, Y.N. and Wiley, J.M.",
```

```
  editor = "S. Watanabe and M. Nagata",
  title = {{Modular rational sparse multivariate polynomial interpolation}},
  booktitle = "Proc. 1990 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "135--139",
  publisher = "ACM Press",
  year = "1990",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/90/KLW90.pdf}",
  paper = "Kalt90.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt88a,
  author = "Kaltofen, E. and Yagati, Lakshman",
  title = {{Improved sparse multivariate polynomial interpolation algorithms}},
  booktitle = "Symbolic Algebraic Comput. Internat. Symp. ISSAC '88 Proc.",
  pages = "467--474",
  year = "1988",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/88/KaLa88.pdf}",
  paper = "Kalt88a.pdf"
}
```

———-

1.30 Divisions and Algebraic Complexity

— **axiom.bib** —

```
@InCollection{Gren11,
  author = "Grenet, Bruno and Kaltofen, Erich L. and Koiran, Pascal
            and Portier, Natacha",
  title = {{Symmetric Determinantal Representation of Formulas and Weakly
           Skew Circuits}},
  booktitle = "Randomization, Relaxation, and Complexity in Polynomial
               Equation Solving",
  year = "2011",
  editor = "Leonid Gurvits and Philippe P\'{e}bay and J. Maurice Rojas
            and David Thompson",
  pages = "61--96",
  publisher = "American Mathematical Society",
  address = "Providence, Rhode Island, USA",
  isbn = "978-0-8218-5228-6",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/10/GKKP10.pdf}",
  paper = "Gren11.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt08a,
  author = "Kaltofen, Erich and Koiran, Pascal",
  title = {{Expressing a Fraction of Two Determinants as a Determinant}},
  year = "2008",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'08",
  pages = "141--146",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/08/KaKoi08.pdf}",
  paper = "Kalt08a.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Hitz95,
  author = "Kitz, M.A. and Kaltofen, E.",
  title = {{Integer division in residue number systems}},
  journal = "IEEE Trans. Computers",
  year = "1995",
  volume = "44",
  number = "8",
  pages = "983--989",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/95/HiKa95.pdf}",
  paper = "Hitz95.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt92a,
  author = "Kaltofen, E.",
  title = {{On computing determinants of matrices without divisions}},
  booktitle = "Proc. 1992 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "342--349",
  year = "1992",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/92/Ka92_issac.pdf}",
  paper = "Kalt92a.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Cant91,
  author = "Cantor, D.G. and Kaltofen, E.",
  title = {{On fast multiplication of polynomials over arbitrary algebras}},
  journal = "Acta Inform.",
  year = "1991",
```

```
  volume = "28",
  number = "7",
  pages = "693--701",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/91/CaKa91.pdf}",
  paper = "Cant91.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt88b,
  author = "Kaltofen, E.",
  title = {{Greatest common divisors of polynomials given by
           straight-line programs}},
  journal = "J. ACM",
  year = "1988",
  volume = "35",
  number = "1",
  pages = "231--264",
  link =
    "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/88/Ka88_jacm.pdf}",
  abstract =
    "Algorithms on multivariate polynomials represented by straight-line
    programs are developed.  First, it is shown that most algebraic
    algorithms can be probabilistically applied to data that are given by
    a straight-line computation.  Testing such rational numeric data for
    zero, for instance, is facilitated by random evaluations modulo random
    prime numbers.  Then, auxiliary algorithms that determine the
    coefficients of a multivariate polynomial in a single variable are
    constructed.  The first main result is an algorithm that produces the
    greatest common divisor of the input polynomials, all in straight-line
    representation.  The second result shows how to find a straight-line
    program for the reduced numerator and denominator from one for the
    corresponding rational function.  Both the algorithm for that
    construction and the greatest common divisor algorithm are in random
    polynomial time for the usual coefftcient fields and output a
    straight-line program, which with controllably high probability
    correctly determines the requested answer.  The running times are
    polynomial functions in the binary input size, the input degrees as
    unary numbers, and the logarithm of the inverse of the failure
    probability.  The algorithm for straight-line programs for the
    numerators and denominators of rational functions implies that every
    degree-bounded rational function can be computed fast in parallel,
    that is, in polynomial size and polylogarithmic depth.",
  paper = "Kalt88b.pdf"
}
```

———-

1.31 Polynomial Factorization

— axiom.bib —

```
@article{Abbo87,
  author = "Abbott, J.A. and Bradford, R.J. and Davenport, J.H.",
  title = {{factorisation of Polynomials: Old Ideas and Recent
            Results}},
  journal = "Lecture Notes in Computer Science",
  volume = "296",
  year = "1987",
  pages = "81-91",
  abstract =
    "The problem of factorising polynomials: that is to say, given a
    polynomial with integer coefficients, to find the irreducible
    polynomials that divide it, is one with a long history. While the last
    word has not been said on the subject, we can say that the past 15
    years have seen major break-throughs, and many computer algebra
    systems now include {\sl efficient} algorithms for this problem. When
    it comes to polynomials with algebraic number coefficients, the
    problem is far harder, and several major questions remain to be
    answered. Nevertheless, the last few years have seen substantial
    improvements, and such factorisations are now possible",
  paper = "Abbo87.pdf",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@InProceedings{Bern97a,
  author = "Bernardin, Laurent and Monagan, Michael B.",
  title = {{Efficient multivariate factorization over finite fields}},
  booktitle = "Applied algebra, algebraic algorithms and error-correcting
               codes",
  series = "AAECC-12",
  year = "1997",
  location = "Toulouse, France",
  publisher = "Springer",
  pages = "15-28",
  link = "\url{http://www.cecm.sfu.ca/~monaganm/papers/AAECC.pdf}",
  abstract =
    "We describe the Maple implementation of multivariate factorization
    over general finite fields. Our first implementation is available in
    Maple V Release 3. We give selected details of the algorithms and show
    several ideas that were used to improve its efficiency. Most of the
    improvements presented here are incorporated in Maple V Release 4. In
    particular, we show that we needed a general tool for implementing
    computations in GF$(p^k)[x_1,x_2,\cdots,x_v]$.  We also needed an
    efficient implementation of our algorithms $\mathbb{Z}_p[y][x]$ in
    because any multivariate factorization may depend on several bivariate
```

```
    factorizations. The efficiency of our implementation is illustrated by
    the ability to factor bivariate polynomials with over a million
    monomials over a small prime field.",
  paper = "Bern97a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Diaz95,
  author = "Diaz, A. and Kaltofen, E.",
  title = {{On computing greatest common divisors with polynomials given by
           black boxes for their evaluation}},
  booktitle = "Proc. 1995 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "232--239",
  year = "1995",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/95/DiKa95.ps.gz}",
  paper = "Diaz95.ps"
}
```

———-

— **axiom.bib** —

```
@article{Gath01,
  author = "von zur Gathen, Joachim and Panario, Daniel",
  title = {{Factoring Polynomials Over Finite Fields: A Survey}},
  journal = "J. Symbolic Computation",
  year = "2001",
  volume = "31",
  pages = "3-17",
  link = "\url{http://people.csail.mit.edu/dmoshdov/courses/codes/poly-factorization.pdf}",
  keywords = "survey",
  abstract =
    "This survey reviews several algorithms for the factorization of
    univariate polynomials over finite fields. We emphasize the main ideas
    of the methods and provide and up-to-date bibliography of the problem.
    This paper gives algorithms for {\sl squarefree factorization},
    {\sl distinct-degree factorization}, and {\sl equal-degree factorization}.
    The first and second algorithms are deterministic, the third is
    probabilistic.",
  paper = "Gath01.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gian88,
```

```
  author = "Gianni, Patrizia. and Trager, Barry. and Zacharias, Gail",
  title = {{Groebner Bases and Primary Decomposition of Polynomial Ideals}},
  journal = "J. Symbolic Computation",
  volume = "6",
  pages = "149-167",
  year = "1988",
  link = "\url{http://www.sciencedirect.com/science/article/pii/S0747717188800403/pdf?md5=40c29b67947035884904fd
  algebra = "\newline\refto{package IDECOMP IdealDecompositionPackage}",
  paper = "Gian88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gian96,
  author = "Gianni, P. and Trager, B.",
  title = {{Square-free algorithms in positive characteristic}},
  journal =
    "J. of Applicable Algebra in Engineering, Communication and Computing",
  volume = "7",
  pages = "1-14",
  year = "1996",
}
```

———-

— **axiom.bib** —

```
@PhdThesis{Kalt82,
  author = "Kaltofen, E.",
  title = {{On the complexity of factoring polynomials with integer
           coefficients}},
  school = "RPI",
  address = "Troy, N. Y.",
  year = "1982",
  month = "December",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/82/Ka82_thesis.pdf}",
  paper = "Kalt82.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Gath85,
  author = "{von zur Gathen}, Joachim and Kaltofen, E.",
  title = {{Factoring sparse multivariate polynomials}},
  journal = "J. Comput. System Sci.",
  year = "1985",
  volume = "31",
```

```
  pages = "265--287",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/85/GaKa85_mathcomp.ps.gz}",
  paper = "Gath85.ps"
}
```

———-

— **axiom.bib** —

```
@Article{Gath85b,
  author = "{von zur Gathen}, Joachim and Kaltofen, E.",
  title = {{Polynomial-Time Factorization of Multivariate Polynomials over
           Finite Fields}},
  journal = "Math. Comput.",
  year = "1985",
  volume = "45",
  pages = "251-261",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/85/GaKa85\_mathcomp.ps.gz}",
  paper = "Gath85.ps",
  abstract =
    "We present a probabilistic algorithm that finds the irreducible
    factors of a bivariate polynomial with coefficients from a finite
    field in time polynomial in the input size, i.e. in the degree of the
    polynomial and $log$(cardinality of field). The algorithm generalizes
    to multivariate polynomials and has polynomial running time for
    densely encoded inputs. Also a deterministic version of the algorithm
    is discussed whose running time is polynomial in the degree of the
    input polynomial and the size of the field."
}
```

———-

— **axiom.bib** —

```
@InCollection{Kalt11c,
  author = "Kaltofen, Erich and Lecerf, Gr{\'e}goire",
  title = {{Section 11.5. {Factorization} of multivariate polynomials}},
  booktitle = "Handbook of Finite Fields",
  publisher = "Springer",
  pages = "382--392",
  year = "2011",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/11/KL11.pdf}",
  paper = "Kalt11c.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt05b,
  author = "Kaltofen, Erich and Koiran, Pascal",
```

```
  title = {{On the complexity of factoring bivariate supersparse
           (lacunary) polynomials}},
  year = "2005",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'05",
  pages = "208--215",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/05/KaKoi05.pdf}",
  paper = "Kalt05b.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt06a,
  author = "Kaltofen, Erich and Koiran, Pascal",
  title = {{Finding Small Degree Factors of Multivariate Supersparse
           (Lacunary) Polynomials Over Algebraic Number Fields}},
  year = "2006",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'06",
  pages = "162--168",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/06/KaKoi06.pdf}",
  paper = "Kalt06a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt97a,
  author = "Kaltofen, E. and Shoup, V.",
  title = {{Fast polynomial factorization over high algebraic extensions of
           finite fields}},
  booktitle = "Proc. 1997 Internat. Symp. Symbolic Algebraic Comput.",
  year = "1997",
  pages = "184--188",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/97/KaSh97.pdf}",
  paper = "Kalt97a.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt98,
  author = "Kaltofen, E. and Shoup, V.",
  title = {{Subquadratic-time factoring of polynomials over finite fields}},
  journal = "Math. Comput.",
  month = "July",
  year = "1998",
  volume = "67",
  number = "223",
```

```
  pages = "1179--1197",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/98/KaSh98.pdf}",
  paper = "Kalt98.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt95a,
  author = "Kaltofen, E. and Shoup, V.",
  title = {{Subquadratic-time factoring of polynomials over finite fields}},
  booktitle = "Proc. 27th Annual ACM Symp. Theory Comput.",
  year = "1995",
  publisher = "ACM Press",
  address = "New York, N.Y.",
  pages = "398--406",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/95/KaSh95.ps.gz}",
  paper = "Kalt95a.ps"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt88,
  author = "Kaltofen, E. and Trager, B.",
  title = {{Computing with polynomials given by black boxes for their
    evaluations: Greatest common divisors, factorization, separation of
    numerators and denominators}},
  booktitle = "Proc. 29th Annual Symp. Foundations of Comp. Sci.",
  pages = "296--305",
  year = "1988",
  organization = "IEEE",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/88/focs88.ps.gz}",
  paper = "Kalt88.ps"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt85b,
  author = "Kaltofen, E.",
  title = {{Computing with polynomials given by straight-line programs {II};
           sparse factorization}},
  booktitle = "Proc. 26th Annual Symp. Foundations of Comp. Sci.",
  year = "1985",
  pages = "451--458",
  organization = "IEEE",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/85/Ka85_focs.ps.gz}",
```

```
  paper = "Kalt85b.ps"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt86,
  author = "Kaltofen, E.",
  title = {{Uniform closure properties of p-computable functions}},
  booktitle = "Proc. 18th Annual ACM Symp. Theory Comput.",
  year = "1986",
  pages = "330--337",
  organization = "ACM",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/86/Ka86_stoc.pdf}",
  paper = "Kalt86.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt87b,
  author = "Kaltofen, E.",
  title = {{Single-factor Hensel lifting and its application to the
           straight-line complexity of certain polynomials}},
  booktitle = "Proc. 19th Annual ACM Symp. Theory Comput.",
  year = "1987",
  pages = "443--452",
  organization = "ACM",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/87/Ka87_stoc.pdf}",
  paper = "Kalt87b.pdf"
}
```

———-

— **axiom.bib** —

```
@InCollection{Kalt89,
  author = "Kaltofen, E.",
  editor = "S. Micali",
  title = {{Factorization of polynomials given by straight-line programs}},
  booktitle = "Randomness and Computation",
  pages = "375--412",
  publisher = "JAI Press Inc.",
  year = "1989",
  volume = "5",
  series = "Advances in Computing Research",
  address = "Greenwhich, Connecticut",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/89/Ka89_slpfac.pdf}",
  paper = "Kalt89.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@Article{Gao04,
  author = "Gao, Shuhong and Kaltofen, E. and Lauder, A.",
  title = {{Deterministic distinct degree factorization for polynomials
           over finite fields}},
  year = "2004",
  journal = "Journal of Symbolic Computation",
  volume = "38",
  number = "6",
  pages = "1461--1470",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/01/GKL01.pdf}",
  paper = "Gao04.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt87c,
  author = "Kaltofen, E.",
  title = {{Deterministic irreducibility testing of polynomials over
           large finite fields}},
  journal = "Journal of Symbolic Computation",
  year = "1987",
  volume = "4",
  pages = "77--82",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/87/Ka87_jsc.ps.gz}",
  paper = "Kalt87c.ps"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt95b,
  author = "Kaltofen, E.",
  title = {{Effective {Noether} irreducibility forms and applications}},
  journal =  "J. Comput. System Sci.",
  year = "1995",
  volume = "50",
  number = "2",
  pages = "274--295",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/95/Ka95_jcss.pdf}",
  paper = "Kalt95b.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt85a,
  author = "Kaltofen, E.",
  title = {{Fast parallel absolute irreducibility testing}},
  journal = "Journal of Symbolic Computation",
  year = "1985",
  volume = "1",
  number = "1",
  pages = "57--67",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/85/Ka85_jsc.pdf}",
  paper = "Kalt85a.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Gath85a,
  author = "{von zur Gathen}, Joachim and Kaltofen, E.",
  title = {{Factoring multivariate polynomials over finite fields}},
  journal = "Math. Comput.",
  year = "1985",
  volume = "45",
  pages = "251--261",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/85/GaKa85_mathcomp.ps.gz}",
  paper = "Gath85a.ps"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt85e,
  author = "Kaltofen, E.",
  title = {{Polynomial-time reductions from multivariate to bi- and univariate
           integral polynomial factorization}},
  journal = "{SIAM} J. Comput.",
  year = "1985",
  volume = "14",
  number = "2",
  pages = "469--489",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/85/Ka85_sicomp.pdf}",
  paper = "Kalt85e.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt82a,
  author = "Kaltofen, E.",
  title = {{A polynomial-time reduction from bivariate to univariate
           integral polynomial factorization}},
  booktitle = "Proc. 23rd Annual Symp. Foundations of Comp. Sci.",
  year = "1982",
  pages = "57--64",
  organization = "IEEE",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/82/Ka82_focs.pdf}",
  paper = "Kalt82a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt03,
  author = "Kaltofen, Erich",
  title = {{Polynomial Factorization: a Success Story}},
  year = "2003",
  booktitle = "Symbolic Algebraic Comput. Internat. Symp. ISSAC '88 Proc.",
  pages = "3--4",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/03/Ka03.pdf}",
  keywords = "survey",
  paper = "Kalt03.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt92b,
  author = "Kaltofen, E.",
  title = {{Polynomial factorization 1987-1991}},
  booktitle = "Proc. LATIN '92",
  editor = "I. Simon",
  series = "Lect. Notes Comput. Sci.",
  volume = "583",
  pages = "294--313",
  publisher = "Springer-Verlag",
  year = "1992",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/92/Ka92_latin.pdf}",
  keywords = "survey",
  paper = "Kalt92b.pdf"
}
```

———-

— **axiom.bib** —

```
@InCollection{Kalt90c,
  author = "Kaltofen, E.",
  editor = "D. V. Chudnovsky and R. D. Jenks",
  title = {{Polynomial Factorization 1982-1986}},
  booktitle = "Computers in Mathematics",
  pages = "285--309",
  publisher = "Marcel Dekker, Inc.",
  year = "1990",
  volume = "125",
  series = "Lecture Notes in Pure and Applied Mathematics",
  address = "New York, N. Y.",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/90/Ka90_survey.ps.gz}",
  keywords = "survey",
  paper = "Kalt90c.ps"
}
```

———-

— **axiom.bib** —

```
@InCollection{Kalt82b,
  author = "Kaltofen, E.",
  title = {{Polynomial factorization}},
  editor = "B. Buchberger and G. Collins and R. Loos",
  booktitle = "Computer Algebra",
  edition = "2",
  pages = "95--113",
  publisher = "Springer-Verlag",
  year = "1982",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/82/Ka82_survey.ps.gz}",
  keywords = "survey",
  paper = "Kalt82b.ps"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Sale04,
  author = "Salem, Fatima Khaled Abu",
  title = {{Factorisation Algorithms for Univariate and Bivariate Polynomials
           over Finite Fields}},
  school = "Meron College",
  year = "2004",
  link = "\url{http://www.cs.aub.edu.lb/fa21/Dissertations/My\_thesis.pdf}",
  abstract =
    "In this thesis we address algorithms for polynomial factorisation
    over finite fields. In the univariate case, we study a recent
    algorithm due to Niederreiter where the factorisation problem is
    reduced to solving a linear system over the finite field in question,
```

```
    and the solutions are used to produce the complete factorisation of
    the polynomials into irreducibles. We develop a new algorithm for
    solving the linear system using sparse Gaussian elimination with the
    Markowitz ordering strategy, and conjecture that the Niederreiter
    linear system is not only initially sparse, but also preserves its
    sparsity throughout the Gaussian elimination phase. We develop a new
    bulk synchronous parallel (BSP) algorithm base on the approach of
    Gottfert for extracting the factors of a polynomial using a basis of
    the Niederreiter solution set of $\mathbb{F}_2$. We improve upon the
    complexity and performance of the original algorithm, and produce
    binary univariate factorisations of trinomials up to degree 400000.

    We present a new approach to multivariate polynomial factorisation
    which incorporates ideas from polyhedral geometry, and generalises
    Hensel lifting. The contribution is an algorithm for factoring
    bivariate polynomials via polytopes which is able to exploit to some
    extent the sparsity of polynomials. We further show that the polytope
    method can be made sensitive to the number of nonzero terms of the
    input polynomial. We describe a sparse adaptation of the polytope
    method over finite fields of prime order which requires fewer bit
    operations and memory references for polynomials which are known to be
    the product of two sparse factors. Using this method, and to the best
    of our knowledge, we achieve a world record in binary bivariate
    factorisation of a sparse polynomial of degree 20000. We develop a BSP
    variant of the absolute irreducibility testing via polytopes given in
    [45], producing a more memory and run time efficient method that can
    provide wider ranges of applicability. We achieve absolute
    irreducibility testing of a bivariate and trivariate polynomial of
    degree 30000, and of multivariate polynomials with up to 3000
    variables.",
  paper = "Sale04.pdf"
}
```

———-

— axiom.bib —

```
@InProceedings{Shou91,
  author = "Shoup, Victor",
  title = {{A Fast Deterministic Algorithm for Factoring Polynomials over
           Finite Fields of Small Characteristic}},
  booktitle = "Proc. ISSAC 1991",
  series = "ISSAC 1991",
  year = "1991",
  pages = "14-21",
  link = "\url{http://www.shoup.net/papers/quadfactor.pdf}",
  abstract =
    "We present a new algorithm for factoring polynomials over finite
    fields. Our algorithm is deterministic, and its running time is
    ``almost'' quadratic when the characteristic is a small fixed
    prime. As such, our algorithm is asymptotically faster than previously
    known deterministic algorithms for factoring polynomials over finite
    fields of small characteristic.",
```

```
  paper = "Shou91.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Trev91,
  author = "Trevisan, Vilmar and Wang, Paul",
  title = {{Practical factorization of univariate polynomials over
           finite fields}},
  booktitle = "Proc. ISSAC 1991",
  series = "ISSAC '91",
  publisher = "ACM",
  isbn = "0-89791-437-6",
  pages = "22-31",
  year = "1991",
  link = "\url{http://lib.org/by/\_djvu\_Papers/Computer\_algebra/Algebraic\%20numbers}",
  abstract =
    "Research presented here is part of an effort to establish
    state-of-the-art factoring routines for polynomials. The foundation of
    such algorithms lies in the efficient factorization over a finite
    field $GF(p^k)$. The Cantor-Zassenhaus algorithm together with
    innovative ideas suggested by others is compared with the Berlekamp
    algorithm. The studies led us to design a hybrid algorithm that
    combine the strengths of the different approaches. The algorithms are
    also implemented and machine timings are obtained to measure the
    performance of these algorithms.",
  paper = "Trev91.djvu"
}
```

———-

1.32 Branch Cuts

— **axiom.bib** —

```
@article{Beau03,
  author = "Beaumont, James and Bradford, Russell and Davenport, James H.",
  title =
    {{Better simplification of elementary functions through power series}},
  journal = "2003 International Symposium on Symbolic and Algebraic Computation",
  series = "ISSAC'03",
  year = "2003",
  month = "August",
  abstract = "
    In [5], we introduced an algorithm for deciding whether a proposed
    simplification of elementary functions was correct in the presence of
    branch cuts. This algorithm used multivalued function simplification
    followed by verification that the branches were consistent.
```

```
    In [14] an algorithm was presented for zero-testing functions defined
    by ordinary differential equations, in terms of their power series.

    The purpose of the current paper is to investigate merging the two
    techniques. In particular, we will show an explicit reduction to the
    constant problem [16].",
  paper = "Beau03.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Brad02,
  author = "Bradford, Russell and Corless, Robert M. and Davenport, James H.
            Jeffrey, David J. and Watt, Stephen M.",
  title = {{Reasoning about the Elementary Functions of Complex Analysis}},
  journal = "Annals of Mathematics and Artificial Intelligence",
  year = "2002",
  issn = "1012-2443",
  volume = "36",
  number = "3",
  doi = "10.1023/A:1016007415899",
  link = "\url{http://dx.doi.org/10.1023/A%3A1016007415899}",
  publisher = "Kluwer Academic Publishers",
  keywords = "elementary functions; branch cuts; complex identities",
  pages = "303-318",
  abstract =
    "There are many problems with the simplification of elementary
    functions, particularly over the complex plane, though not
    exclusively. Systems tend to make ‘‘howlers’’ or not to simplify
    enough. In this paper we outline the ‘‘unwinding number’’ approach to
    such problems, and show how it can be used to prevent errors and to
    systematise such simplification, even though we have not yet reduced
    the simplification process to a complete algorithm. The unsolved
    problems are probably more amenable to the techniques of artificial
    intelligence and theorem proving than the original problem of complex
    variable analysis.",
  paper = "Brad02.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Chyz11,
  author = "Chyzak, Frederic and Davenport, James H. and
            Koutschan, Christoph and Salvy, Bruno",
  title = {{On Kahan's Rules for Determining Branch Cuts}},
  booktitle = "Proc. 13th Int. Symp. on Symbolic and Numeric Algorithms
               for Scientific Computing",
```

```
  year = "2011",
  isbn = "978-1-4673-0207-4",
  location = "Timisoara",
  pages = "47-51",
  doi = "10.1109/SYNASC.2011.51",
  acmid = "258794",
  publisher = "IEEE",
  abstract =
    "In computer algebra there are different ways of approaching the
    mathematical concept of functions, one of which is by defining them as
    solutions of differential equations. We compare different such
    appraoches and discuss the occurring problems. The main focus is on
    the question of determining possible branch cuts. We explore the
    extent to which the treatment of branch cuts can be rendered (more)
    algorithmic, by adapting Kahan's rules to the differential equation
    setting.",
  paper1 = "Chyz11a.pdf",
  paper = "Chyz11.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave10,
  author = "Davenport, James",
  title = {{The Challenges of Multivalued "Functions"}},
  journal = "Lecture Notes in Computer Science",
  volume = "6167",
  year = "2010",
  pages = "1-12",
  abstract = "
    Although, formally, mathematics is clear that a function is a
    single-valued object, mathematical practice is looser, particularly
    with n-th roots and various inverse functions. In this paper, we point
    out some of the looseness, and ask what the implications are, both for
    Artificial Intelligence and Symbolic Computation, of these practices.
    In doing so, we look at the steps necessary to convert existing tests
    into
    \begin{itemize}
    \item (a) rigorous statements
    \item (b) rigorously proved statements
    \end{itemize}
    In particular we ask whether there might be a constant ``de Bruij factor''
    [18] as we make these texts more formal, and conclude that the answer
    depends greatly on the interpretation being placed on the symbols.",
  paper = "Dave10.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Phis11,
  author = "Phisanbut, Nalina and Bradford, Russell J. and
            Davenport, James H.",
  title = {{Geometry of branch cuts}},
  journal = "ACM Communications in Computer Algebra",
  volume = "44",
  number = "3-4",
  pages = "132-135",
  year = "2011",
  abstract =
    "'Simplification' is a key concept in Computer Algebra. But many
    simplification rules, such as $\sqrt{x}\sqrt{y} \rightarrow \sqrt{xy}$
    are not universally valid, due to the fact that many elementary
    functions are multi-valued. Hence a key question is ``Is this
    simplification correct?'', which involves algorithmic analysis of the
    branch cuts involved. In this paper, we look at variable ordering and
    pre-conditioning as supporting technologies for this analysis.",
  paper = "Phis11.pdf"
}
```

———-

— axiom.bib —

```
@article{Dave12,
  author = "Davenport, James H. and Bradford, Russell and England, Matthew
            and Wilson, David",
  title = {{Program Verification in the presence of complex numbers, functions
           with branch cuts etc}},
  journal = "14th Int. Symp. on Symbolic and Numeric Algorithms for
            Scientific Computing",
  link = "\url{http://arxiv.org/pdf/1212.5417.pdf}",
  year = "2012",
  series = "SYNASC'12",
  pages = "83-88",
  publisher = "IEEE",
  abstract = "
    In considering the reliability of numerical programs, it is normal to
    ``limit our study to the semantics dealing with numerical precision''.
    On the other hand, there is a great deal of work on the reliability of
    programs that essentially ignores the numerics. The thesis of this
    paper is that there is a class of problems that fall between the two,
    which could be described as ``does the low-level arithmetic implement
    the high-level mathematics''. Many of these problems arise because
    mathematics, particularly the mathematics of the complex numbers, is
    more difficult than expected; for example the complex function log is
    not continuous, writing down a program to compute an inverse function
    is more complicated than just solving an equation, and many algebraic
    simplification rules are not universally valid.

    The good news is that these problems are theoretically capable of
    being solved, and are practically close to being solved, but not yet
    solved, in several real-world examples. However, there is still a long
```

```
    way to go before implementations match the theoretical possibilities.",
  paper = "Dave12.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Engl13,
  author = "England, M. and Bradford, R. and Davenport, J. H. and
            Wilson, D.",
  title = {{Understanding Branch Cuts of Expressions}},
  journal = "Intelligent Computer Mathematics",
  year = "2013",
  series = " LNCS 7961",
  publisher = "Springer, Berlin",
  pages = "136-151",
  isbn = "9783642393198",
  abstract = "
    We assume some standard choices for the branch cuts of a group of
    functions and consider the problem of then calculating the branch cuts
    of expressions involving those functions. Typical examples include the
    addition formulae for inverse trigonometric functions. Understanding
    these cuts is essential for working with the single-valued
    counterparts, the common approach to encoding multi-valued functions
    in computer algebra systems. While the defining choices are usually
    simple (typically portions of either the real or imaginary axes) the
    cuts induced by the expression may be surprisingly complicated. We
    have made explicit and implemented techniques for calculating the cuts
    in the computer algebra programme Maple. We discuss the issues raised,
    classifying the different cuts produced. The techniques have been
    gathered in the BranchCuts package, along with tools for visualising
    the cuts. The package is included in Maple 17 as part of the
    FunctionAdvisor tool.",
  paper = "Engl13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Jeff04,
  author = "Jeffrey, D. J. and Norman, A. C.",
  title = {{Not Seeing the Roots for the Branches: Multivalued Functions in
           Computer Algebra}},
  journal = "SIGSAM Bull.",
  issue_date = "September 2004",
  volume = "38",
  number = "3",
  month = "September",
  year = "2004",
  issn = "0163-5824",
```

```
  pages = "57--66",
  numpages = "10",
  link = "\url{http://doi.acm.org/10.1145/1040034.1040036}",
  doi = "10.1145/1040034.1040036",
  acmid = "1040036",
  publisher = "ACM",
  address = "New York, NY, USA",
  abstract = "
    We discuss the multiple definitions of multivalued functions and their
    suitability for computer algebra systems. We focus the discussion by
    taking one specific problem and considering how it is solved using
    different definitions. Our example problem is the classical one of
    calculating the roots of a cubic polynomial from the Cardano formulae,
    which contains fractional powers. We show that some definitions of
    these functions result in formulae that are correct only in the sense
    that they give candidates for solutions; these candidates must then be
    tested. Formulae that are based on single-valued functions, in
    contract, are efficient and direct.",
  paper = "Jeff04.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kaha86,
  author = "Kahan, W.",
  title = {{Branch cuts for complex elementary functions}},
  booktitle = "The State of the Art in Numerical Analysis",
  year = "1986",
  month = "April",
  editor = "Powell, M.J.D and Iserles, A.",
  publisher = "Oxford University Press",
  paper1 = "Kaha86a.pdf",
  paper = "Kaha86.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Rich96,
  author = "Rich, Albert D. and Jeffrey, David J.",
  title = {{Function Evaluation on Branch Cuts}},
  journal = "SIGSAM Bull.",
  issue_date = "June 1996",
  volume = "30",
  number = "2",
  month = "June",
  year = "1996",
  issn = "0163-5824",
  pages = "25--27",
```

```
  numpages = "3",
  link = "\url{http://doi.acm.org/10.1145/235699.235704}",
  doi = "10.1145/235699.235704",
  acmid = "235704",
  publisher = "ACM",
  address = "New York, NY, USA",
  abstract = "
    Once it is decided that a CAS will evaluate multivalued functions on
    their principal branches, questions arise concerning the branch
    definitions. The first questions concern the standardization of the
    positions of the branch cuts. These questions have largely been
    resolved between the various algebra systems and the numerical
    libraries, although not completely. In contrast to the computer
    systems, many mathematical textbooks are much further behind: for
    example, many popular textbooks still specify that the argument of a
    complex number lies between 0 and $2\pi$. We do not intend to discuss
    these first questions here, however. Once the positions of the branch
    cuts have been fixed, a second set of questions arises concerning the
    evaluation of functions on their branch cuts."
}
```

———-

— axiom.bib —

```
@article{Patt96,
  author = "Patton, Charles M.",
  title = {{A Representation of Branch-cut Information}},
  journal = "SIGSAM Bull.",
  issue_date = "June 1996",
  volume = "30",
  number = "2",
  month = "June",
  year = "1996",
  issn = "0163-5824",
  pages = "21--24",
  numpages = "4",
  link = "\url{http://doi.acm.org/10.1145/235699.235703}",
  doi = "10.1145/235699.235703",
  acmid = "235703",
  publisher = "ACM",
  address = "New York, NY, USA",
  abstract = "
    Handling (possibly) multi-valued functions is a problem in all current
    computer algebra systems. The problem is not an issue of technology.
    Its solution, however, is tied to a uniform handling of the issues by
    the mathematics community.",
  paper = "Patt96.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Squi91,
  author = "Squire, Jon S.",
  title = {{Rationale for the Proposed Standard for a Generic Package of
           Complex Elementary Functions}},
  journal = "Ada Lett.",
  issue_date = "Fall 1991",
  volume = "XI",
  number = "7",
  month = "September",
  year = "1991",
  issn = "1094-3641",
  pages = "166--179",
  numpages = "14",
  link = "\url{http://doi.acm.org/10.1145/123533.123545}",
  doi = "10.1145/123533.123545",
  acmid = "123545",
  publisher = "ACM",
  address = "New York, NY, USA",
  abstract = "
    This document provides the background on decisions that were made
    during the development of the specification for Generic Complex
    Elementary fuctions. It also rovides some information that was used to
    develop error bounds, range, domain and definitions of complex
    elementary functions.",
  paper = "Squi91.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Squi91a,
  author = "Squire, Jon S.",
  title = {{Proposed Standard for a Generic Package of Complex
           Elementary Functions}},
  journal = "Ada Lett.",
  issue_date = "Fall 1991",
  volume = "XI",
  number = "7",
  month = "September",
  year = "1991",
  issn = "1094-3641",
  pages = "140--165",
  numpages = "26",
  link = "\url{http://doi.acm.org/10.1145/123533.123544}",
  doi = "10.1145/123533.123544",
  acmid = "123544",
  publisher = "ACM",
  address = "New York, NY, USA",
  abstract = "
    This document defines the specification of a generic package of
```

```
    complex elementary functions called Generic Complex Elementary
    Functions. It does not provide the body of the package."
}
```

———-

— **axiom.bib** —

```
@misc{Unkn15,
  author = "Unknown",
  title = {{Branches of Functions}},
  link =
     "\url{http://scipp.ucsc.edu/~haber/ph116A/ComplexFunBranchTheory.pdf}",
  paper = "Unkn15.pdf"
}
```

———-

1.33 Square-free Decomposition

— **axiom.bib** —

```
@article{Bern97,
  author = "Bernardin, Laurent",
  title = {{On square-free factorization of multivariate polynomials over a
           finite field}},
  journal = "Theoretical Computer Science",
  volume = "187",
  number = "1-2",
  year = "1997",
  month = "November",
  pages = "105-116",
  abstract = "
    In this paper we present a new deterministic algorithm for computing
    the square-free decomposition of multivariate polynomials with
    coefficients from a finite field.

    Our algorithm is based on Yun's square-free factorization algorithm
    for characteristic 0. The new algorithm is more efficient than
    existing, deterministic algorithms based on Musser's squarefree
    algorithm

    We will show that the modular approach presented by Yun has no
    significant performance advantage over our algorithm. The new
    algorithm is also simpler to implement and it can rely on any existing
    GCD algorithm without having to worry about choosing ``good'' evaluation
    points.

    To demonstrate this, we present some timings using implementations in
    Maple (Char et al. 1991), where the new algorithm is used for Release
```

```
    4 onwards, and Axiom (Jenks and Sutor, 1992) which is the only system
    known to the author to use and implementation of Yun's modular
    algorithm mentioned above.",
  paper = "Bern97.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Chez07,
  author = "Ch\'eze, Guillaume and Lecerf, Gr\'egoire",
  title = {{Lifting and recombination techniques for absolute factorization}},
  journal = "Journal of Complexity",
  volume = "23",
  number = "3",
  year = "2007",
  month = "June",
  pages = "380-420",
  abstract = "
    In the vein of recent algorithmic advances in polynomial factorization
    based on lifting and recombination techniques, we present new faster
    algorithms for computing the absolute factorization of a bivariate
    polynomial. The running time of our probabilistic algorithm is less
    than quadratic in the dense size of the polynomial to be factored.",
  paper = "Chez07.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Lece07,
  author = "Lecerf, Gr\'egoire",
  title = {{Improved dense multivariate polynomial factorization algorithms}},
  journal = "Journal of Symbolic Computation",
  volume = "42",
  number = "4",
  year = "2007",
  month = "April",
  pages = "477-494",
  abstract = "
    We present new deterministic and probabilistic algorithms that reduce
    the factorization of dense polynomials from several variables to one
    variable.  The deterministic algorithm runs in sub-quadratic time in
    the dense size of the input polynomial, and the probabilistic
    algorithm is softly optimal when the number of variables is at least
    three. We also investigate the reduction from several to two variables
    and improve the quantitative versions of Bertini's irreducibility
    theorem.",
  paper = "Lece07.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Wang77,
  author = "Wang, Paul S.",
  title = {{An efficient squarefree decomposition algorithm}},
  journal = "ACM SIGSAM Bulletin",
  volume = "11",
  number = "2",
  year = "1977",
  month = "May",
  pages = "4-6",
  abstract = "
    The concept of polynomial squarefree decomposition is an important one
    in algebraic computation. The squarefree decomposition process has
    many uses in computer symbolic computation. A recent survey by D. Yun
    [3] describes many useful algorithms for this purpose. All of these
    methods depend on computing the greated common divisor (gcd) of the
    polynomial to be decomposed and its first derivative (with repect to
    some variable). In the multivariate case, this gcd computation is
    non-trivial and dominates the cost for the squarefree decompostion.",
  paper = "Wang77.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wang79,
  author = "Wang, Paul S. and Trager, Barry M.",
  title = {{New Algorithms for Polynomial Square-Free Decomposition
           over the Integers}},
  journal = "SIAM Journal on Computing",
  volume = "8",
  number = "3",
  year = "1979",
  publisher = "Society for Industrial and Applied Mathematics",
  issn = "00975397",
  abstract = "
    Previously known algorithms for polynomial square-free decomposition
    rely on greatest common divisor (gcd) computations over the same
    coefficient domain where the decomposition is to be performed. In
    particular, gcd of the given polynomial and its first derivative (with
    respect to some variable) is obtained to begin with. Application of
    modular homomorphism and $p$-adic construction (multivariate case) or
    the Chinese remainder algorithm (univariate case) results in new
    square-free decomposition algorithms which, generally speaking, take
    less time than a single gcd between the given polynomial and its first
    derivative. The key idea is to obtain one or several ‘‘correct’’
```

```
    homomorphic images of the desired square-free decomposition
    first. This provides information as to how many different square-free
    factors there are, their multiplicities and their homomorphic
    images. Since the multiplicities are known, only the square-free
    factors need to be constructed. Thus, these new algorithms are
    relatively insensitive to the multiplicities of the square-free factors.",
  paper = "Wang79.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Yun76,
  author = "Yun, D.Y.Y",
  title = {{On square-free decomposition algorithms}},
  booktitle = "Proceedings of SYMSAC'76",
  year = "1976",
  keywords = "survey",
  pages = "26-35"
}
```

———-

1.34 Symbolic Summation

— **axiom.bib** —

```
@article{Abra71,
  author = "Abramov, S.A.",
  title = {{On the summation of rational functions}},
  year = "1971",
  journal = "USSR Computational Mathematics and Mathematical Physics",
  volume = "11",
  number = "4",
  pages = "324--330",
  abstract = "
    An algorithm is given for solving the following problem: let
    $F(x_1,\ldots,x_n)$ be a rational function of the variables
    $x_i$ with rational (read or complex) coefficients; to see if
    there exists a rational function $G(v,w,x_2,\ldots,x_n)$ with
    coefficients from the same field, such that
    \[\sum_{x_1=v}^w{F(x_1,\ldots,x_n)} = G(v,w,x_2,\ldots,x_n)\]
    for all integral values of $v \le w$. If $G$ exists, to obtain it.
    Realization of the algorithm in the LISP language is discussed.",
  paper = "Abra71.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gosp78,
  author = "Gosper, R. William",
  title = {{Decision procedure for indefinite hypergeometric summation}},
  year = "1978",
  journal = "Proc. Natl. Acad. Sci. USA",
  volume = "75",
  number = "1",
  pages = "40--42",
  month = "January",
  abstract = "
    Given a summand $a_n$, we seek the ``indefinite sum'' $S(n)$
    determined (within an additive constant) by
    \[\sum_{n=1}^m{a_n} = S(m)=S(0)\]
    or, equivalently, by
    \[a_n=S(n)-S(n-1)\]
    An algorithm is exhibited which, given $a_n$, finds those $S(n)$
    with the property
    \[\displaystyle\frac{S(n)}{S(n-1)}=\textrm{a rational function of n}\]
    With this algorithm, we can determine, for example, the three
    identities
    \[\displaystyle\sum_{n=1}^m{
    \frac{\displaystyle\prod_{j=1}^{n-1}{bj^2+cj+d}}
    {\displaystyle\prod_{j=1}^n{bj^2+cj+e}}=
    \frac{1-{\displaystyle\prod_{j=1}^m{\frac{bj^2+cj+d}{bj^2+cj+e}}}}{e-d}}\]
    \[\displaystyle\sum_{n=1}^m{
    \frac{\displaystyle\prod_{j=1}^{n-1}{aj^3+bj^2+cj+d}}
         {\displaystyle\prod_{j=1}^n{aj^3+bj^2+cj+e}}=
    \frac{1-{\displaystyle\prod_{j=1}^m{
    \frac{aj^3+bj^2+cj+d}{aj^3+bj^2+cj+e}}}}{e-d}}\]
    \[\displaystyle\sum_{n=1}^m{
    \displaystyle\frac{\displaystyle\prod_{j=1}^{n-1}{bj^2+cj+d}}
    {\displaystyle\prod_{j=1}^{n+1}{bj^2+cj+e}}=
    \displaystyle\frac{
    \displaystyle\frac{2b}{e-d}-
    \displaystyle\frac{3b+c+d-e}{b+c+e}-
    \left(
    \displaystyle\frac{2b}{e-d}-\frac{b(2m+3)+c+d-e}{b(m+1)^2+c(m+1)+e}
    \right)
    \displaystyle\prod_{j=1}^m{\frac{bj^2+cj+d}{bj^2+cj+e}}}
    {b^2-c^2+d^2+e^2+2bd-2de+2eb}}\]",
  paper = "Gosp78.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Karr81,
  author = "Karr, Michael",
  title = {{Summation in Finite Terms}},
  journal = "Journal Association for Computing Machinery",
```

```
  year = "1981",
  volume = "28",
  number = "2",
  month = "April",
  issn = "0004-5411",
  pages = "305--350",
  link = "\url{http://doi.acm.org/10.1145/322248.322255}",
  publisher = "ACM",
  abstract = "
    Results which allow either the computation of symbolic solutions to
    first-order linear difference equations or the determination that
    solutions of a certain form do not exist are presented. Starting with
    a field of constants, larger fields may be constructed by the formal
    adjunction of symbols which behave like solutions to first-order
    linear equations (with a few restrictions). It is in these extension
    fields that the difference equations may be posed and in which the
    solutions are requested. The principal application of these results is
    in finding formulas for a broad class of finite sums or in showing the
    nonexistence of such formula.",
  paper = "Karr81"
}
```

———-

— **axiom.bib** —

```
@inbook{Lafo82,
  author = "Lafon, J.C.",
  title = {{Summation in Finite Terms}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "71-77",
  abstract =
    "A survey on algorithms for summation in finite terms is given. After a
    precise definition of the problem the cases of polynomial and rational
    summands are treated. The main concern of this paper is a description
    of Gosper's algorithm, which is applicable for a wide class of
    summands.  Karr's theory of extension difference fields and some
    heuristic techniques are touched on briefly.",
  paper = "Buch82.pdf",
  keywords = "axiomref, survey"
}
```

———-

— **axiom.bib** —

```
@article{Abra85,
  author = "Abramov, S.A.",
  title = {{Separation of variables in rational functions}},
```

```
  year = "1985",
  journal = "USSR Computational Mathematics and Mathematical Physics",
  volume = "25",
  number = "5",
  pages = "99--102",
  abstract =
    "The problem of expanding a rational function of several variables into
    terms with separable variables is formulated. An algorithm for solving
    this problem is given. Programs which implement this algorithm can
    occur in sets of algebraic alphabetical transformations on a computer
    and can be used to reduce the multiplicity of sums and integrals of
    rational functions for investigating differential equations with
    rational right-hand sides etc.",
  paper = "Abra85.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Karr85,
  author = "Karr, Michael",
  title = {{Theory of Summation in Finite Terms}},
  year = "1985",
  journal = "Journal of Symbolic Computation",
  volume = "1",
  number = "3",
  month = "September",
  pages = "303-315",
  abstract = "
    This paper discusses some of the mathematical aspects of an algorithm
    for finding formulas for finite sums. The results presented here
    concern a property of difference fields which show that the algorithm
    does not divide by zero, and an analogue to Liouville's theorem on
    elementary integrals.",
  paper = "Karr85.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Koep98,
  author = "Koepf, Wolfram",
  title = {{Hypergeometric Summation}},
  publisher = "Springer",
  year = "1998",
  isbn = "978-1-4471-6464-7",
  abstract = "
    Modern algorithmic techniques for summation, most of which were
    introduced in the 1990s, are developed here and carefully implemented
    in the computer algebra system Maple.
```

```
    The algorithms of Fasenmyer, Gosper, Zeilberger, Petkovsek and van
    Hoeij for hypergeometric summation and recurrence equations, efficient
    multivariate summation as well as q-analogues of the above algorithms
    are covered.  Similar algorithms concerning differential equations are
    considered. An equivalent theory of hyperexponential integration due
    to Almkvist and Zeilberger completes the book.

    The combination of these results gives orthogonal polynomials and
    (hypergeometric and q-hypergeometric) special functions a solid
    algorithmic foundation. Hence, many examples from this very active
    field are given.

    The materials covered are sutiable for an introductory course on
    algorithmic summation and will appeal to students and researchers
    alike.",
  paper = "Koep98.pdf"
}
```

———-

,

— **axiom.bib** —

```
@article{Liso93,
  author = "Lisonek, Petr and Paule, Peter and Strehl, Volker",
  title = {{Improvement of the Degree Setting in Gosper's Algorithm}},
  journal = "J. Symbolic Computation",
  volume = "16",
  year = "1993",
  pages = "243-258",
  link =
   "\url{http://www.sciencedirect.com/science/article/pii/S0747717183710436}",
  abstract =
    "A detailed study of the degree setting for Gosper's algorithm for
    indefinite hypergeometric summation is presented. In particular, we
    discriminate between rational and proper hypergeometric input. As a
    result, the critical degree bound can be improved in the former case.",
  paper = "Liso93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Manx93,
  author = "Man, Yiu-Kwong",
  title = {{On Computing Closed Forms for Indefinite Summations}},
  journal = "J. Symbolic Computation",
  volume = "16",
  pages = "335-376",
  year = "1993",
  link =
```

```
   "\url{http://www.sciencedirect.com/science/article/pii/S0747717183710539}",
  abstract =
    "A decision procedure for finding closed forms for indefinite
    summation of polynomials, rational functions, quasipolynomials and
    quasirational functions is presented. It is also extended to deal with
    some non-hypergeometric sums with rational inputs, which are not
    summable by means of Gosper's algorithm. Discussion of its
    implementation, analysis of degree bounds and some illustrative
    examples are included.",
  paper = "Manx93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Paul95,
  author = "Paule, Peter",
  title = {{Greatest Factorial Factorization and Symbolic Summation}},
  journal = "Journal of Symbolic Computation",
  year = "1995",
  volume = "20",
  pages = "235-268",
  link =
   "\url{http://www.sciencedirect.com/science/article/pii/S0747717185710498}",
  abstract =
    "The greatest factorial factorization (GFF) of a polynomial provides
    an analogue to square-free factorization but with respect to integer
    shifts instead to multiplicities. We illustrate the fundamental role
    of that concept in the context of symbolic summation. Besides a
    detailed discussion of the basic GFF notions we present a new approach
    to the indefinite rational summation problem as well as to Gosper's
    algorithm for summing hypergeometric sequences.",
  paper = "Paul95.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Petk92,
  author = "Petkovsek, Marko",
  title = {{Hypergeometric solutions of linear recurrences with
           polynomial coefficients}},
  journal = "J. Symbolic Computation",
  volume = "14",
  pages = "243-264",
  year = "1992",
  link =
   "\url{http://www.sciencedirect.com/science/article/pii/0747717192900386}",
  abstract =
    "We describe algorithm Hyper which can be used to find all
```

```
    hypergeometric solutions of linear recurrences with polynomial
    coefficients.",
  paper = "Petk92.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Schn00,
  author = "Schneider, Carsten",
  title = {{An implementation of Karr's summation algorithm in Mathematica}},
  year = "2000",
  booktitle = "S\'eminaire Lotharingien de Combinatoire",
  volume = "S43b",
  pages = "1-10",
  abstract = "
    Implementations of the celebrated Gosper algorithm (1978) for
    indefinite summation are available on almost any computer algebra
    platform. We report here about an implementation of an algorithm by
    Karr, the most general indefinite summation algorithm known. Karr's
    algorithm is, in a sense, the summation counterpart of Risch's
    algorithm for indefinite integration. This is the first implementation
    of this algorithm in a major computer algebra system. Our version
    contains new extensions to handle also definite summation problems. In
    addition we provide a feature to find automatically appropriate
    difference field extensions in which a closed form for the summation
    problem exists. These new aspects are illustrated by a variety of
    examples.",
  paper = "Schn00.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Schn01,
  author = "Schneider, Carsten",
  title = {{Symbolic Summation in Difference Fields}},
  school = "RISC Research Institute for Symbolic Computation",
  year = "2001",
  link = "\url{http://www.risc.jku.at/publications/download/risc_3017/SymbSumTHESIS.pdf}",
  abstract =
    "There are implementations of the celebrated Gosper algorithm (1978) on
    almost any computer algebra platform. Within my PhD thesis work I
    implemented Karr's Summation Algorithm (1981) based on difference
    field theory in the Mathematica system. Karr's algorithm is, in a
    sense, the summation counterpart of Risch's algorithm for indefinite
    integration.  Besides Karr's algorithm which allows us to find closed
    forms for a big clas of multisums, we developed new extensions to
    handle also definite summation problems. More precisely we are able to
    apply creative telescoping in a very general difference field setting
```

```
    and are capable of solving linear recurrences in its context.

    Besides this we find significant new insights in symbolic summation by
    rephrasing the summation problems in the general difference field
    setting. In particular, we designed algorithms for finding appropriate
    difference field extensions to solve problems in symbolic summation.
    For instance we deal with the problem to find all nested sum
    extensions which provide us with additional solutions for a given
    linear recurrence of any order. Furthermore we find appropriate sum
    extensions, if they exist, to simplify nested sums to simpler nested
    sum expressions. Moreover we are able to interpret creative
    telescoping as a special case of sum extensions in an indefinite
    summation problem. In particular we are able to determine sum
    extensions, in case of existence, to reduce the order of a recurrence
    for a definite summation problem.",
  paper = "Schn01.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Scho95,
  author = "Schorn, Markus",
  title = {{Contributions to Symbolic Summation}},
  school = "Johannes Kepler University, RISC",
  year = "1995",
  link = "\url{http://www.risc.jku.at/publications/download/risc_2246/diplom.pdf}",
  paper = "Scho95.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gerh03,
  author = "Gerhard, J. and Giesbrecht, M. and Storjohann, A. and Zima, E.V.",
  title = {{Shiftless decomposition and polynomial-time rational summation}},
  booktitle = "Proceedings of ISSAC'03",
  year = "2003",
  pages = "119--126",
  abstract = "
    New algorithms are presented for computing the dispersion set of two
    polynomials over {\bf Q} and for {\sl shiftless} factorization. Together
    with a summability criterion by Abramov, these are applied to get a
    polynomial-time algorithm for indefinite rational summation, using a
    sparse representation of the output.",
  paper = "Gerh03.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Schn05,
  author = "Schneider, Carsten",
  title = {{A new Sigma approach to multi-summation}},
  year = "2005",
  journal = "Advances in Applied Mathematics",
  volume = "34",
  number = "4",
  pages = "740--767",
  abstract = "
    We present a general algorithmic framework that allows not only to
    deal with summation problems over summands being rational expressions
    in indefinite nested syms and products (Karr, 1981), but also over
    $\delta$-finite and holonomic summand expressions that are given by a
    linear recurrence. This approach implies new computer algebra tools
    implemented in Sigma to solve multi-summation problems efficiently.
    For instacne, the extended Sigma package has been applied successively
    to provide a computer-assisted proof of Stembridge's TSPP Theorem.",
  paper = "Schn05.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kaue08a,
  author = "Kauers, Manuel and Schneider, Carsten",
  title = {{Indefinite summation with unspecified summands}},
  year = "2006",
  journal = "Discrete Mathematics",
  volume = "306",
  number = "17",
  pages = "2073--2083",
  abstract = "
    We provide a new algorithm for indefinite nested summation which is
    applicable to summands involving unspecified sequences $x(n)$. More
    than that, we show how to extend Karr's algorithm to a general
    summation framework by which additional types of summand expressions
    can be handled. Our treatment of unspecified sequences can be seen as
    a first illustrative application of this approach.",
  paper = "Kaue08a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kaue07,
  author = "Kauers, Manuel",
  title = {{Summation algorithms for Stirling number identities}},
  year = "2007",
```

```
  journal = "Journal of Symbolic Computation",
  volume = "42",
  number = "10",
  month = "October",
  pages = "948--970",
  abstract =
    "We consider a class of sequences defined by triangular recurrence
    equations.  This class contains Stirling numbers and Eulerian numbers
    of both kinds, and hypergeometric multiples of those. We give a
    sufficient criterion for sums over such sequences to obey a recurrence
    equation, and present algorithms for computing such recurrence
    equations efficiently. Our algorithms can be used for verifying many
    known summation identities on Stirling numbers instantly, and also for
    discovering new identities.",
  paper = "Kaue07.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Schn07,
  author = "Schneider, Carsten",
  title = {{Symbolic Summation Assists Combinatorics}},
  year = "2007",
  booktitle = "S\'eminaire Lotharingien de Combinatoire",
  volume = "56",
  article = "B56b",
  abstract = "
    We present symbolic summation tools in the context of difference
    fields that help scientists in practical problem solving. Throughout
    this article we present multi-sum examples which are related to
    combinatorial problems.",
  paper = "Schn07.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Schn08,
  author = "Schneider, Carsten",
  title = {{A refined difference field theory for symbolic summation}},
  year = "2008",
  journal = "Journal of Symbolic Computation",
  volume = "43",
  number = "9",
  pages = "611--644",
  abstract = "
    In this article we present a refined summation theory based on Karr's
    difference field approach. The resulting algorithms find sum
    representations with optimal nested depth. For instance, the
```

```
    algorithms have been applied successively to evaluate Feynman
    integrals from Perturbative Quantum Field Theory",
  paper = "Schn08.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Schn09,
  author = "Schneider, Carsten",
  title = {{Structural theorems for symbolic summation}},
  journal = "Proc. AAECC-2010",
  year = "2010",
  volume = "21",
  pages = "1--32",
  abstract = "
    Starting with Karr's structural theorem for summation - the discrete
    version of Liouville's structural theorem for integration - we work
    out crucial properties of the underlying difference fields. This leads
    to new and constructive structural theorems for symbolic summation.
    E.g., these results can be applied for harmonic sums which arise
    frequently in particle physics.",
  paper = "Schn09.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Eroc10,
  author = {Er\"ocal, Bur\c{c}in},
  title = {{Summation in Finite Terms Using Sage}},
  journal = "ACM Commun. Comput. Algebra",
  volume = "44",
  number = "3/4",
  month = "January",
  year = "2011",
  issn = "1932-2240",
  pages = "190--193",
  link = "\url{http://doi.acm.org/10.1145/1940475.1940517}",
  publisher = "ACM",
  abstract = "
    The summation analogue of the Risch integration algorithm developed by
    Karr uses towers of difference fields to model nested indefinite sums
    and products, as the Risch algorithm uses towers of differential
    fields to model the so called {\sl elementary functions}. The
    algorithmic machinery developed by Karr, and later generalized and
    extended, allows one to find solutions of first order difference
    equations over such towers of difference fields, in turn simplifying
    expressions involving sums and products.
```

```
    We present an implementation of this machinery in the open source
    computer algebra system Sage. Due to the nature of open source
    software, this allows direct experimentation with the algorithms and
    structures involved while taking advantage of the state of the art
    primitives provided by Sage. Even though these methods are used behind
    the scenes in the summation package Sigma and they were previously
    implemented, this is the first open source implementation.",
  paper = "Eroc10.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Eroc11,
  author = {Er\"ocal, Bur\c{c}in},
  title = {{Algebraic Extensions for Symbolic Summation}},
  school = "RISC Research Institute for Symbolic Computation",
  year = "2011",
  link = "\url{http://www.risc.jku.at/publications/download/risc_4320/erocal_thesis.pdf}",
  abstract =
    "The main result of this thesis is an effective method to extend Karr's
    symbolic summation framework to algebraic extensions. These arise, for
    example, when working with expressions involving $(-1)^n$. An
    implementation of this method, including a modernised version of
    Karr's algorithm is presented.

    Karr's algorithm is the summation analogue of the Risch algorithm for
    indefinite integration. In the summation case, towers of specialized
    difference fields called $\prod\sum$-fields are used to model nested
    sums and products. This is similar to the way elementary functions
    involving nested logarithms and exponentials are represented in
    differential fields in the integration case.

    In contrast to the integration framework, only transcendental
    extensions are allowed in Karr's construction. Algebraic extensions of
    $\prod\sum$-fields can even be rings with zero divisors. Karr's
    methods rely heavily on the ability to solve first-order linear
    difference equations and they are no longer applicable over these
    rings.

    Based on Bronstein's formulation of a method used by Singer for the
    solution of differential equations over algebraic extensions, we
    transform a first-order linear equation over an algebraic extension to
    a system of first-order equations over a purely transcendental
    extension field. However, this domain is not necessarily a
    $\prod\sum$-field. Using a structure theorem by Singer and van der
    Put, we reduce this system to a single first-order equation over a
    $\prod\sum$-field, which can be solved by Karr's algorithm. We also
    describe how to construct towers of difference ring extensions on an
    algebraic extension, where the same reduction methods can be used.

    A common bottleneck for symbolic summation algorithms is the
```

```
    computation of nullspaces of matrices over rational function
    fields. We present a fast algorithm for matrices over $\mathbb{Q}(x)$
    which uses fast arithmetic at the hardware level with calls to BLAS
    subroutines after modular reduction. This part is joint work with Arne
    Storjohann.",
  paper = "Eroc11.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Poly11,
  author = "Polyadov, S.P.",
  title = {{Indefinite summation of rational functions with factorization
           of denominators}},
  year = "2011",
  month = "November",
  journal = "Programming and Computer Software",
  volume = "37",
  number = "6",
  pages = "322--325",
  abstract = "
    A computer algebra algorithm for indefinite summation of rational
    functions based on complete factorization of denominators is
    proposed. For a given $f$, the algorithm finds two rational functions
    $g$, $r$ such that $f=g(x+1)-g(x)+r$ and the degree of the denominator
    of $r$ is minimal. A modification of the algorithm is also proposed
    that additionally minimizes the degree of the denominator of
    $g$. Computational complexity of the algorithms without regard to
    denominator factorization is shown to be $O(m^2)$, where $m$ is the
    degree of the denominator of $f$.",
  paper = "Poly11.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Schn13,
  author = "Schneider, Carsten",
  title = {{Fast Algorithms for Refined Parameterized Telescoping in
            Difference Fields}},
  journal = "CoRR",
  year = "2013",
  volume = "abs/1307.7887",
  keywords = "survey",
  abstract = "
    Parameterized telescoping (including telescoping and creative
    telescoping) and refined versions of it play a central role in the
    research area of symbolic summation. In 1981 Karr introduced
    $\prod\sum$-fields, a general class of difference fields, that enables
```

```
    one to consider this problem for indefinite nested sums and products
    covering as special cases, e.g., the (q-)hypergeometric case and their
    mixed versions. This survey article presents the available algorithms
    in the framework of $\prod\sum$-extensions and elaborates new results
    concerning efficiency.",
  paper = "Schn13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Zima13,
  author = "Zima, Eugene V.",
  title = {{Accelerating Indefinite Summation: Simple Classes of Summands}},
  journal = "Mathematics in Computer Science",
  year = "2013",
  month = "December",
  volume = "7",
  number = "4",
  pages = "455--472",
  abstract = "
    We present the history of indefinite summation starting with classics
    (Newton, Montmort, Taylor, Stirling, Euler, Boole, Jordan) followed by
    modern classics (Abramov, Gosper, Karr) to the current implementation
    in computer algebra system Maple. Along with historical presentation
    we describe several ‘‘acceleration techniques’’ of algorithms for
    indefinite summation which offer not only theoretical but also
    practical improvements in running time. Implementations of these
    algorithms in Maple are compared to standard Maple summation tools",
  paper = "Zima13.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Schn14,
  author = "Schneider, Carsten",
  title = {{A Difference Ring Theory for Symbolic Summation}},
  year = "2014",
  abstract = "
    A summation framework is developed that enhances Karr's difference
    field approach. It covers not only indefinite nested sums and products
    in terms of transcendental extensions, but it can treat, e.g., nested
    products defined over roots of unity. The theory of the so-called
    $R\prod\sum*$-extensions is supplemented by algorithms that support the
    construction of such difference rings automatically and that assist in
    the task to tackle symbolic summation problems. Algorithms are
    presented that solve parameterized telescoping equations, and more
    generally parameterized first-order difference equations, in the given
    difference ring. As a consequence, one obtains algorithms for the
```

```
    summation paradigms of telescoping and Zeilberger's creative
    telescoping. With this difference ring theory one obtains a rigorous
    summation machinery that has been applied to numerous challenging
    problems coming, e.g., from combinatorics and particle physics.",
  paper = "Schn14.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Vazq14,
  author = "Vazquez-Trejo, Javier",
  title = {{Symbolic Summation in Difference Fields}},
  year = "2014",
  school = "Carnegie-Mellon University",
  abstract = "
    We seek to understand a general method for finding a closed form for a
    given sum that acts as its antidifference in the same way that an
    integral has an antiderivative. Once an antidifference is found, then
    given the limits of the sum, it suffices to evaluate the
    antidifference at the given limits. Several algorithms (by Karr and
    Schneider) exist to find antidifferences, but the apers describing
    these algorithms leave out several of the key proofs needed to
    implement the algorithms. We attempt to fill in these gaps and find
    that many of the steps to solve difference equations rely on being
    able to solve two problems: the equivalence problem and the homogenous
    group membership problem. Solving these two problems is essential to
    finding the polynomial degree bounds and denominator bounds for
    solutions of difference equations. We study Karr and Schneider's
    treatment of these problems and elaborate on the unproven parts of
    their work. Section 1 provides background material; section 2 provides
    motivation and previous work; Section 3 provides an outline of Karr's
    Algorithm; section 4 examines the Equivalance Problem, and section 5
    examines the Homogeneous Group Membership Problem. Section 6 presents
    some proofs for the denominator and polynomial bounds used in solving
    difference equations, and Section 7 gives some directions for future
    work.",
  paper = "Vazq14.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Petk97,
  author = "Petkov\v{s}ek, Marko and Wilf, Herbert S. and
            Zeilberger, Doran",
  title = {{A=B}},
  publisher = "A.K. Peters, Ltd",
  year = "1997",
  paper = "Petk97.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Temm14,
  author = "Temme, N.M.",
  title = {{Bernoulli Polynomials Old and New}},
  abstract =
    "We consider two problems on generalized Bernoulli polynomials
    $B_n^u(z)$.  One is connected with defining functions instead of
    polynomials by making the degree $n$ of the polynomial a complex
    variable. In the second problem we are concerned with the asymptotic
    behaviour of $B_n^u(z)$ when the degree $n$ tends to infinity.",
  paper = "Temm14.pdf"
}
```

———-

1.35 Differential Forms

— **axiom.bib** —

```
@book{Cart06,
  author = {Cartan, Henri},
  title = {{Differential Forms}},
  year = "2006",
  location = {Mineola, N.Y},
  edition = {Auflage: Tra},
  isbn = {9780486450100},
  pagetotal = {166},
  publisher = {Dover Pubn Inc},
  date = {2006-05-26}
}
```

———-

— **axiom.bib** —

```
@book{Flan03a,
  author = "Flanders, Harley",
  title = {{Differential Forms with Applications to the Physical Sciences}},
  year = "2003",
  location = "Mineola, N.Y",
  isbn = "9780486661698",
  pagetotal = "240",
  publisher = "Dover Pubn Inc",
  date = "2003-03-28",
  algebra = "\newline\refto{domain DERHAM DeRhamComplex}"
```

```
}
```

———-

— **axiom.bib** —

```
@book{Whit12,
  author = {Whitney, Hassler},
  title = {{Geometric Integration Theory: Princeton Mathematical Series,
           No. 21}},
  year = "2012",
  isbn = {9781258346386},
  shorttitle = {Geometric Integration Theory},
  pagetotal = {402},
  publisher = {Literary Licensing, {LLC}},
  date = {2012-05-01}
}
```

———-

— **axiom.bib** —

```
@book{Fede13,
  author = {Federer, Herbert},
  title = {{Geometric Measure Theory}},
  year = "2013",
  location = {Berlin ; New York},
  edition = {Reprint of the 1st ed. Berlin, Heidelberg, New York 1969},
  isbn = {9783540606567},
  pagetotal = {700},
  publisher = {Springer},
  date = {2013-10-04},
  abstract =
    "This book is a major treatise in mathematics and is essential in the
    working library of the modern analyst. (Bulletin of the London
    Mathematical Society)"
}
```

———-

— **axiom.bib** —

```
@book{Abra93,
  author = "Abraham, Ralph and Marsden, Jerrold E. and Ratiu, Tudor",
  title = {{Manifolds, Tensor Analysis, and Applications}},
  year = "1993",
  location = "New York",
  edition = "2nd Corrected ed. 1988. Corr. 2nd printing 1993",
  isbn = "9780387967905",
  pagetotal = "656",
  publisher = "Springer",
```

```
  date = "1993-08-26",
  abstract = "
    The purpose of this book is to provide core material in nonlinear
    analysis for mathematicians, physicists, engineers, and mathematical
    biologists. The main goal is to provide a working knowledge of
    manifolds, dynamical systems, tensors, and differential forms. Some
    applications to Hamiltonian mechanics, fluid mechanics,
    electromagnetism, plasma dynamics and control theory are given using
    both invariant and index notation. The prerequisites required are
    solid undergraduate courses in linear algebra and advanced calculus."
}
```

———-

— axiom.bib —

```
@book{Lamb97a,
  author = {Lambe, L. A. and Radford, D. E.},
  title = {{Introduction to the Quantum Yang-Baxter Equation and
           Quantum Groups: An Algebraic Approach}},
  year = "1997",
  location = {Dordrecht ; Boston},
  edition = {Auflage: 1997},
  isbn = {9780792347217},
  shorttitle = {Introduction to the Quantum Yang-Baxter Equation and
                Quantum Groups},
  abstract = {
    Chapter 1 The algebraic prerequisites for the book are covered here
    and in the appendix. This chapter should be used as reference material
    and should be consulted as needed. A systematic treatment of algebras,
    coalgebras, bialgebras, Hopf algebras, and represen tations of these
    objects to the extent needed for the book is given. The material here
    not specifically cited can be found for the most part in [Sweedler,
    1969] in one form or another, with a few exceptions. A great deal of
    emphasis is placed on the coalgebra which is the dual of n x n
    matrices over a field. This is the most basic example of a coalgebra
    for our purposes and is at the heart of most algebraic constructions
    described in this book. We have found pointed bialgebras useful in
    connection with solving the quantum Yang-Baxter equation. For this
    reason we develop their theory in some detail. The class of examples
    described in Chapter 6 in connection with the quantum double consists
    of pointed Hopf algebras. We note the quantized enveloping algebras
    described Hopf algebras. Thus for many reasons pointed bialgebras are
    elsewhere are pointed of fundamental interest in the study of the
    quantum Yang-Baxter equation and objects quantum groups.},
  pagetotal = {300},
  publisher = {Springer},
  date = {1997-10-31}
}
```

———-

— **axiom.bib** —

```
@misc{Paga16,
  author = "Pagani, Kurt",
  title = {{SurfaceComplex}},
  year = "2016",
  link = "\url{https://groups.google.com/forum/\#!topic/fricas-devel/FRDGVFsoAKw}",
  abstract =
    "This manual describes the FriCAS domains {\bf CellMap} and
    {\bf SurfaceComplex}. These domains provide methods to compute various
    differential geometric properties of so-called $p$-surfaces in
    $\mathbb{R}^n$, a notion which is used by Walter Rudin in his famous
    {\sl Principles of Mathematical Analysis}.",
  paper = "Paga16.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Paga16a,
  author = "Pagani, Kurt",
  title = {{DifferentialGeometry1}},
  year = "2016",
  link = "\url{https://groups.google.com/forum/\#!topic/fricas-devel/FRDGVFsoAKw}",
  abstract =
    "This manual describes the FriCAS package {\bf DifferentialGeometry1}.
    This package combines differential forms and cell mappings to provide
    methods which compute {\bf pull backs}, {\bf integrals} as well as some
    other quantities of differential forms living on a surface complex.",
  paper = "Paga16a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Paga16b,
  author = "Pagani, Kurt",
  title = {{DifferentialForms}},
  year = "2016",
  link = "\url{https://groups.google.com/forum/\#!topic/fricas-devel/FRDGVFsoAKw}",
  abstract = "Reference manual for the package {\bf {\tt DifferentialForms}}",
  paper = "Paga16b.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Whee12,
  author = "Wheeler, James T.",
```

```
  title = {{Differential Forms}},
  year = "2012",
  month = "September",
  link = "\url{http://www.physics.usu.edu/Wheeler/ClassicalMechanics/CMDifferentialForms.pdf}",
  paper = "Whee12.pdf"
}
```

———-

1.36 Cylindrical Algebraic Decomposition

1.36.1 A

— **axiom.bib** —

```
@inproceedings{Anai00,
  author = "Anai, Hirokazu and Weispfenning, Volker",
  title = {{Deciding linear-trigonometric problems}},
  booktitle = "Proc ISSAC'00",
  publisher = "ACM",
  isbn = "1-58113-218-2",
  year = "2000",
  pages = "14-22",
  abstract =
    "In this paper, we present a decision procedure for certain
    linear-trigonometric problems for the reals and integers formalized in
    a suitable first-order language. The inputs are restricted to
    formulas, where all but one of the quantified variables occur linearly
    and at most one occurs both linearly and in a specific trigonometric
    function. Moreover we may allow in addition the integer-part operation
    in formulas. Besides ordinary quantifiers, we allow also counting
    quantifiers. Furthermore we also determine the qualitative structure
    of the connected components of the satisfaction set of the mixed
    linear-trigonometric variable. We also consider the decision of these
    problems in subfields of the real algebraic numbers.",
  paper = "Anai00.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Arno81,
  author = {Arnon, Dennis Soul\'e},
  title = {{Algorithms for the Geometry of Semi-algebraic Sets}},
  school = "University of Wisconsin-Madison",
  year = "1981",
  abstract =
    "Let A be a set of polynomials in r variables with integer
    coefficients. An $A$-invariant cylindrical algebraic decomposition
```

```
    (cad) of $r$-dimensional Euclidean space (G. Collins, Lect. Notes
    Comp. Sci., 33, Springer-Verlag, 1975, pp 134-183) is a certain
    cellular decomposition of $r$-space, such that each cell is a
    semi-algebraic set, the polynomials of $A$ are sign-invariant on
    each cell, and the cells are arranged into cylinders. The cad
    algorithm given by Collins provides, among other applications,
    the fastest known decision procedure for real closed fields, a
    cellular decomposition algorithm for semi-algebraic sets, and a
    method of solving nonlinear (polynomial) optimization problems
    exactly. The time-consuming calculations with real algebraic
    numbers required by the algorithm have been an obstacle to its
    implementation and use. The major contribution of this thesis
    is a new version of the cad algorithm for $r \le 3$, in which
    one works with maximal connected $A$-invariant collections of
    cells, in such a way as to often avoid the most time-consuming
    algebraic number calculations. Essential to this new cad
    algorithm is an algorithm we present for determination of
    adjacenies among the cells of a cad. Computer programs for
    the cad and adjacency algorithms have been written, providing
    the first complete implementation of a cad algorithm. Empirical
    data obtained from application of these programs are presented
    and analyzed."
}
```

———-

— axiom.bib —

```
@techreport{Arno82,
  author = "Arnon, Dennis S. and Collins, George E. and McCallum, Scott",
  title = {{Cylindrical Algebraic Decomposition I: The Basic Algorithm}},
  year = "1982",
  institution = "Purdue University",
  type = "Technical Report",
  number = "82-427A",
  link = "\url{https://pdfs.semanticscholar.org/7643/4b54250f05ebf0dcc27c33b7dc250419fb94.pdf}",
  abstract =
    "Given a set of r-variate integral polynomials, a {\sl cylindrical
    algebraic decomposition (cad)} of euclidean r-space $E^r$ into connected
    subsets compatible with the zeros of the polynomials. Collins gave a
    cad construction algorithm in 1975, as part of a quantifier elimination
    procedure for real closed fields. The algorithm has subsequently found
    diverse applications (optimization, curve display); new applications
    have been proposed (term rewriting systems, motion planning). In the
    present two-part paper, we give an algorithm for determining the pairs
    of adjacent cells in a cad of $E^2$. This capability is often needed
    in applications. In Part I we describe the essential features of the
    r-space cad algorithm, to provide a framework for the adjacency algorithm
    in Part II.",
  paper = "Arno82.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Arno82a,
  author = "Arnon, Dennis S. and McCallum, Scott",
  title = {{Cylindrical Algebraic Decomposition by Quantifier Eliminations}},
  journal = "Lecture Notes in Computer Science",
  volume = "144",
  pages = "215-222",
  year = "1982",
  abstract =
    "Cylindrical algebraic decompositions were introduced as a major
    component of a new quantifier elimination algorithm for elementary
    algebra and geometry (G. Collins, ~973). In the present paper we turn
    the tables and show that one can use quantifier elimination for ele-
    mentary algebra and geometry to obtain a new version of the
    cylindrical algebraic decomposi- tion algorithm. A key part of our
    result is a theorem, of interest in its own right, that relates the
    multiplicities of the roots of a polynomial to their continuity.",
  paper = "Arno82a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Arno84,
  author = "Arnon, Dennis S. and Collins, George E. and McCallum, Scott",
  title = {{Cylindrical Algebraic Decomposition II: An Adjacency Algorithm
           for the Plane}},
  year = "1984",
  journal = "SIAM J. Comput.",
  volume = "13",
  number = "4",
  pages = "878-889",
  abstract =
    "Given a set of r-variate integral polynomials, a {\sl cylindrical
    algebraic decomposition (cad)} of euclidean r-space $E^r$ partitions
    $E^r$ into connected subsets compaitible with the zeros of the
    polynomials. Each subset is a {\sl cell}. Informally, two cells of
    a cad are {\sl adjacent} if they touch each other; formally, they are
    adjacent if their union is connected. In applications of cad's one
    often wishes to know the adjacent pairs of cells. Previous algorithms
    for cad construction (such as that given in Part I of this paper) have
    not actually determined them. We give here in Part II an algorithm
    which determines the pairs of adjacent cells as it constructs a cad
    of $E^2$.",
  paper = "Arno84.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Arno88,
  author = "Arnon, D.S. and Mignotte, M.",
  title = {{On Mechanical Quantifier Elimination for Elementary Algebra
           and Geometry}},
  journal = "J. Symbolic Computation",
  volume = "5",
  pages = "237-259",
  year = "1988",
  abstract = "
    We give solutions to two problems of elementary algebra and geometry:
    (1) find conditions on real numbers $p$, $q$, and $r$ so that the
    polynomial function $f(x)=x^4+px^2+qx+r$ is nonnegative for all real
    $x$ and (2) find conditions on real numbers $a$, $b$, and $c$ so that
    the ellipse $\frac{(x-e)^2}{q^2}+\frac{y^2}{b^2}-1=0$ lies inside the
    unit circle $y^2+x^2-1=0$. Our solutions are obtained by following the
    basic outline of the method of quantifier elimination by cylindrical
    algebraic decomposition (Collins, 1975), but we have developed, and
    have been considerably aided by, modified versions of certain of its
    steps. We have found three equally simple but not obviously equivalent
    solutions for the first problem, illustrating the difficulty of
    obtaining unique ``simplest'' solutions to quantifier elimination
    problems of elementary algebra and geometry.",
  paper = "Arno88.pdf"

}
```

———-

— **axiom.bib** —

```
@misc{Arno88a,
  author = "Arnon, Dennis and Buchberger, Bruno",
  title = {{Algorithms in Real Algebraic Geometry}},
  publisher = "Academic Press",
  year = "1988",
  journal = "Journal of Symbolic Computation"
}
```

———-

— **axiom.bib** —

```
@article{Arno88b,
  author = "Arnon, Dennis S. and Collins, George E. and McCallum, Scott",
  title = {{An Adjacency algorithm for cylindrical algebraic decompositions
           of three-dimensional space}},
  journal = "J. Symbolic Computation",
  volume = "5",
  number = "1-2",
  pages = "163-187",
  year = "1988",
```

```
  abstract =
    "Let $A \subset \mathbb{Z}[x_1,\ldots,x_r]$
    be a finite set. An {\sl A-invariant cylindrical
    algebraic decomposition (cad)} is a certain partition of $r$-dimenslonal
    euclidean space $E^r$ into semi-algebraic cells such that the value of
    each $A_i \in A$  has constant sign (positive, negative, or zero)
    throughout each cell. Two cells are adjacent if their union is
    connected. We give an algorithm that determines the adjacent pairs
    of cells as it constructs a cad of $E^3$. The general teehnlque
    employed for $^3$ adjacency determination is projection into $E^2$,
    followed by application of an existing $E^2$ adjacency elgorlthm
    (Arnon, Collins, McCallum, 1984). Our algorithm has the following
    properties: (1) it requires no coordinate changes, end (2) in any
    cad of $E^1$, $E^2$, or $E^3$ that it builds, the boundary of each cell
    is a (disjoint) union of lower-dlmenaionel cells.",
  paper = "Arno88b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Arno88c,
  author = "Arnon, Dennis S.",
  title = {{A bibliography of quantifier elimination for real closed fields}},
  journal = "J. of Symbolic Computation",
  volume = "5",
  number = "1-2",
  pages = "267-274",
  year = "1988",
  link =
    "\url{http://www.sciencedirect.com/science/article/pii/S0747717188800166}",
  abstract =
    "A basic collection of literature relating to algorithmic quantifier
     elimination for real closed fields is assembled",
  paper = "Arno88c.pdf"
}
```

———-

1.36.2 B

— **axiom.bib** —

```
@book{Basu06,
  author = "Basu, Saugata and Pollack, Richard and
            Roy, Marie-Francoise",
  title = {{Algorithms in Real Algebraic Geometry}},
  publisher = "Springer",
  year = "2006",
  isbn = "978-3-540-33098-1"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Beau07,
  author = "Beaumont, James C. and Bradford, Russell J. and
            Davenport, James H. and Phisanbut, Nalina",
  title = {{Testing elementary function identities using CAD}},
  journal = "Applicable Algebra in Engineering, Communication and Computing",
  year = "2007",
  volume = "18",
  number = "6",
  issn = "0938-1279",
  publisher = "Springer-Verlag",
  pages = "513-543",
  abstract = "
    One of the problems with manipulating function identities in computer
    algebra systems is that they often involve functions which are
    multivalued, whilst most users tend to work with single-valued
    functions.  The problem is that many well-known identities may no
    longer be true everywhere in the complex plane when working with their
    single-valued counterparts. Conversely, we cannot ignore them, since
    in particular contexts they may be valid. We investigate the
    practicality of a method to verify such identities by means of an
    experiment; this is based on a set of test examples which one might
    realistically meet in practice.  Essentially, the method works as
    follows. We decompose the complex plane via means of cylindrical
    algebraic decomposition into regions with respect to the branch cuts
    of the functions. We then test the identity numerically at a sample
    point in the region. The latter step is facilitated by the notion of
    the {\sl adherence} of a branch cut, which was previously introduced
    by the authors. In addition to presenting the results of the
    experiment, we explain how adherence relates to the proposal of
    {\sl signed zeros} by W. Kahan, and develop this idea further in order to
    allow us to cover previously untreatable cases. Finally, we discuss
    other ways to improve upon our general methodology as well as topics
    for future research.",
  paper = "Beau07.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Beno86,
  author = "Ben-Or, Michael and Kozen, Dexter and Reif, John",
  title = {{The complexity of elementary algebra and geometry}},
  journal = "J. Computer and System Sciences",
  volume = "32",
  number = "2",
```

```
  year = "1986",
  pages = "251-264",
  abstract =
    "The theory of real closed fields can be decided in exponential space
    or parallel exponential time. In fixed dimesnion, the theory can be
    decided in NC.",
  paper = "Beno86.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Bezh05,
  author = "Bezhanishvili, Nick and de Jongh, Dick",
  title = {{Intuitionistic Logic}},
  year = "2005",
  link = "\url{https://www.cs.le.ac.uk/people/nb118/Publications/ESSLLI'05.pdf}",
  paper = "Bezh05.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Brad14,
  author = "Bradford, Russell and Chen, Changbo and Davenport, James H. and
            England, Matthew and Maza, Marc Moreno and Wilson, David",
  title = {{Truth Table Invariant Cylindrical Algebraic Decomposition by
           Regular Chains}},
  link = "\url{https://arxiv.org/pdf/1401.6310.pdf}",
  year = "2014",
  abstract =
    "A new algorithm to compute cylindrical algebraic decompositions
    (CADs) is presented, building on two recent advances. Firstly, the
    output is truth table invariant (a TTICAD) meaning given formulae have
    constant truth value on each cell of the decomposition.  Secondly, the
    computation uses regular chains theory to first build a cylindrical
    decomposition of complex space (CCD) incrementally by polynomial.
    Significant modification of the regular chains technology wa s used to
    achieve the more sophisticated invariance criteria. Experimental
    results on an implementation in the {\tt RegularChains} Library for Maple
    verify that combining these advances gives an algorithm superior to
    its individual components and competitive with the state of the art.",
  paper = "Brad14.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Brad15,
  author = "Bradford, Russell and Davenport, James H. and England, Matthew and
            McCallum, Scott",
  title = {{Truth Table Invariant Cylindrical Algebraic Decomposition}},
  link = "\url{https://arxiv.org/pdf/1401.0645.pdf}",
  year = "2015",
  abstract =
    "When using cylindrical algebraic decomposition (CAD) to solve a
    problem with respect to a set of polynomials, it is likely not the
    signs of those polynomials that are of paramount importance but rather
    the truth values of certain quantifier free formulae involving
    them. This observation motivates our article and definition of a Truth
    Table Invariant CAD (TTICAD).  In ISSAC 2013 the current authors
    presented an algorithm that can efficiently and directly construct a
    TTICAD for a list of formulae in which each has an equational
    constraint. This was achieved by generalising McCallum's theory of
    reduced projection operators. In this paper we present an extended
    version of our theory which can be applied to an arbitrary list of
    formulae, achieving savings if at least one has an equational
    constraint. We also explain how the theory of reduced projection
    operators can allow for further improvements to the lifting phase of
    CAD algorithms, even in the context of a single equational constraint.
    The algorithm is implemented fully in Maple and we present both
    promising results from experimentation and a complexity analysis
    showing the benefits of our contributions.",
  paper = "Brad15.pdf"
}
```

———-

— axiom.bib —

```
@phdthesis{Brow99,
  author = "Brown, Christopher W.",
  title = {{Solution Formula Construction for Truth Invariant CADs}},
  school = "University of Delaware",
  year = "1999",
  website = "http://www.usna.edu/CS/qepcadweb/B/impl/Implementation.html",
  link = "\url{http://www.usna.edu/Users/cs/wcbrown/research/thesis.ps.gz}",
  abstract =
    "The CAD-based quantifier elimination algorithm takes a formula from
    the elementary theory of real closed fields as input, and constructs a
    CAD of the space of the formula's unquantified variables. This
    decomposition is truth invariant with respect to the input formula,
    meaning that the formula is either identically true or identically
    false in each cell of the decomposition. The method determines the
    truth of the input formula for each cell of the CAD, and then uses the
    CAD to construct a solution formula -- a quantifier free formula that
    is equivalent to the input formula. This final phase of the algorithm,
    the solution formula construction phase, is the focus of this thesis.

    An optimal solution formula construction algorithm would be {\sl
    complete} -- i.e. applicable to any truth-invariant CAD, would be {\sl
```

```
    efficient}, and would produce {\sl simple} solution formulas. Prior to
    this thesis, no method was available with even two of these three
    properties. Several algorithms are presented, all addressing problems
    related to solution formula construction. In combination, these
    provide an efficient and complete method for constructing solution
    formulas that are simple in a variety of ways.

    Algorithms presented in this thesis have been implemented using the
    SACLIB library, and integrated into QEPCAD, a SACLIB-based
    implementation of quantifier elimination by CAD. Example computations
    based on these implementations are discussed.",
  paper = "Brow99.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Brow01,
  author="Brown, Christopher W.",
  title = {{The McCallum projection, lifting, and order-invariance}},
  year="2001",
  link = "\url{http://www.usna.edu/Users/cs/wcbrown/research/MOTS2001.1.ps.gz}",
  abstract =
    "The McCallum Projection for Cylindrical Algebraic Decomposition (CAD)
    produces a smaller projection factor set than previous projections,
    however it does not always produce a sign-invariant CAD for the set of
    input polynomials. Problems may arise when a ($k+1$)-level projection
    factor vanishes identically over a k-level cell. According to
    McCallum's paper, when this happens (and $k+1$ is not the highest
    level in the CAD) we do not know whether the projection is valid,
    i.e. whether or not a sign-invariant CAD for the set of input
    polynomials will be produced when lifting is performed in the usual
    way. When the $k$-level cell in question has dimension 0, McCallum
    suggests a modification of the lifting method that will ensure the
    validity of his projection, although to my knowledge this has never
    been implemented.

    In this paper we give easily computable criteria that often allow us
    to conclude that McCallum's projection is valid even though a
    projection factor vanishes identically over a cell. We also improve on
    McCallum's modified lifting method.

    We've incorporated the ideas contained in the paper into QEPCAD, the
    most complete implementation of CAD. When McCallum's projection is
    invalid because of a projection factor not being order-invariant over a
    region on which it vanishes identically, at least a warning message
    ought to be issued. Currently, QEPCAD may print warning messages that
    are not needed, and may fail to print warning messages when they are
    needed. Our implementation in QEPCAD ensures that warning messages are
    printed when needed, and reduces the number of times warning messages
    are printed when not needed. Neither McCallum's modified lifting
    method nor our improvement of it have been implemented in QEPCAD. The
```

```
    design of the system would make implementing such a feature quite
    difficult.",
  paper = "Brow01.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Brow01a,
  author = "Brown, Christopher W.",
  title = {{Simple CAD Construction and its Applications}},
  journal = "J. Symbolic Computation",
  year = "2001",
  volume = "31",
  pages = "521-547",
  abstract =
    "This paper presents a method for the simplification of truth-invariant
    cylindrical algebraic decompositions (CADs). Examples are given that
    demonstrate the usefulness of the method in speeding up the solutoin
    formula construction phase of the CAD-based quantifier elimination
    algorithm. Applications of the method to the construction of
    truth-invariant CADs for very large quantifier-free formulas and
    quantifier elimination of non-prenex formulas are also discussed.",
  paper = "Brow01a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Brow02,
  author = "Brown, Christopher W.",
  title = {{QEPCAD B -- A program for computing with semi-algebraic sets
           using CADs}},
  year = "2002",
  abstract =
    "This report introduces QEPCAD B, a program for computing with real
    algebraic sets using cylindrical algebraic decomposition (CAD). QEPCAD
    B both extends and improves upon the QEPCAD system for quantifier
    elimination by partial cylindrical algebraic decomposition written by
    Hoon Hong in the early 1990s. This paper briefly discusses some of the
    improvements in the implementation of CAD and quantifier elimination
    vis CAD, and provides somewhat more detail on extensions to the system
    that go beyond quantifier elimination. The author is responsible for
    most of the extended features of QEPCAD B, but improvements to the
    basic CAD implementation and to the SACLIB library on which QEPCAD is
    based are the results of many people's work.",
  paper = "Brow02.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Brow11,
  author = "Brown, Christopher W.",
  title = {{Fast simplifications for Tarski formulas based on monomial
           inequalities}},
  year = "2011",
  journal = "Journal of Symbolic Computation",
  volume = "47",
  pages = "859-882",
  abstract =
    "We define the ``combinatorial part'' of a Tarski formula in which
    equalities and inequalities are in factored or partially factored
    form.  The combinatorial part of a formula contains only ``monomial
    inequalities'', which are sign conditions on monomials. We give
    efficient algorithms for answering some basic questions about
    conjunctions of monomial inequalities and prove the
    NP-Completness/Hardness of some others. By simplifying the
    combinatorial part back to a Tarski formula, we obtain non-trivial
    simplifications without algebraic operations.",
  paper = "Brow11.pdf"
}
```

———-

1.36.3 C

— **axiom.bib** —

```
@inproceedings{Cann87,
  author = "Canny, John",
  title = {{A new algebraic method of robot motion planning and
            real geometry}},
  booktitle = "IEEE Symp. on Foundations of Comp. Sci.",
  pages = "39-48",
  year = "1987",
  abstract =
    "We present an algorithm which solves the findpath or generalized
    movers' problem in single exponential sequential time. This is the
    first algorithm for the problem whose sequential time bound is less
    than double exponential. In fact, the combinatorial exponent of the
    algorithm is equal to the number of degrees of freedom, making it
    worst-case optimal, and equaling or improving the time bounds of many
    special purpose algorithms. The algorithm accepts a formula for a
    semi-algebraic set S describing the set of free configurations and
    produces a one-dimensional skeleton or ``roadmap'' of the set, which is
    connected within each connected component of S. Additional points may
    be linked to the roadmap in linear time. Our method draws from results
    of singularity theory, and in particular makes use of the notion of
    stratified sets as an efficient alternative to cell decomposition. We
    introduce an algebraic tool called the multivariate resultant which
    gives a necessary and sufficient condition for a system of homogeneous
```

```
    polynomials to have a solution, and show that it can be computed in
    polynomial parallel time. Among the consequences of this result are
    new methods for quantifier elimination and an improved gap theorem for
    the absolute value of roots of a system of polynomials.",
  paper = "Cann87.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Cann88,
  author = "Canny, John",
  title = {{Some algebraic and geometric computations in PSPACE}},
  booktitle = "Proc 20th ACM Symp. on the theory of computing",
  pages = "460-467",
  year = "1988",
  isbn = "0-89791-264-0",
  link = "\url{http://digitalassets.lib.berkeley.edu/techreports/ucb/text/CSD-88-439.pdf}",
  abstract =
    "We give a PSPACE algorithm for determining the signs of multivariate
    polynomials at the common zeros of a system of polynomial
    equations. One of the consequences of this result is that the
    ``Generalized Movers' Problem'' in robotics drops from EXPTIME into
    PSPACE, and is therefore PSPACE-complete by a previous hardness result
    [Rei]. We also show that the existential theory of the real numbers
    can be decided in PSPACE. Other geometric problems that also drop into
    PSPACE include the 3-d Euclidean Shortest Path Problem, and the ``2-d
    Asteroid Avoidance Problem'' described in [RS]. Our method combines the
    theorem of the primitive element from classical algebra with a
    symbolic polynomial evaluation lemma from [BKR]. A decision problem
    involving several algebraic numbers is reduced to a problem involving
    a single algebraic number or primitive element, which rationally
    generates all the given algebraic numbers.",
  paper = "Cann88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Cann93,
  author = "Canny, John",
  title = {{Improved algorithms for sign and existential quantifier
           elimination}},
  journal = "The Computer Journal",
  volume = "36",
  pages = "409-418",
  year = "1993",
  abstract =
    "Recently there has been a lot of activity in algorithms that work
    over real closed fields, and that perform such calculations as
```

```
    quantifier elimination or computing connected components of
    semi-algebraic sets. A cornerstone of this work is a symbolic sign
    determination algorithm due to Ben-Or, Kozen and Reif. In this
    paper we describe a new sign determination method based on the earlier
    algorithm, but with two advantages: (i) It is faster in the univariate
    case, and (ii) In the general case, it allows purely symbolic
    quantifier elimination in pseudo-polynomial time. By purely symbolic,
    we mean that it is possible to eliminate a quantified variable from a
    system of polynomials no matter what the coefficient values are. The
    previous methods required the coefficients to be themselves
    polynomials in other variables. Our new method allows transcendental
    functions or derivatives to appear in the coefficients.

    Another corollary of the new sign-determination algorithm is a very
    simple, practical algorithm for deciding existentially-quantified
    formulae of the theory of the reals. We present an algorithm that has
    a bit complexity of $n^{k+1}d^{O(k)}(c log n)^{1+\epsilon}$ randomized, or
    $n^{n+1}d^{O(k^2)}c(1+\epsilon)$ deterministic, for any
    $\epsilon>0$, where $n$ is the number
    of polynomial constraints in the defining formula, $k$ is the number of
    variables, $d$ is a bound on the degree, $c$ bounds the bit length of the
    coefficient. The algorithm makes no general position assumptions, and
    its constants are much smaller than other recent quantifier
    elimination methods.",
  paper = "Cann93.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Cavi98,
  author = "Caviness, B. F. and Johnson, J. R.",
  title = {{Quantifier Elimination and Cylindrical Algebraic Decomposition}},
  publisher = "Springer",
  year = "1998",
  isbn = "3-221-82794-3",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Chen10,
  author = "Chen, Changbo and Davenport, James H. and May, John P. and
            Maza, Marc Moreno and Xia, Bican and Xiao, Rong",
  title = {{Triangular Decomposition of Semi-algebraic Systems}},
  year = "2010",
  link = "\url{https://arxiv.org/pdf/1002.4784.pdf}",
  abstract =
    "Regular chains and triangular decompositions are fundamental and
```

```
    well-developed tools for describing the complex solutions of
    polynomial systems. This paper proposes adaptations of these tools
    focusing on solutions of the real analogue: semi-algebraic systems.

    We show that any such system can be decomposed into finitely many
    regular semi-algebraic systems. We propose two specifications of such
    a decomposition and present corresponding algorithms. Under some
    assumptions, one type of decomposition can be computed in singly
    exponential time w.r.t. the number of variables. We implement our
    algorithms and the experimental results illustrate their
    effectiveness.",
  paper = "Chen10.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Chen12,
  author = "Chen, Changbo and Maza, Marc Moreno",
  title = {{An Incremental Algorithm for Computing Cylindrical Algebraic
           Decompositions}},
  link = "\url{https://arxiv.org/pdf/1210.5543.pdf}",
  year = "2012",
  abstract =
    "In this paper, we propose an incremental algorithm for computing
    cylindrical al gebraic decompositions. The algorithm consists of two
    parts: computing a complex cylindrical tree and refining this complex
    tree into a cylindrical tree in real space. The incrementality comes
    from the first part of the algorithm, where a complex cylindrical tree
    is constructed by refining a previous complex cylindrical tree with a
    polynomial constraint. We have implemented our algorithm in Maple. The
    experimentation shows that the proposed algorithm outperforms existing
    ones for many examples taken from the literature",
  paper = "Chen12.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Coll71,
  author = "Collins, George E.",
  title = {{The Calculation of Multivariate Polynomial Resultants}},
  journal = "ACM SYMSAC",
  volume = "18",
  number = "4",
  year = "1971",
  pages = "515-532",
  abstract =
    "An efficient algorithm is presented for the exact calculation of
    resultants of multivariate polynomials with integer coefficients.
```

```
    The algorithm applies modular homomorphisms and the Chinese remainder
    theorem, evaluation homomorphisms and interpolation, in reducing
    the problem to resultant calculation for univariate polynomials
    over GF(p), whereupon a polynomial remainder sequence algorithm is used.

    The computing time of the algorithm is analyzed theoretically as a
    function of the degrees and coefficient sizes of its inputs. As a very
    special case, it is shown that when all degrees are equal and the
    coefficient size is fixed, its computing time is approximately
    proportional to $\lambda^{2r+1}$, where $\lambda$ is the common
    degree and $r$ is the number of variables.

    Empirically observed computing times of the algorithm are tabulated
    for a large number of examples, and other algorithms are compared.
    Potential application of the algorithm to the solution of systems of
    polynomial equations is discussed.",
  paper = "Coll71.pdf"
}
```

———-

— axiom.bib —

```
@article{Coll75,
  author = "Collins, George E.",
  title = {{Quantifier Elimination for Real Closed Fields by
           Cylindrical Algebraic Decomposition}},
  year = "1975",
  journal = "Lecture Notes in Computer Science",
  volume = "33",
  pages = "134-183",
  abstract =
    "I.  Introduction.  Tarski in 1948, published a quantifier
    elimination method for the elementary theory of real closed fields
    (which he had discoverd in 1930).  As noted by Tarski, any quantifier
    elimination method for this theory also provides a decision method,
    which enables one to decide whether any sentence of the theory is
    true or false.  Since many important and difficult mathematical
    problems can be expressed in this theory, any computationally
    feasible quantifier elimination algorithm would be of utmost
    significance.

    However, it became apparent that Tarski's method required too much
    computation to be practical except for quite trivial problems.
    Seidenberg in 1954, described another method which he thought
    would be more efficient.  A third method was published by Cohen in
    1969.  Some significant improvements of Tarski's method have been
    made by W. Boge, which are described in a thesis by Holthusen

    This paper describes a completely new method which I discoverd in
    February 1973.  This method was presented in a seminar at Stanford
    University in March 1973 and in abstract form at a symposium at
    Carnegie-Mellon University in May 1973.  In August 1974 a full
```

```
    presentation of the method was delivered at the EUROSAM 74 Conference
    in Stockholm, and a preliminary version of the present paper was
    published in the proceedings of that conference.",
  paper = "Coll75.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Coll82a,
  author = "Collins, George E.",
  title = {{Factorization in Cylindrical Algebraic Decomposition - Abstract}},
  journal = "Lecture Notes in Computer Science",
  volume = "144",
  pages = "212-214",
  year = "1982",
  paper = "Coll82a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Coll91,
  author = "Collins, George E. and Hong, Hoon",
  title = {{Partial Cylindrical Algebraic Decomposition for Quantifier
           Elimination}},
  journal = "J. Symbolic Computation",
  year = "1991",
  volume = "12",
  pages = "299-328",
  abstract =
    "The Cylindrical Algebraic Decomposition method (CAD) decomposes $R^r$
    into regions over which given polynomials have constant signs.  An
    important application of GAD is quantifier elimination in elementary
    algebra and geometry.  In this paper we present a method which
    intermingles CAD construction with truth evaluation so that parts of
    the CAD are constructed only as needed to further truth evaluation and
    aborts CAD construction as soon as no more truth evaluation is needed.
    The truth evaluation utilizes in an essential way any quantifiers
    which are present and additionally takes account of atomic formulas
    from which some variables are absent.  Preliminary observations show
    that the new method is always more efficient than the original, and
    often significantly more efficient.",
  paper = "Coll91.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Coll98,
  author = "Collins, George E.",
  title = {{Quantifier Elimination by Cylindrical Algebraic Decomposition --
           Twenty Years of Progress}},
  booktitle = "Quantifier Elimination and Cylindrical Algebraic
               Decomposition",
  isbn = "3-211-82794-3",
  year = "1998",
  pages = "8-23",
  abstract =
    "The CAD (cylindrical algebraic decomposition) method and its
    application to QE (quantifier elimination) for ERA (elementary real
    algebra) was announced by the author in 1973 at Carnegie Mellon
    University (Collins 1973b). In the twenty years since then several
    very important improvements have been made to the method which,
    together with a very large increase in available computational power,
    have made it possible to solve in seconds or minutes some interesting
    problems. In the following we survey these improvements and present
    some of these problems with their solutions."
}
```

———-

— **axiom.bib** —

```
@article{Coll02,
  author = "Collins, George E. and Johnson, Jeremy R. and Krandick, Werner",
  title = {{Interval arithmetic in cylindrical algebraic decomposition}},
  journal = "J. Symbolic Computation",
  volume = "34",
  number = "2",
  pages = "145-157",
  year = "2002",
  publisher = "Elsevier",
  abstract =
    "Cylindrical algebraic decomposition requires many very time consuming
    operations, including resultant computation, polynomial factorization,
    algebraic polynomial gcd computation and polynomial real root
    isolation. We show how the time for algebraic polynomial real root
    isolation can be greatly reduced by using interval arithmetic instead
    of exact computation. This substantially reduces the overall time for
    cylindrical algebraic decomposition.",
  paper = "Coll02.pdf"
}
```

———-

1.36.4 D

— **axiom.bib** —

```
@techreport{Dave85a,
  author = "Davenport, J.H.",
  title = {{Computer Algebra for Cylindrical Algebraic Decomposition}},
  institution = "NADA Kth Stockholm / Bath Ccomputer Science",
  link = "\url{http://staff.bath.ac.uk/masjhd/TRITA.pdf}",
  type = "technical report",
  number = "88-10",
  year = "1985",
  abstract =
    "This report describes techniques for resolving systems of polynomial
    equations and inequalities. The general technique is {\sl cylindrical
    algebraic decomposition}, which decomposes space into a number of
    regions, on each of which the equations and inequalities have the
    same sign. Most of the report is spent describing the algebraic and
    algorithmic pre-requisites (resultants, algebraic numbers, Sturm
    sequences, etc.), and then describing the method, first in two
    dimensions and then in an artitrary number of dimensions.",
  paper = "Dave85a.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Dolz97a,
  author = "Dolzmann, Andreas and Sturm, Thomas and
            Weispfenning, Volker",
  title = {{Real Quantifier Elimination in Practice}},
  type = "technical report",
  institution = "University of Passau",
  number = "MIP-9720",
  year = "1997",
  abstract =
    "We give a survey of three implemented real quantifier elimination
    methods: partial cylindrical algebraic decomposition, virtual
    substitution of test terms, and a combination of Groebner basis
    computations with multivariate real root counting. We examine the
    scope of these implementations for applications in various fields of
    science, engineering, and economics",
  paper = "Dolz97a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Dolz04,
  author = "Dolzmann, Andreas and Seidl, Andreas and Sturm, Thomas",
  title = {{Efficient projection orders for CAD}},
  booktitle = "proc ISSAC'04",
  year = "2004",
  pages = "111-118",
```

```
  publisher = "ACM",
  isbn = "1-58113-827-X",
  abstract =
    "We introduce an efficient algorithm for determining a suitable
    projection order for performing cylindrical algebraic
    decomposition. Our algorithm is motivated by a statistical analysis of
    comprehensive test set computations. This analysis introduces several
    measures on both the projection sets and the entire computation, which
    turn out to be highly correlated. The statistical data also shows that
    the orders generated by our algorithm are significantly close to optimal.",
  paper = "Dolz04.pdf"
}
```

———-

1.36.5 E

— **axiom.bib** —

```
@misc{Engl16,
  author = "England, Matthew and Davenport, James H.",
  title = {{Experience with Heuristics, Benchmarks and Standards for
           Cylindrical Algebraic Decomposition}},
  link = "\url{https://arxiv.org/pdf/1609.09269.pdf}",
  abstract =
    "n the paper which inspired the $SC^2$ project, [E. Abraham,
    Building Bridges between Symbolic Computation and Satisfiability
    Checking , Proc. ISSAC 15, pp. 16, ACM, 2015] the author identified
    the use of sophisticated heuristics as a technique that the
    Satisfiability Checking community excels in and from which it is
    likely the Symbolic Computation community could learn and prosper.  To
    start this learning process we summarise our experience with heuristic
    development for the computer algebra algorithm Cylindrical Algebraic
    Decomposition. We also propose and discuss standards and benchmarks as
    another area where Symbolic Computation could prosper from
    Satisfiability Checking expertise, noting that these have been
    identified as initial actions for the new $SC^2$ community in the CSA
    project, as described in [E. Abraham et al., SC 2 : {\sl Satisfiability
    Checking meets Symbolic Computation (Project Paper)}, Intelligent
    Computer Mathematics (LNCS 9761), pp.  2843, Springer, 2015].",
  paper = "Engl16.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Emir04,
  author = "Emiris, Ioannis Z. and Tsigaridas, Elias P.",
  title = {{Comparing real algebraic numbers of small degree}},
  booktitle = "12th annual European symposium",
```

```
  series = "ESA 2004",
  year = "2004",
  isbn = "3-540-23025-4",
  location = "Bergen, Norway",
  pages = "652-663",
  link = "\url{http://www-polsys.lip6.fr/~elias//et-esa-04.pdf}",
  algebra = "\newline\refto{domain RECLOS RealClosure}",
  abstract =
    "We study polynomials of degree up to 4 over the rationals or a
    computable real subfield. Our motivation comes from the need to
    evaluate predicates in nonlinear computational geometry efficiently
    and exactly. We show a new method to compare real algebraic numbers by
    precomputing generalized Sturm sequences, thus avoiding iterative
    methods; the method, moreover handles all degenerate cases. Our first
    contribution is the determination of rational isolating points, as
    functions of the coefficients, between any pair of real roots. Our
    second contribution is to exploit invariants and Bezoutian
    subexpressions in writing the sequences, in order to reduce bit
    complexity. The degree of the tested quantities in the input
    coefficients is optimal for degree up to 3, and for degree 4 in
    certain cases. Our methods readily apply to real solving of pairs of
    quadratic equations, and to sign determination of polynomials over
    algebraic numbers of degree up to 4. Our third contribution is an
    implementation in a new module of library SYNAPS v2.1. It improves
    significantly upon the efficiency of certain publicly available
    implementations: Rioboos approach on AXIOM, the package of
    Guibas-Karavelas-Russel, and CORE v1.6, MAPLE v9, and SYNAPS
    v2.0. Some existing limited tests had shown that it is faster than
    commercial library LEDA v4.5 for quadratic algebraic numbers.",
  paper = "Emir04.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@misc{Engl14,
  author = "England, Matthew and Wilson, David and Bradford, Russell and
            Davenport, James H.",
  title = {{Using the Regular Chains Library to build cylindrical algebraic
           decompositions by projecting and lifting}},
  link = "\url{https://arxiv.org/pdf/1405.6090.pdf}",
  year = "2014",
  abstract =
    "Cylindrical algebraic decomposition (CAD) is an important tool, both
    for quantifier elimination over the reals and a range of other
    applications. Traditionally, a CAD is built through a process of
    projection and lifting to move the problem within Euclidean spaces of
    changing dimension. Recently, an alternative approach which first
    decomposes complex space using triangular decomposition before
    refining to real space has been introduced and implemented within the
    Regular-Chains Library of Maple. We here describe a freely available
```

```
    package ProjectionCAD which utilises the routines within the
    RegularChains Library to build CADs by projection and lifting. We
    detail how the projection and lifting algorithms were modified to
    allow this, discuss the motivation and survey the functionality of the
    package.",
  paper = "Engl14.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Engl14a,
  author = "England, Matthew and Bradford, Russell and Davenport, James H. and
            Wilson, David",
  title = {{Choosing a variable ordering for truth-table invariant cylindrical
           algebraic decomposition by incremental triangular decomposition}},
  link = "\url{https://arxiv.org/pdf/1405.6094.pdf}",
  year = "2014",
  abstract =
    "Cylindrical algebraic decomposition (CAD) is a key tool for solving
    problems in real algebraic geometry and beyond. In recent years a new
    approach has been developed, where regular chains technology is used
    to first build a decomposition in complex space. We consider the
    latest variant of this which builds the complex decomposition
    incrementally by polynomial and produces CADs on whose cells a
    sequence of formulae are truth-invariant. Like all CAD algorithms the
    user must provide a variable ordering which can have a profound impact
    on the tractability of a problem. We evaluate existing heuristics to
    help with the choice for this algorithm, suggest improvements and then
    derive a new heuristic more closely aligned with the mechanics of the
    new algorithm.",
  paper = "Engl14a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Engl14b,
  author = "England, Matthew and Bradford, Russell and Chen, Changbo and
            Davenport, James H. and Maza, Marc Moreno",
  title = {{Problem formulation for truth-table invariant cylindrical
           algebraic decomposition by incremental triangular decomposition}},
  journal = "LNCS",
  volume = "8543",
  link = "\url{https://arxiv.org/pdf/1404.6371.pdf}",
  year = "2014",
  abstract =
    "Cylindrical algebraic decompositions (CADs) are a key tool for
    solving problems in real algebraic geometry and beyond.  We recently
    presented a new CAD algorithm combining two advances: truth-table
```

```
    invariance, making the CAD invariant with respect to the truth of
    logical formulae rather than the signs of polynomials; and CAD
    construction by regular chains technology, where first a complex
    decomposition is constructed by refining a tree incrementally by
    constraint. We here consider how best to formulate problems for input
    to this algorithm. We focus on a choice (not relevant for other CAD
    algorithms) about the order in which constraints are presented. We
    develop new heuristics to help make this choice and thus allow the
    best use of the algorithm in practice. We also consider other choices
    of problem formulation for CAD, as discussed in CICM 2013, revisiting
    these in the context of the new algorithm.",
  paper = "Engl14b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Engl15,
  author = "England, M. and Bradford, R. and Davenport, J. H.",
  title = {{Improving the use of equational constraints in cylindrical
           algebraic decomposition}},
  journal = "ISSAC 15",
  year = "2015",
  series = "LNCS 7961",
  publisher = "ACM",
  link = "\url{http://opus.bath.ac.uk/42451/}",
  abstract = "
    When building a cylindrical algebraic decomposition (CAD) savings can
    be made in the presence of an equational constraint (EC): an equation
    logically implied by a formula.

    The present paper is concerned with how to use multiple ECs,
    propagating those in the input throughout the projection set. We
    improve on the approach of McCallum in ISSAC 2001 by using the reduced
    projection theory to make savings in the lifting phase (both to the
    polynomials we lift with and the cells lifted over). We demonstrate
    the benefits with worked examples and a complexity analysis.",
  paper = "Engl15.pdf"
}
```

———-

1.36.6 F

— **axiom.bib** —

```
@techreport{Fitc87,
  author = "Fitchas, N. and Galligo, A. and Morgenstern, J.",
  title = {{Algorithmes repides en s\'equential et en parallele pour
           l'\'elimination de quantificateurs en g\'eom\'etrie
```

```
          \'el\'ementaire}},
  type = "technical report",
  institution = {UER de Math\'ematiques Universite de Paris VII},
  year = "1987"
}
```

———-

1.36.7 G

— **axiom.bib** —

```
@inproceedings{Gonz89,
  author = "Gonzalex, Laureano and Henri, Lombardi and Recio, Tomas and
            Roy, Marie-Francoise",
  title = {{Sturm-Habicht sequence}},
  booktitle = "Proc. ACM-SIGSAM 1989",
  year = "1989",
  pages = "136-146",
  isbn = "0-89791-325-6",
  abstract =
    "Formal computations with inequalities is a subject of general interest
    in computer algebra. In particular it is fundamental in the
    parallelisation of basic algorithms and quantifier elimination for real
    closed filed ([BKR], [HRS]).

    In $\S{}I$ we give a generalisation of Sturm theorem essentially due to
    Sylvester which is the key for formal computations with inequalities.
    Our result is an improvement of previously known results (see [BKR])
    since no hypotheses have to be made on polynomials.

    In $\S{}II$ we study the subresultant sequence. We precise some of the
    classical definitions in order to avoid some problems appearing in the
    paper by Loos ([L]) and study specialisation properties in detail.

    In $\S{}III$ we introduce the Sturm-Habicht sequence, which generalises
    Habicht's work ([H]). This new sequence obtained automatically from a
    subresultant sequence has some remarkable properties:
    \begin{itemize}
    \item it gives the same information than the Sturm sequence, and this
    information may be recovered by looking only at its principal
    coefficients
    \item it can be computed by ring operations and exact divisions only,
    in polynomial time in case of integer coefficients, eventually by
    modular methods
    \item it has goos specialisation properties
    \end{itemize}

    Finally in $\S{}IV$ we give some information about applications and
    implementation of the Sturm-Habicht sequence.",
  paper = "Gonz89.pdf"
}
```

———-

— axiom.bib —

```
@inproceedings{Gonz98,
  author = "Gonzalex-Vega, L.",
  title = {{A combinatorial algorithm solving some quantifier elimination
           problems}},
  booktitle = "Quantifier Elimination and Cylindrical Algebraic
               Decomposition",
  isbn = "3-211-82794-3",
  year = "1998",
  pages = "365-374",
}
```

———-

— axiom.bib —

```
@article{Grig88,
  author = "Grigor'ev, D. Yu. and Vorobjov, N. N.",
  title = {{Solving systems of polynomial inequalities in
            subexponential time}},
  journal = "J. Symbolic Computation",
  volume = "5",
  number = "1-2",
  pages = "37-64",
  year = "1988",
  abstract =
    "Let the polynomials $f_1,\ldots,f_k \in \mathbb{Z}[X_1,\ldots,X_n]$
    have degrees $deg(f_i)<d$ and absolute value of any coefficient of less
    than or equal to $s^M$ for all $1 \le i \le k$. We describe an algorithm
    which recognizes the existence of a real solution of the system of
    inequalities $f_1 > 0,\ldots,f_k \ge 0$. In the case of a positive
    answer the algorithm constructs a certain finite set of solutions
    (which is, in fact, a representative set for the family of components
    of connectivity of the set of all real solutions of the system). The
    algorithm runs in time polynomial in $M(kd)^{n^2}$. The previously
    known upper time bound for this problem was $(Mkd)^{2^{O(n)}}$.",
  paper = "Grig88.pdf"
}
```

———-

— axiom.bib —

```
@article{Grig88a,
  author = "Grigor'ev, D. Yu.",
  title = {{The complexity of deciding Tarski algebra}},
  journal = "J. Symbolic Computation",
```

```
  volume = "5",
  number = "1-2",
  pages = "65-108",
  year = "1988",
  abstract =
    "Let a formula of Tarski algebra contain $k$ atomic subformulas of the
    kind $(f_i \ge 0)$, $1 \le i \le k$, where the polynomials
    $f_i \in \mathbb{Z}[X_1,\ldots,X_n]$ have degrees $deg(f_i)<d$, let
    $2^M$ be an upper bound for the absolute value of every coefficient
    of the polynomials $f_i$, $1 \le i \le k$, let $a \le n$ be the number
    of quantifier alternations in the prenex form of the formula. A decision
    method for Tarski algebra is described with the running time polynomial in
    $M(kd)^{(O(n))^{4a-2}}$. Previously known decision procedures have a
    time complexity polynomial in $(Mkd)^{2^{O(n)}}$",
  paper = "Grig88a.pdf"
}
```

———-

1.36.8 H

— **axiom.bib** —

```
@inproceedings{Hein89,
  author = "Heintz, J. and Roy, M-F. and Solerno, P.",
  title = {{On the complexity of semialgebraic sets}},
  booktitle = "Proc. IFIP",
  pages = "293-298",
  year = "1989"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Hong90a,
  author = "Hong, Hoon",
  title = {{Improvements in CAD-based Quantifier Elimination}},
  school = "Ohio State University",
  year = "1990",
  abstract =
    "Many important mathematical and applied mathematical problems can be
    formulated as quantifier elimination problems (QE) in elementary
    algebra and geometry. Among several proposed QE methods, the best one
    is the CAD-based method due to G. E. Collins. The major contributions
    of this thesis are centered around improving each phase of the
    CAD-based method: the projection phase, the truth-invariant CAD
    construction phase, and the solution formula construction phase

    Improvements in the projection phase. By generalizing a lemma on which
    the proof of the original projection operation is based, we are able
```

```
    to reduce the size of the projection set significantly, and thus speed
    up the whole QE process.

    Improvements in the truth-invariant CAD construction phase. By
    intermingling the stack constructions with the truth evaluations, we
    are usually able to complete quantifier elimination with only a
    partially built CAD, resulting in significant speedup.

    Improvements in the solution formula construction phase. By
    constructing defining formulas for a collection of cells instead of
    individual cells, we are often able to produce simple solution
    formulas, without incurring the enormous cost of augmented projection.

    The new CAD-based QE algorithm, which integrates all the improvements
    above, has been implemented and tested on ten QE problems from diverse
    application areas. The overall speedups range from 2 to perhaps
    300,000 times at least

    We also implemented D. Lazard's recent improvement on the projection
    phase. Tests on the ten QE problems show additional speedups by up to
    1.8 times."
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hong90,
  author = "Hong, Hoon",
  title = {{An improvement of the projection operator in cylindrical
           algebraic decomposition}},
  booktitle = "ISSAC'90",
  publisher = "ACM",
  pages = "261-264",
  year = "1990",
  abstract =
    "The cylindrical algebraic decomposition (CAD) of Collins (1975)
    provides a potentially powerful method for solving many important
    mathematical problems, provided that the required amount of
    computation can be sufficiently reduced. An important component
    of the CAD method is the projection operation. Given a set $A$ of
    $r$-variate polynomials, the projection operation produces a
    certain set $P$ of ($r  1$)-variate polynomials such that a CAD
    of $r$-dimensional space for $A$ can be constructed from a CAD
    of ($r  1$)-dimensional space for $P$. The CAD algorithm begins
    by applying the projection operation repeatedly, beginning
    with the input polynomials, until univariate polynomials are
    obtained. This process is called the projection phase."
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hong93,
  author = "Hong, Hoon",
  title = {{Quantifier elimination for formulas constrained by
           quadratic equations}},
  booktitle = "Proc. ISSAC'93",
  year = "1993",
  pages = "264-274",
  publisher = "ACM",
  isbn = "0-89791-604-2",
  abstract =
    "An algorithm is given for constructing a quantifier free formula
    (a boolean expression of polynomial equations and inequalitier)
    equivalent to a given formula of the form:
    $(\exists x \in \mathbb{R})[a_x^2+a_1x+a_0=0 \land F]$, where $F$
    is a quantifier free formula in $x_1,\ldots,x_r,x,$ and
    $a_2,a_1,a_0$ are polynomials in $x_1,\ldots,x_r$ with real
    coefficients such that the system
    $\{a_2=0,a_1=0,a_0=0\}$ has no solution in $\mathbb{R}^r$.
    Formulas of this form frequently occur in the context of
    constraint logic programming over the real numbers. The output
    formulas are made of resultants and two variants, which we call
    {\sl trace} and {\sl slope} resultants. Both of these variant
    resultants can be expressed as determinants of certain matrices.",
  paper = "Hong93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Hong93a,
  author = "Hong, Hoon",
  title = {{Quantifier Elimination for Formulas Constrained by
           Quadratic Equations via Slope Resultants}},
  journal = "The Computer Journal",
  volume = "36",
  number = "5",
  year = "1993",
  pages = "439-449",
  abstract =
    "
    An algorithm is given for eliminating the quantifier from a formula
    $(\exists x \in \mathbb{R})[a_x^2+a_1x+a_0=0 \land F]$, where $F$
    is a quantifier free formula in $x_1,\ldots,x_r,x$ and
    $a_2,a_1,a_0$ are polynomials in $x_1,\ldots,x_r$ with real
    coefficients such that the system
    $\{a_2=0,a_1=0,a_0=0\}$ has no solution in $\mathbb{R}^r$.
    The output formulas are made of resultants and their variants,
    which we call {\sl slope} resultants. The slope resultants can be,
    like the resultants, expressed as determinants of certain matrices.
    If we allow the determinant symbol in the output the computing time
    of the algorithm is {\sl linear} in the length of the input. If not,
```

```
    the computing time is dominated by $N(n^{2r+1}\ell+n^{2r}\ell^2)$
    where $N$ is the number of polynomials in the input formula, $r$
    is the number of variables, $n$ is the maximum of the degrees for
    every variable, and $\ell$ is the maximum of the integer
    coefficient bit lengths. Experimants with implementation suggest
    that the algorithm is sufficiently efficient to be useful in practice.",
  paper = "Hong93a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Hong94,
  author = "Hong, Hoon and Stahl, V.",
  title = {{Safe start regions by fixed points and tightening}},
  journal = "Computing",
  year = "1994",
  volume = "53",
  number = "3",
  pages = "323-335",
  abstract =
    "In this paper, we present a method for finding safe starting regions
    for a given system of non-linear equations. The method is an
    improvement of the usual method which is based on the fixed point
    theorem. The improvement is obtained by enclosing the components of
    the equation system by univariante interval polynomials whose zero
    sets are found. This operation is called ``ightening''. Preliminary
    experiments show that the tightening operation usually reduces the
    number of bisections, and thus the computing time. The reduction seems
    to become more dramatic when the number of variables increases."
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hong98,
  author = "Hong, Hoon",
  title = {{Simple Solution Formula Construction in Cylindrical
           Algebraic Decomposition Based Quantifier Elimination}},
  booktitle = "Quantifier Elimination and Cylindrical Algebraic
               Decomposition",
  publisher = "Springer",
  year = "1998",
  isbn = "3-211-82794-3",
  pages = "201-219",
  comment = "also ISSAC'92 pages 177-188, 1992",
  abstract =
    "In this paper, we present several improvements to the last step
    (solution formula construction step) of Collins' cylindrical
    algebraic decomposition based quantifier elimination algorithm.
```

```
    The main improvements are
    \begin{itemize}
    \item that it does {\sl not} use the expensive augmented projection
    used by Collins' original algorithm, and
    \item that it produces {\sl simple} solution formulas by using
    three-valued logic minimization
    \end{itemize}

    For example, for the quadratic problem studied by Arnon, Mignotte,
    and Lazard, the solution formula produced by the original algorithm
    consists of 401 atomic formulas, but that by the improved algorithm
    consists of 5 atomic formulas.",
  paper = "Hong98.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hong98a,
  author = "Hong, Hoon and Sendra, J. Rafael",
  title = {{Computation of Variant Results}},
  booktitle = "Quantifier Elimination and Cylindrical Algebraic
               Decomposition",
  publisher = "Springer",
  year = "1998",
  isbn = "3-211-82794-3",
  pages = "327-337"
}
```

———-

— **axiom.bib** —

```
@misc{Hong05,
  author = "Hong, Hoon",
  title = {{Tutorial on CAD}},
  year = "2005",
  paper = "Hong05.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Hong12,
  author = "Hong, Hoon",
  title = {{Variant Quantifier Elimination}},
  journal = "J. of Symbolic Computation",
  volume = "47",
  number = "7",
```

```
  year = "2012",
  pages = "883-901",
  abstract =
    "We describe an algorithm (VQE) for a variant of the real quantifier
    elimination problem (QE). The variant problem requires the input to
    satisfy a certain {\sl extra condition},
    and allows the output to be {\sl almost}
    equivalent to the input. The motivation/rationale for studying such a
    variant QE problem is that many quantified formulas arising in
    applications do satisfy the extra conditions. Furthermore, in most
    applications, it is sufficient that the output formula is almost
    equivalent to the input formula. The main idea underlying the
    algorithm is to substitute the repeated projection step of CAD by a
    single projection without carrying out a parametric existential
    decision over the reals. We find that the algorithm can tackle
    important and challenging problems, such as numerical stability
    analysis of the widely-used MacCormacks scheme. The problem has been
    practically out of reach for standard QE algorithms in spite of many
    attempts to tackle it. However, the current implementation of VQE can
    solve it in about 12 hours. This paper extends the results reported at
    the conference ISSAC 2009.",
  paper = "Hong12.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Huan14,
  author = "Huang, Zongyan and England, Matthew and Wilson, David and
            Davenport, James H. and Paulson, Lawrence C.",
  title = {{A Comparison of Three Heuristics to Choose the Variable
            Ordering for Cylindrical Algebraic Decomposition}},
  journal = "ACM Comm. in Computer Algebra",
  volume = "48",
  number = "3",
  year = "2014",
  abstraact =
    "Cylindrical algebraic decomposition (CAD) is a key tool for problems
    in real algebraic geometry and beyond. When using CAD there is often a
    choice over the variable ordering to use, with some problems
    infeasible in one ordering but simple in another. Here we discuss a
    recent experiment comparing three heuristics for making this choice on
    thousands of examples.",
  paper = "Huan14.pdf"
}
```

———-

1.36.9 J

— **axiom.bib** —

```
@misc{Jova12,
  author = "Jovanovic, Dejan and de Moura, Leonardo",
  title = {{Solving Non-Linear Arithmetic}},
  year = "2012",
  link = "\url{http://cs.nyu.edu/~dejan/nonlinear/ijcar2012_extended.pdf}",
  comment = "see: http://cs.nyu.edu/~dejan/nonlinear/",
  abstract =
    "We present a new algorithm for deciding satisfiability of non-linear
    arithmetic constraints. The algorithm performs a Conflict-Driven
    Clause Learning (CDCL)- style search for a feasible assignment, while
    using projection operators adapted from cylindrical algebraic
    decomposition to guide the search away from the conflicting states.",
  paper = "Jova12.pdf"
}
```

———-

1.36.10 K

— **axiom.bib** —

```
@article{Kaue11,
  author = "Kauers, Manuel",
  title = {{How to use Cylindrical Algebraic Decomposition}},
  journal = {S\'emminaire Lotharingien de Combinatoire},
  volume = "65",
  year = "2011",
  comment = "Article B65a",
  abstract =
    "We take some items from a textbook on inequalities and show how to
    prove them with computer algebra using the Cylindrical Algebraic
    Decomposition algorithm.  This is an example collection for standard
    applications of this algorithm, intended as a guide for potential users.",
  paper = "Kaue11.pdf",
  keywords = "printed"
}
```

———-

1.36.11 L

— **axiom.bib** —

```
@book{LaVa06,
  author = "LaValle, Steven M.",
  title = {{Planning Algorithms}},
  year = "2006",
  publisher = "Cambridge University Press"
}
```

———-

— **axiom.bib** —

```
@article{Loos93,
  author = "Loos, Rudiger and Weispfenning, Volker",
  title = {{Applying linear quantifier elimination}},
  journal = "The Computer Journal",
  volume = "36",
  number = "5",
  year = "1993",
  abstract =
    "The linear quantifier elimination algorithm for ordered fields in
    [15] is applicable to a wide range of examples involving even
    non-linear parameters. The Skolem sets of the original algorithm
    are generalized to elimination sets whose size is linear in the
    number of atomic formulas, whereas the size of Skolem sets is
    quadratic in this number. Elimiation sets may contain non-standard
    terms which enter the computation symoblically. Many cases can be
    treated by special methods improving further the empirical computing
    times.",
  paper = "Loos93.pdf"
}
```

———-

1.36.12 M

— **axiom.bib** —

```
@article{Mahb07,
  author = "Mahboubi, Assia",
  title = {{Implementing the cylindrical algebraic decomposition within
           the Coq system}},
  journal = "Math. Struct. in Comp. Science",
  volume = "17",
  year = "2007",
  pages = "99-127",
  abstract = "The Coq system is a Curry-Howard based proof assistant.
    Therefore, it contains a full functional, strongly typed programming
    language, which can be used to enhance the system with powerful
    automation tools through the implementation of reflexive tactics.
    We present the implementation of a cylindrical algebraic decomposition
    algorithm within the Coq system, whose certification leads to a proof
    producing decision procedure for the first-order theory of real numbers.",
  paper = "Mahb07.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Mcca84,
  author = "McCallum, Scott",
  title = {{An Improved Projection Operator for Cylindrical
           Algebraic Decomposition}},
  school = "University of Wisconsin-Madison",
  year = "1984",
  comment = "Computer Sciences Technical Report \#578",
  link = "\url{ftp://ftp.cs.wisc.edu/pub/techreports/1985/TR578.pdf}",
  abstract =
    "A fundamental algorithm pertaining to the solution of polynomial
    equations in several variables is the {\sl cylindrical algebraic
    decomposition (cad)} algorithm due to G.E. Collins. Given as input
    a set $A$ of integral polynomials in $r$ variables, the cad
    algorithm produces a decomposition of the euclidean space of $r$
    dimensions into cells, such that each polynomial in $A$ is
    invariant in sign throughout each of the cells in the decomposition.

    A key component of the cad algorithm is the projection operation:
    the {\sl projection} of a set $A$ of $r$-variate polynomials is
    defined to be a certain set $P$ of $(r-1)$-variate polynomials.
    The solution set, or variety, of the polynomials in $P$ comprises
    a projection in the geometric sense of the variety of $A$. The cad
    algorithm proceeds by forming successive projections of the input
    set $A$, each projection resulting in the elimination of one
    variable.

    This thesis is concerned with a refinement to the cad algorithm,
    and to its projection operation in particular. It is shown, using
    a theorem from real algebraic geometry, that the original
    projection set that Collins used can be substantially reduced in
    size, without affecting its essential properties. The results of
    theoretical analysis and empirical observations suggest that the
    reduction in the projection set size leads to an overall decrease
    in the computing time of the cad algorithm.",
  paper = "Mcca84.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mcca88,
  author = "McCallum, Scott",
  title = {{An improved projection operation for cylindrical algebraic
           decomposition of three-dimensional space}},
  journal = "J. Symbolic Computation",
  volume = "5",
  number = "1-2",
  year = "1998",
  pages = "141-161",
  abstract =
```

```
    "A key component of the cylindrical algebraic decomposition (cad)
    algorithm of Collins (1975) is the projection operation: the
    {\sl projection} of a set $A$ of $r$-variate polynomials is defined
    to be a certain set or $(r-1)$-variate polynomials. Tile zeros of the
    polynomials in the projection comprise a ``shadow'' of the critical
    zeros of $A$. The cad algorithm proceeds by forming successive
    projections of the input set $A$, each projection resulting in the
    elimination of one variable. This paper is concerned with a refinement
    to the cad algorithm, and to its projection operation in
    particular. It is shown, using a theorem from complex analytic
    geometry, that the original projection set for trivariate polynomials
    that Collins used can be substantially reduced in size, without
    affecting its essential properties. Observations suggest that the
    reduction in the projection set size leads to a substantial decrease
    in the computing time of the cad algorithm.",
  paper = "Mcca88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mcca93,
  author = "McCallum, Scott",
  title = {{Solving Polynomial Strict Inequalities Using Cylindrical
           Algebraic Decomposition}},
  journal = "The Computer Journal",
  volume = "36",
  number = "5",
  pages = "432-438",
  year = "1993",
  abstract =
    "We consider the problem of determining the consistency over the real
    number of a system of integral polynomial strict inequalities. This
    problem has applications in geometric modelling. The cylindrical
    algebraic decomposition (cad) algorithm [2] can be used to solve this
    problem, though not very efficiently. In this paper we present a less
    powerful version of the cad algorithm which can be used to solve the
    consistency problem for conjunctions of strict inequalities, and which
    runs considerably faster than the original method applied to this
    problem. In the case that a given conjunction of strict inequalities
    is consistent, the modified cad algorithm constructs solution points
    with rational coordinates.",
  paper = "Mcca93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Mcca02,
  author = "McCallum, Scott and Collins, George E.",
```

```
  title = {{Local box adjacency algorithms for cylindrical algebraic
           decompositions}},
  journal = "J. of Symbolic Computation",
  volume = "33",
  number = "3",
  year = "2002",
  pages = "321-342",
  publisher = "Elsevier",
  abstract =
    "We describe new algorithms for determining the adjacencies between
    zero-dimensional cells and those one-dimensional cells that are
    sections (not sectors) in cylindrical algebraic decompositions
    (cad). Such adjacencies constitute a basis for determining all other
    cell adjacencies. Our new algorithms are local, being applicable to a
    specified 0D cell and the 1D cells described by specified
    polynomials. Particularly efficient algorithms are given for the 0D
    cells in spaces of dimensions two, three and four. Then an algorithm
    is given for a space of arbitrary dimension. This algorithm may on
    occasion report failure, but it can then be repeated with a modified
    isolating interval and a likelihood of success.",
  paper = "Mcca02.pdf"
}
```

———-

1.36.13 P

— **axiom.bib** —

```
@inproceedings{Pede93,
  author = "Pedersen, P. and Roy, F.-F. and Szpirglas, A.",
  title = {{Counting real zeroes in the multivariate case}},
  booktitle = "Proc. MEGA'92: Computational Algebraic Geometry",
  volume = "109",
  pages = "203-224",
  year = "1993",
  abstract =
    "In this paper we show, by generalizing Hermites theorem to the
    multivariate setting, how to count the number of real or complex
    points of a discrete algebraic set which lie within some algebraic
    constraint region. We introduce a family of quadratic forms determined
    by the algebraic constraints and defined in terms of the trace from
    the coordinate ring of the variety to the ground field, and we show
    that the rank and signature of these forms are sufficient to determine
    the number of real points lying within a constraint region. In all
    cases we count geometric points, which is to say, we count points
    without multiplicity. The theoretical results on these quadratic forms
    are more or less classical, but forgotten too, and can be found also
    in [3].

    We insist on effectivity of the computation and complexity analysis:
    we show how to calculate the trace and signature using Grbner bases,
```

```
    and we show how the information provided by the individual quadratic
    forms may be combined to determine the number of real points
    satisfying a conjunction of constraints. The complexity of the
    computation is polynomial in the dimension as a vector space of the
    quotient ring associated to the defining equations. In terms of the
    number of variables, the complexity of the computation is singly
    exponential. The algorithm is well parallelizable.

    We conclude the paper by applying our machinery to the problem of
    effectively calculating the Euler characteristic of a smooth
    hypersurface."
}
```

———-

1.36.14 R

— **axiom.bib** —

```
@article{Rats02,
  author = "Ratschan, Stefan",
  title = {{Approximate quantified constraint solving by cylindrical box
           decomposition}},
  year = "2002",
  journal = "Reliable Computing",
  volume = "8",
  number = "1",
  pages = "21-42",
  abstract =
    "This paper applies interval methods to a classical problem in
    computer algebra. Let a quantified constraint be a first-order formula
    over the real numbers. As shown by A. Tarski in the 1930's, such
    constraints, when restricted to the predicate symbols $<$, $=$ and
    function symbols $+$, $\times$, are in general solvable.
    However, the problem
    becomes undecidable, when we add function symbols like
    sin. Furthermore, all exact algorithms known up to now are too slow
    for big examples, do not provide partial information before computing
    the total result, cannot satisfactorily deal with interval constants
    in the input, and often generate huge output. As a remedy we propose
    an approximation method based on interval arithmetic. It uses a
    generalization of the notion of cylindrical decompositionas
    introduced by G. Collins. We describe an implementation of the method
    and demonstrate that, for quantified constraints without equalities,
    it can efficiently give approximate information on problems that are
    too hard for current exact methods."
}
```

———-

— **axiom.bib** —

```
@article{Rene92,
  author = "Renegar, James",
  title = {{On the computational complexity and geometry of the first-order
           theory of the reals. Part I: Introduction}},
  journal = "J. of Symbolic Computation",
  volume = "13",
  number = "3",
  year = "1992",
  pages = "255-299",
  link =
    "\url{http://www.sciencedirect.com/science/article/pii/S0747717110800033}",
  abstract =
    "This series of papers presents a complete development and complexity
    analysis of a decision method, and a quantifier elimination method,
    for the first order theory of the reals. The complexity upper bounds
    which are established are the best presently available, both for
    sequential and parallel computation, and both for the bit model of
    computation and the real number model of computation; except for the
    bounds pertaining to the sequential decision method in the bit model
    of computation, all bounds represent significant improvements over
    previously established bounds.",
  paper = "Rene92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Rene92a,
  author = "Renegar, James",
  title = {{On the computational complexity and geometry of the first-order
           theory of the reals. Part II: The general decision problem}},
  journal = "J. of Symbolic Computation",
  volume = "13",
  number = "3",
  year = "1992",
  pages = "301-327",
  link =
    "\url{http://www.sciencedirect.com/science/article/pii/S0747717110800045}",
  abstract =
    "This series of papers presents a complete development and complexity
    analysis of a decision method, and a quantifier elimination method,
    for the first order theory of the reals. The complexity upper bounds
    which are established are the best presently available, both for
    sequential and parallel computation, and both for the bit model of
    computation and the real number model of computation; except for the
    bounds pertaining to the sequential decision method in the bit model
    of computation, all bounds represent significant improvements over
    previously established bounds.",
  paper = "Rene92a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Rene92b,
  author = "Renegar, James",
  title = {{On the computational complexity and geometry of the first-order
           theory of the reals. Part III: Quantifier elimination}},
  journal = "J. of Symbolic Computation",
  volume = "13",
  number = "3",
  year = "1992",
  pages = "329-352",
  link =
    "\url{http://www.sciencedirect.com/science/article/pii/S0747717110800057}",
  abstract =
    "This series of papers presents a complete development and complexity
    analysis of a decision method, and a quantifier elimination method,
    for the first order theory of the reals. The complexity upper bounds
    which are established are the best presently available, both for
    sequential and parallel computation, and both for the bit model of
    computation and the real number model of computation; except for the
    bounds pertaining to the sequential decision method in the bit model
    of computation, all bounds represent significant improvements over
    previously established bounds.",
  paper = "Rene92b.pdf"
}
```

———-

— **axiom.bib** —

```
@InCollection{Rich98,
  author = "Richardson, Daniel",
  title = {{Local Theories and Cylindrical Decomposition}},
  booktitle = "Quantifier Elimination and Cylindrical Algebraic Decomposition",
  publisher = "Springer",
  year = "1998",
  isbn = "3-211-82794-3",
  abstract =
    "There are many interesting problems which can be expressed in the
    language of elementary algebra, or in one of its extensions, but which
    do not really depend on the coordinate system, and in which the
    variables can be restricted to an arbitrary small neighborhood of some
    point. It seems that it ought to be possible to use cylindrical
    decomposition techniques to solve such problems, taking advantage
    of their special features. This article attempts to do this, but
    many unsolved problems remain.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Riobxx,
  author = "Rioboo, Renaud",
  title = {{Cylindrical Algebraic Decomposition class notes}},
  link = "\url{http://people.bath.ac.uk/masjhd/TRITA.pdf}",
  year = "2016",
  abstract =
    "This report describes techniques for resolving systems of polynomial
    equations and inequalities. The general technique is {\sl cylindrical
    algebraic decomposition}, which decomposes space into a number of
    regions, on each of which the equations and inequalities have the same
    sign. Most of the report is spent describing the algebraic and
    algorithmic pre-requisites (resultants, algebraic numbers, Sturm
    sequences, etc.), and then describing the method, first in two
    dimensions and then in an arbitrary number of dimensions",
  paper = "Riobxx.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Riob92,
  author = "Rioboo, Renaud",
  title = {{Real algebraic closure of an ordered field,
           Implementation in Axiom}},
  booktitle = "Proc. ISSAC 92",
  series = "ISSAC 92",
  year = "1992",
  isbn = "978-3-540-75186-1",
  location = "Berkeley, California",
  pages = "206-215",
  algebra = "\newline\refto{domain RECLOS RealClosure}",
  abstract =
    "Real algebraic numbers appear in many Computer Algebra problems.  For
    instance the determination of a cylindrical algebraic decomposition
    for an euclidean space requires computing with real algebraic numbers.
    This paper describes an implementation for computations with the real
    roots of a polynomial. This process is designed to be recursively
    used, so the resulting domain of computation is the set of all real
    algebraic numbers. An implementation for the real algebraic closure
    has been done in Axiom (previously called Scratchpad).",
  paper = "Riob92.pdf",
  keywords = "axiomref",
  beebe = "Rioboo:1992:RAC"
}
```

———-

— **Rioboo:1992:RAC** —

```
@InProceedings{Rioboo:1992:RAC,
```

```
  author =       "Renaud Rioboo",
  title =        {{Real algebraic closure of an ordered field,
                 implementation in Axiom}},
  crossref =     "Wang:1992:ISS",
  pages =        "206--215",
  year =         "1992",
  bibdate =      "Tue Sep 17 06:35:39 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Real algebraic numbers appear in many computer algebra
                 problems. For instance the determination of a
                 cylindrical algebraic decomposition for an Euclidian
                 space requires computing with real algebraic numbers.
                 This paper describes an implementation for computations
                 with the real roots of a polynomial. This process is
                 designed to be recursively used, so the resulting
                 domain of computation is the set of all real algebraic
                 numbers. An implementation for the real algebraic
                 closure has been done in Axiom (previously called
                 Scratchpad).",
  acknowledgement = ack-nhfb,
  affiliation =  "LITP, Univ. Pierre et Marie Curie, Paris, France",
  classification = "C4130 (Interpolation and function approximation);
                 C6130 (Data handling techniques); C7310 (Mathematics)",
  keywords =     "Axiom; Computer algebra; Cylindrical algebraic
                 decomposition; Euclidian space; Ordered field;
                 Polynomial; Real algebraic closure",
  language =     "English",
  thesaurus =    "Polynomials; Symbol manipulation",
}
```

———-

1.36.15 S

— **axiom.bib** —

```
@article{Seid54,
  author = "Seidenberg, A.",
  title = {{A New Decision Method for Elementary Algebra}},
  journal = "Annals of Mathematics",
  volume = "60",
  number = "2",
  year = "1954",
  pages = "365-374",
  abstract =
    "A. Tarski [4] has given a decision method for elementary algebra. In
     essence this comes to giving an algorithm for deciding whether a
     given finite set of polynomial inequalities has a solution. Below we
     offer another proof of this result of Tarski. The main point of our
     proof is accomplished upon showing how to decide whether a given
     polynomial $f(x,y)$ in two variables, defined over the field
     $\mathbb{R}$
```

```
    of rational numbers, has a zero in a real-closed field $\mathbb{K}$
    containing $R^1$.
    This is done in \S{}2, but for purposes of induction it is necessary to
    consider also the case that the coefficients of $f(x,y)$ involve
    parameters; the remarks in \S{}3 will be found sufficient for this
    point. In \S{}1, the problem is reduced to a decision for equalities,
    but an induction (on the number of unknowns) could not possibly be
    carried out on equalities alone; we consider a simultaneous system
    consisting of one equality $f(x,y) = 0$ and one inequality $F(x) \ne 0$.
    Once the decision for this case is achieved, at least as in \S{}3,
    the induction is immediate.",
  paper = "Seid54.pdf"
}
```

———-

1.36.16 T

— **axiom.bib** —

```
@article{Tang18,
  author = "Tang, Min and Li, Bingyu and Zeng, Zhenbing",
  title = {{Computing Sparse GCD of Multivariate Polynomials via
            Polynomial Interpolation}},
  journal = "System Science and Complexity",
  volume = "31",
  pages = "552-568",
  year = "2018",
  abstract =
    "The problem of computing the greatest common divisor (GCD) of
    multivariate polynomials, as one of the most important tasks of
    computer algebra and symbolic computation in more general scope,
    has been studied extensively since the beginning of the
    interdisciplinary of mathematics with computer science. For many
    real applications such as digital image restoration and
    enhancement, robust control theory of nonlinear systems,
    $L_1$-norm convex optimizations in compressed sensing techniques,
    as well as algebraic decoding of Reed-Solomon and BCH codes, the
    concept of sparse GCD plays a core role where only the greatest
    common divisors with much fewer terms than the original
    polynomials are of interest due to the nature of problems or data
    structures. This paper presents two methods via multivariate
    polynomial interpolation which are based on the variation of
    Zippel's method and Ben-Or/Tiwari algorithm, respectively. To
    reduce computational complexity, probabilistic techniques and
    randomization are employed to deal with univariate GCD computation
    and univariate polynomial interpolation. The authors demonstrate
    the practical performance of our algorithms on a significant body
    of examples. The implemented experiment illustrates that our
    algorithms are efficient for a quite wide range of input.",
  paper = "Tang18.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Tars48,
  author = "Tarski, Alfred",
  title = {{A Decision Method for Elementary Algebra and Geometry}},
  year = "1948",
  link =
   "\url{https://www.rand.org/content/dam/rand/pubs/reports/2008/R109.pdf}",
  paper = "Tars48.pdf"
}
```

———-

1.36.17 W

— **axiom.bib** —

```
@article{Weis88,
  author = "Weispfenning, V.",
  title = {{The complexity of linear problems in fields}},
  journal = "J. of Symbolic Computation",
  volume = "5",
  number = "1-2",
  year = "1988",
  pages = "3-27",
  link = "\url{http://www.sciencedirect.com.proxy.library.cmu.edu/science/article/pii/S0747717188800038}",
  abstract =
    "We consider linear problems in fields, ordered fields, discretely
    valued fields (with finite residue field or residue field of
    characteristic zero) and fields with finitely many independent
    orderings and discrete valuations. Most of the fields considered will
    be of characteristic zero. Formally, linear statements about these
    structures (with parameters) are given by formulas of the respective
    first-order language, in which all bound variables occur only
    linearly. We study symbolic algorithms
    ({\sl linear elimination procedures})
    that reduce linear formulas to linear formulas of a very simple form,
    i.e. quantifier-free linear formulas, and algorithms
    ({\sl linear decision procedures})
    that decide whether a given linear sentence holds in all
    structures of the given class. For all classes of fields considered,
    we find linear elimination procedures that run in double exponential
    space and time. As a consequence, we can show that for fields (with
    one or several discrete valuations), linear statements can be
    transferred from characteristic zero to prime characteristic $p$,
    provided $p$ is double exponential in the length of the statement. (For
    similar bounds in the non-linear case, see Brown, 1978.) We find
    corresponding linear decision procedures in the Berman complexity
```

```
    classes $\bigcup_{c\in N} STA(*,2^{cn},dn)$
    for $d = 1, 2$. In particular, all these procedures run in
    exponential space. The technique employed is quantifier elimination
    via Skolem terms based on Ferrante and Rackoff (1975). Using ideas of
    Fischer and Rabin (1974), Berman (1977), Frer (1982), we establish
    lower bounds for these problems showing that our upper bounds are
    essentially tight. For linear formulas with a bounded number of
    quantifiers all our algorithms run in polynomial time. For linear
    formulas of bounded quantifier alternation most of the algorithms run
    in time $2^{O(n^k)}$ for fixed $k$.",
  paper = "Weis88.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Weis92,
  author = "Weispfenning, V.",
  title = {{Comprehensive Groebner bases}},
  journal = "J. Symbolic Computation",
  volume = "14",
  number = "1",
  year = "1992",
  pages = "1-29",
  abstract =
    "Let $K$ be an integral domain and let $S$ be the polynomial ring
    $K[U_1,\ldots,U_m; X_1,\ldots,X_n]$. For any finite $F \subseteq S$,
    we construct a comprehensive Groebner basis of the ideal $Id(F)$, a
    finite ideal basis of $Id(F)$ that is a Groebner basis of $Id(F)$
    in $K^\prime[X_1,\ldots,X_n]$ for every specialization of the
    parameters $U_1,\ldots,U_m$ in an arbitrary field $K^1$. We show
    that this construction can be performed with the same worst case
    degree bounds in the main variable $X_i$, as for ordinary Groebner
    bases; moreover, examples computed in an ALDES/SAC-2 implementation
    show that the construction is of practical value. Comprehensive
    Groebner bases admit numerous applications to parametric problems
    in algebraic geometry; in particular, they yield a fast elimination
    of quantifier-blocks in algebraically closed fields",
  paper = "Weis92.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Weis94,
  author = "Weispfenning, V.",
  title = {{Quantifier elimination for real algebra  the cubic case}},
  booktitle = "Proc ISSAC'94",
  publisher = "ACM",
  isbn = "0-89791-638-7",
```

```
  pages = "258-263",
  year = "1994",
  abstract =
    "We present a special purpose quantifier elimination method that
    eliminates a quantifier $\exists x$ in formulas $\exists x(\rho)$
    where $\rho$ is a boolean combination of polynomial inequalities of
    degree $\le 3$ with respect to $x$. The method extends the virtual
    substitition of parameterized test points developed in
    [Weispfenning 1, Loos and Weispf.] for the linear case and in
    [Weispfenning 2] for the quadratic case. It has similar upper
    complexity bounds and offers similar advantages (relatively large
    preprocessing part, explicit parametric solutions). Small examples
    suggest that the method will be of practical significance.",
  paper = "Weis94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Weis97,
  author = "Weispfenning, V.",
  title = {{Quantifier elimination for real algebra -
           the quadratic case and beyond}},
  journal =
    "Applicable Algebra in Engineering, Communication and Computing",
  volume = "8",
  number = "2",
  year = "1997",
  pages = "85-101",
  link = "\url{https://link-springer-com.proxy.library.cmu.edu/article/10.1007%2Fs002000050055}",
  abstract =
    "We present a new, ``elementary'' quantifier elimination method for
    various special cases of the general quantifier elimination problem
    for the first-order theory of real numbers. These include the
    elimination of one existential quantifier $\exists x$ in front of
    quantifier-free formulas restricted by a non-trivial quadratic
    equation in $x$ (the case considered also in [7]), and more generally in
    front of arbitrary quantifier-free formulas involving only polynomials
    that are quadratic in $x$. The method generalizes the linear quantifier
    elimination method by virtual substitution of test terms in [9]. It
    yields a quantifier elimination method for an arbitrary number of
    quantifiers in certain formulas involving only linear and quadratic
    occurrences of the quantified variables. Moreover, for existential
    formulas $\phi$ of this kind it yields sample answers to the query
    represented by $\phi$. The method is implemented in REDUCE as part of the
    REDLOG package (see [4,5]). Experiments show that the method is
    applicable to a range of benchmark examples, where it runs in most
    cases significantly faster than the QEPCAD package of Collins and
    Hong. An extension of the method to higher degree polynomials using
    Thoms lemma is sketched.",
  paper = "Weis97.pdf"
}
```

———-

— **axiom.bib** —

```
@InCollection{Weis98,
  author = "Weispfenning, V.",
  title = {{A New Approach to Quantifier Elimination for Real Algebra}},
  booktitle = "Quantifier Elimination and Cylindrical Algebraic Decomposition",
  publisher = "Springer",
  year = "1998",
  isbn = "3-211-82794-3",
  abstract =
    "Quantifier elimination for the elementary formal theory of real
    numbers is a facinating area of research at the intersection of
    various field of mathematics and computer science, such as
    mathematical logic, commutative algebra and algebraic geometry,
    computer algebra, computational geometry and complexity
    theory. Originally the method of quantifier elimination was invented
    (among others by Th. Skolem) in mathematical logic as a technical tool
    for solving the decision problem for a formalized mathematical
    theory. For the elementary formal theory of real numbers (or more
    accurately of real closed fields) such a quantifier elimination
    procedure was established in the 1930s by A. Tarski, using an
    extension of Sturm's theorem of the 1830s for counting the number of
    real zeros of a univariate polynomial in a given interval. Since then
    an abundance of new decision and quantifier elimination methods for
    this theory with variations and optimizations has been published with
    the aim both of establishing the theoretical complexity of the problem
    and of finding methods that are of practical importance (see Arnon
    1988a and the discussion and references in Renegar 1992a, 1992b, 1992c
    for a comparison of these methods). For subproblems such as
    elimination of quantifiers with respect to variables, that are
    linearly or quadratically restricted, specialized methods have been
    developed with good success (see Weispfenning 1988, Loos and
    Weispfenning 1993; Hong 1992d; Weispfenning 1997).",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Wils14,
  author = "Wilson, David and Bradford, Russell and Davenport, James H. and
            England, Matthew",
  title = {{Cylindrical Algebraic Sub-Decompositions}},
  link = "\url{https://arxiv.org/pdf/1401.0647.pdf}",
  year = "2014",
  abstract =
    "Cylindrical algebraic decompositions (CADs) are a key tool in real
    algebraic geometry, used primarily for eliminating quantifiers over
```

```
    the reals and studying semi-algebraic sets. In this paper we
    introduce cylindrical algebraic sub-decompositions (sub-CADs), which
    are subsets of CADs containing all the information needed to specify a
    solution for a given problem.  We define two new types of sub-CAD:
    variety sub-CADs which are those cells in a CAD lying on a designated
    variety; and layered sub-CADs which have only those cells of
    dimension higher than a specified value. We present algorithms to
    produce these and describe how the two approaches may be combined with
    each other and the recent theory of truth-table invariant CAD.  We
    give a complexity analysis showing that these techniques can offer
    substantial theoretical savings, which is supported by experimentation
    using an implementation in Maple.",
  paper = "Wils14.pdf"
}
```

———-

1.36.18 Z

— **axiom.bib** —

```
@book{Zach20,
  author = "Zach, Richard",
  title = {{The Open Logic Text}},
  year = "2020",
  publisher = "The Open Logic Project",
  link = "\url{http://builds.openlogicproject.org/open-logic-complete.pdf}",
  paper = "Zach20.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Zhao11,
  author = "Zhao, Ting and Wang, Dongming and Hong, Hoon",
  title = {{Solution formulats for cubic equations without or
            with constraints}},
  journal = "J. Symbolic Computation",
  volume = "46",
  pages = "904-918",
  year = "2011",
  abstract =
    "We present a convention (for square/cubic roots) which provides
    correct interpretations of the Lagrange formula for all cubic
    polynomial equations with real coefficients. Using this convention, we
    also present a real solution formula for the general cubic equation
    with real coefficients under equality and inequality constraints.",
  paper = "Zhao11.pdf"
}
```

———-

1.37 Comparison of Computer Algebra System

— axiom.bib —

```
@article{Bern96,
  author = "Benardin, Laurent",
  title = {{A review of symbolic solvers}},
  journal = "SIGSAM Bull.",
  volume = "30",
  number = "1",
  pages = "9-20",
  year = "1996",
  abstract =
    "Solving equations and systems of equations symbolically is a key
    feature of every Computer Algebra System. This review examines the
    capabilities of the six best known general purpose systems to date in
    the area of general algebraic and transcendental equation
    solving. Areas explicitly not covered by this review are differential
    equations and numeric or polynomial system solving as special purpose
    systems exist for these kinds of problems. The aim is to provide a
    benchmark for comparing Computer Algebra Systems in a specific
    domain. We do not intend to give a rating of overall capabilities as
    for example in [9]. 1 The Contestants We compare six major Computer
    Algebra Systems. Axiom 2.0 [7], Derive 3.06 [1], Macsyma 420 [8],
    Maple V R4 [3], Mathematica 2.2 [10], MuPAD 1.2.9 [5] and Reduce 3.6
    [6]. When available, we tried to use the latest shipping version of
    each system. 2 The Problem Set The following table presents the set of
    80 problems that we used to evaluate the different solvers...",
  paper = "Bern96.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Kajl98,
  author = "Kajler, Norbert and Soiffer, Neil",
  title = {{A Survey of User Interfaces for Computer Algebra Systems}},
  journal = "J. Symbolic Computation",
  volume = "25",
  pages = "127-159",
  year = "1998",
  abstract =
    "This paper surveys work within the Computer Algebra community (and
    elsewhere) directed towards improving user interfaces for scientific
    computation during the period 1963--1994. It is intended to be useful
    to two groups of people: those who wish to know what work has been
    done and those who would like to do work in the field. It contains an
```

```
    extensive bibliography to assist readers in exploring the field in more
    depth. Work related to improving human interaction with computer
    algebra systems is the main focus of the paper. However, the paper
    includes additional materials on some closely related issues such as
    structured document editing, graphics, and communication protocols.",
  paper = "Kajl98.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Open11,
  author = "Dos Reis, G.",
  title = {{OpenAxiom}},
  link = "\url{http://www.open-axiom.org}",
  year = "2011"
}
```

———-

— **axiom.bib** —

```
@misc{West95,
  author = "Wester, Michael J.",
  title = {{A Review of CAS Mathematical Capabilities}},
  year = "1995",
  link = "\url{http://math.unm.edu/~wester/cas/Paper.ps}",
  abstract =
    "Computer algebra systems (CASs) have become an important
    computational tool in the last decade. General purpose CASs, which are
    designed to solve a wide variety of problems, have gained special
    prominance. In this paper, the capabilities of seven major general
    purpose CASs (Axiom, Derive, Macsyma, Maple, Mathematica, MuPAD, and
    Reduce) are reviewed on 131 short problems covering a broad range of
    (primarily) symbolic mathematics.

    A demo was developed for each CAS, run and the results
    evaluated. Problems were graded in terms of whether it was easy or
    difficult or possible to produce an answer and if an answer was
    produced, whether it was correct. It is the author's hope that this
    review will encourage the development of a comprehensive CAS test
    suite.",
  paper = "West95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{West99a,
  author = "Wester, Michael J.",
  title = {{A Critique of the Mathematical Abilities of CA Systems}},
  year = "1999",
  link = "\url{http://math.unm.edu/~wester/cas/book/Wester.pdf}",
  url2 = "http://math.unm.edu/~wester/cas_review.html",
  abstract =
    "Computer algebra systems (CASs) have become an essential computational
    tool in the last decade. General purpose CASs, which are designed to
    solve a wide variety of problems, have gained special prominence. In
    this chapter, the capabilities of seven major general purpose CASs
    (Axiom, Derive, Macsyma, Maple, Mathmatica, MuPAD and Reduce) are
    reviewed on 542 short problems covering a broad range of (primarily)
    symbolic mathematics.",
  paper = "West99a.pdf",
  keywords = "axiomref"
}
```

———-

1.38 Finite Fields

— **axiom.bib** —

```
@article{Ashx89,
  author = "Ash, D.W. and Black, I.F. and Vanstone, S.A.",
  title = {{Low Complexity Normal Bases}},
  journal = "Discrete Applied Mathematics",
  volume = "25",
  pages = "191-210",
  year = "1989"
}
```

———-

— **axiom.bib** —

```
@article{Beth91,
  author = "Beth, T. and Geiselmann, W. and Meyer, F.",
  title = {{Finding (Good) Normal Bases in Finite Fields}},
  journal = "Proc. ISSAC '91",
  year = "1991"
}
```

———-

— **axiom.bib** —

```
@article{Gath90,
  author = "Gathen, J. von zur and Giesbrecht, M.",
```

```
  title =  {{Constructing normal bases in finite fields}},
  journal = "J. Symb. Comp.",
  volume = "10",
  pages = "547-570",
  year = "1990"
}
```

———-

— **axiom.bib** —

```
@article{Geis89,
  author = "Geiselmann, W. and Gollmann, D.",
  title = {{Symmetry and Duality in Normal Basis Multiplication}},
  journal = "Proc. AAECC-6, LNCS",
  volume = "357",
  year = "1989"
}
```

———-

— **axiom.bib** —

```
@article{Hube90,
  author = "Huber, K.",
  title = {{Some Comments on Zech's Logarithm}},
  journal = "IEEE Trans. Information Theory",
  volume = "IT-36",
  pages = "946-950",
  year = "1990"
}
```

———-

— **axiom.bib** —

```
@article{Itoh88,
  author = "Itoh, T. and Tsujii, S.",
  title = {{A fast algorithm for computing multiplicative inverses in
           $GF(2^m)$ using normal bases}},
  journal = "Inf. and Comp.",
  volume = "78",
  pages = "171-177",
  year = "1988",
  algebra = "\newline\refto{package INBFF InnerNormalBasisFieldFunctions}",
  abstract =
    "This paper proposes a fast algorithm for computing multiplicative
    inverses in $GF(2^m)$ using normal bases. Normal bases have the
    following useful property: In the case that an element $x$ in
    $GF(2^m)$ is represented by normal bases, $2^k$ power operation of an
    element $x$ in $GF(2^m)$ can be carried out by $k$ times cyclic shift
```

```
    of its vector representation.  C.C. Wang et al. proposed an algorithm
    for computing multiplicative inverses using normal bases, which
    requires $(m-2)$ multiplications in $GF(2^m)$ and $(m-1)$ cyclic
    shifts. The fast algorithm proposed in this paper also uses normal
    bases, and computes multiplicative inverses iterating multiplications
    in $GF(2^m)$. It requires at most $2[log_2(m-1)]$ multiplications in
    $GF(2^m)$ and $(m-1)$ cyclic shifts, which are much less than those
    required in Wang's method. The same idea of the proposed fast
    algorithm is applicable to the general power operation in $GF(2^m)$
    and the computation of multiplicative inverses in $GF(q^m)$
    $(q=2^n)$.",
  paper = "Itoh88.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Jaco85,
  author = "Jacobson, N.",
  title = {{Basic Algebra I, 2nd ed.}},
  publisher = "W.H. Freeman and Co.",
  year = "1985"
}
```

———-

— **axiom.bib** —

```
@article{Lens82,
  author = "Lenstra, A.K.",
  title = {{Factorization of Polynomials, Comp. Methods in Number Theory
           (part 1)}},
  journal = "Math. Centre Tracts",
  volume = "154",
  year = "1982"
}
```

———-

— **axiom.bib** —

```
@article{Lens91,
  author = "Lenstra Jr., H.W.",
  title = {{Finding Isomorphisms between Finite Fields}},
  journal = "Math. of Comp.",
  volume = "56",
  number = "193",
  pages = "329-347",
  year = "1991"
}
```

———-

— **axiom.bib** —

```
@article{Lens87,
  author = "Lenstra, H. W. and Schoof, R. J.",
  title = {{Primitive Normal Bases for Finite Fields}},
  journal = "Mathematics of Computation",
  volume = "48",
  number = "177",
  year = "1987",
  pages = "217-231",
  link = "\url{http://www.math.leidenuniv.nl/~hwl/PUBLICATIONS/}",
  algebra = "\newline\refto{package FFPOLY FiniteFieldPolynomialPackage}",
  abstract =
    "It is proved that any finite extension of a finite field has a normal
    basis consisting of primitive roots",
  paper = "Lens87.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Lidl83,
  author = "Lidl, Rudolf and Niederreiter, Harald",
  title = {{Finite Field, Encyclopedia of Mathematics and Its Applications}},
  volume = "20",
  publisher = "Cambridge Univ. Press",
  year = "1983",
  isbn = "0-521-30240-4",
  algebra =
   "\newline\refto{category FAXF FiniteAlgebraicExtensionField}
    \newline\refto{domain FF FiniteField}
    \newline\refto{domain FFCG FiniteFieldCyclicGroup}
    \newline\refto{domain FFCGX FiniteFieldCyclicGroupExtension}
    \newline\refto{domain FFCGP FiniteFieldCyclicGroupExtensionByPolynomial}
    \newline\refto{domain FFX FiniteFieldExtension}
    \newline\refto{domain FFP FiniteFieldExtensionByPolynomial}
    \newline\refto{domain FFNB FiniteFieldNormalBasis}
    \newline\refto{domain FFNBX FiniteFieldNormalBasisExtension}
    \newline\refto{domain FFNBP FiniteFieldNormalBasisExtensionByPolynomial}
    \newline\refto{domain IFF InnerFiniteField}
    \newline\refto{domain IPF InnerPrimeField}
    \newline\refto{domain PF PrimeField}
    \newline\refto{package INBFF InnerNormalBasisFieldFunctions}
    \newline\refto{package FFPOLY2 FiniteFieldPolynomialPackage2}
    \newline\refto{package FFPOLY FiniteFieldPolynomialPackage}
    \newline\refto{package FFHOM FiniteFieldHomomorphisms}
    \newline\refto{package FFF FiniteFieldFunctions}"
}
```

———-

— **axiom.bib** —

```
@book{Lips81,
  author = "Lipson, John D.",
  title = {{Elements of Algebra and Algebraic Computing}},
  publisher = "Addison-Wesley Educational Publishers",
  year = "1981",
  isbn = "978-0201041156",
  algebra = "\newline\refto{category FFIELDC FiniteFieldCategory}"
}
```

———-

— **axiom.bib** —

```
@misc{Lune87,
  author = {L\"uneburg, H},
  title = {{On the Rational Normal Form of Endomorphisms}},
  comment = "BI-Wissenschaftsverlag",
  year = "1987"
}
```

———-

— **axiom.bib** —

```
@techreport{Grab92,
  author = "Grabmeier, Johannes and Scheerhorn, Alfred",
  title = {{Finite fields in Axiom}},
  type = "technical report",
  number = "AXIOM Technical Report TR7/92 (ATR/5)(NP2522)",
  institution = "Numerical Algorithms Group, Inc.",
  address = "Downer's Grove, IL, USA and Oxford, UK",
  year = "1992",
  link = "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
  algebra =
   "\newline\refto{category CHARNZ CharacteristicNonZero}
    \newline\refto{category FPC FieldOfPrimeCharacteristic}
    \newline\refto{category XF ExtensionField}
    \newline\refto{category FFIELDC FiniteFieldCategory}
    \newline\refto{category FAXF FiniteAlgebraicExtensionField}
    \newline\refto{domain SAE SimpleAlgebraicExtension}
    \newline\refto{domain IPF InnerPrimeField}
    \newline\refto{domain PF PrimeField}
    \newline\refto{domain FFP FiniteFieldExtensionByPolynomial}
    \newline\refto{domain FFCGP FiniteFieldCyclicGroupExtensionByPolynomial}
    \newline\refto{domain FFNBP FiniteFieldNormalBasisExtensionByPolynomial}
    \newline\refto{domain FFX FiniteFieldExtension}
```

```
    \newline\refto{domain FFCGX FiniteFieldCyclicGroupExtension}
    \newline\refto{domain FFNBX FiniteFieldNormalBasisExtension}
    \newline\refto{domain IFF InnerFiniteField}
    \newline\refto{domain FF FiniteField}
    \newline\refto{domain FFCG FiniteFieldCyclicGroup}
    \newline\refto{domain FFNB FiniteFieldNormalBasis}
    \newline\refto{package DLP DiscreteLogarithmPackage}
    \newline\refto{package FFF FiniteFieldFunctions}
    \newline\refto{package INBFF InnerNormalBasisFieldFunctions}
    \newline\refto{package FFPOLY FiniteFieldPolynomialPackage}
    \newline\refto{package FFPOLY2 FiniteFieldPolynomialPackage2}
    \newline\refto{package FFHOM FiniteFieldHomomorphisms}
    \newline\refto
        {package FFFACTSE FiniteFieldFactorizationWithSizeParseBySideEffect}",
  abstract =
    "Finite fields play an important role for many applications (e.g. coding
    theory, cryptograpy). There are different ways to construct a finite
    field for a given prime power. The paper describes the different
    constructions implemented in AXIOM. These are {\sl polynomial basis
    representation}, {\sl cyclic group representation}, and {\sl normal
    basis representation}. Furthermore, the concept of the implementation,
    the used algorithms and the various datatype coercions between these
    representations are discussed.",
  paper = "Grab92.pdf",
  keywords = "axiomref",
  beebe = "Grabmeier:1992:FFA"
}
```

———-

— Grabmeier:1992:FFA —

```
@TechReport{Grabmeier:1992:FFA,
  author =       "J. Grabmeier and A. Scheerhorn",
  title =        {{Finite Fields in AXIOM}},
  type =         "AXIOM Technical Report",
  number =       "TR7/92 (ATR/5) (NP2522)",
  institution =  inst-NAG,
  address =      inst-NAG:adr,
  pages =        "??",
  month =        dec,
  year =         "1992",
  bibdate =      "Fri Dec 29 16:31:49 1995",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
  acknowledgement = ack-nhfb,
}
```

———-

— axiom.bib —

```
@article{Mull88,
  author = "Mullin, R.C. and Onyszchuk, I.M. and Vanstone, S.A.",
  title = {{Optimal Normal Bases in $GF(p^n)$}},
  journal = "Discrete Applied Mathematics",
  volume = "22",
  pages = "149-161",
  year = "1988"
}
```

———-

— **axiom.bib** —

```
@misc{Nick88,
  author = "Nickel, W.",
  title = {{Endliche K\"orper in dem gruppentheoretischen Programmsystem GAP}},
  comment = "Diplomarbeit, RWTH Aachen",
  year = "1988"
}
```

———-

— **axiom.bib** —

```
@article{Odly85,
  author = "Odlyzko, A.M.",
  title = {{Discrete logarithms in finite fields and their cryptographic
           significance}},
  journal = "Proc. Eurocrypt '84, LNCS",
  volume = "209",
  publisher = "Springer-Verlag",
  pages = "224-314",
  year = "1985"
}
```

———-

— **axiom.bib** —

```
@article{Pinc89,
  author = "Pincin, A",
  title = {{Bases for finite fields and a canonical decomposition for a
           normal basis generator}},
  journal = "Communications in Algebra",
  volume = "17",
  number = "6",
  pages = "1337-1352",
  year = "1989"
}
```

———-

— **axiom.bib** —

```
@article{Pohl78,
  author = "Pohlig, S.C. and Hellman, M.",
  title = {{An improved algorith for computing logarithms over $GF(p)$
           and its cryptographic significance}},
  journal = "IEEE Trans Information Theory",
  volume = "IT-24",
  pages = "106-110",
  year = "1978"
}
```

———-

— **axiom.bib** —

```
@article{Rybo89,
  author = "Rybowicz, M",
  title = {{Search of primitive polynomials over finite fields}},
  journal = "J. Pure Appl.",
  volume = "65",
  pages = "139-151",
  year = "1989"
}
```

———-

— **axiom.bib** —

```
@misc{Salo16a,
  author = "Salomone, Matthew",
  title = {{Group Theory II}},
  year = "2016",
  institution = "Bridgewater State University",
  link = "\url{http://axiom-developer.org/axiom-website/GroupTheoryII/Salomone.html}"
}
```

———-

— **axiom.bib** —

```
@misc{Sche92,
  author = "Scheerhorn, A.",
  title = {{Trace- and Norm-Compatible Extensions of Finite Fields}},
  journal = "Appl. Alg. in Eng., Comm. and Comp.",
  year = "1992"
}
```

———-

— **axiom.bib** —

```
@misc{Sche93,
  author = "Scheerhorn, Alfred",
  title = {{Presentation of the algebraic closure of finite fields and
           trace-compatible polynomial sequences}},
  comment = "Darstellungen des algebraischen Abschlusses endlicher Korper
             und spur-kompatible Polynomfolgen",
  year = "1993",
  abstract =
    "For numerical experiments concerning various problems in a finite
    field $\mathbb{F}_q$ it is useful to have an explicit data
    presentation $\mathbb{F}_{q^m}$ of for large $m$, and a method for the
    construction of towers
    \[\mathbb{F}_q \subset \mathbb{F}_{q^{d_1}} \subset \cdots \subset
    \mathbb{F}_{q^{d_k}} = \mathbb{F}_{q^m}\]
    In order to avoid the identification problem it is advantageous to
    have all fields in the tower presented by properly chosen normal bases,
    whereby the embedding
    $\mathbb{F}_{q^{d_i}} \subset \mathbb{F}_{q^{d_{i+1}}}$
    is given by the trace function.

    The following notion is introduced: A sequence of polynomials
    $\{f_n | n \ge 1\}$ with degree$(f_n)=n$ called trace-compatible over
    $\mathbb{F}_q$ if (1) $f_n$ is a normal polynomial over $\mathbb{F}_q$,
    (2) if $\alpha_n \in \mathbb{F}_{q^n}$ is a root of $f_n$, then for any
    proper divisor $d$ of $n$ the trace of $\alpha_n$ over $\mathbb{F}_{q^d}$
    is a root of $f_d$.

    The main goal of the dissertation is to give algorithms for
    construction of sequences of trace-compatible polynomials and to
    present explicit numerical data. An analogous notion of
    norm-compatible sequences is also introduced and studied.

    The dissertation consists of four chapters and a supplement, as
    follows: (1) Basic notions (1-31). (2) Presentation of the algebraic
    closure of a finite field (32-59). (3) Sequences of polynomials and
    sequences of elements (60-115). (4) Implementations (118-139). (5)
    Supplement (142-171).

    In chapters (1)(3) various known results and algorithms are
    collected, and new results are added and compared with those
    previously used.

    The numerical results in the supplement contain sequences of
    trace-compatible polynomials of degree $n$, where $n \le 100$, and
    $q=2,3,5,7,11,13$. For implementation, the computer-algebra system
    AXIOM has been used. The details contained in this dissertation are
    not readily describable in a short review.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Shou92,
  author = "Shoup, V.",
  title = {{Searching for Primitive Roots in Finite Fields}},
  journal = "Math. of Comp.",
  volume = "58",
  number = "197",
  pages = "369-380",
  year = "1992"
}
```

———-

— **axiom.bib** —

```
@article{Stin90,
  author = "Stinson, D.R.",
  title = {{Some observations on parallel Algorithms for fast exponentiation
           in $GF(2^n)$}},
  journal = "Siam J. Comp.",
  volume = "19",
  number = "4",
  pages = "711-717",
  year = "1990",
  algebra = "\newline\refto{package INBFF InnerNormalBasisFieldFunctions}",
  abstract =
    "A normal basis represention in $GF(2^n)$ allows squaring to be
    accomplished by a cyclic shift. Algorithms for multiplication in
    $GF(2^n)$ using a normal basis have been studied by several
    researchers. In this paper, algorithms for performing exponentiation
    in $GF(2^n)$ using a normal basis, and how they can be speeded up by
    using parallelization, are investigated.",
  paper = "Stin90.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Vars81,
  author = "Varshamov, Gamkrelidze",
  title = {{Method of construction of primitive polynomials over
            finite fields}},
  journal = "Soobsheh. Akad. Nauk Gruzin.",
  volume = "99",
  pages = "61-64",
  year = "1981"
}
```

———-

— **axiom.bib** —

```
@article{Wass89,
  author = "Wassermann, A.",
  title = {{Konstruktion von Normalbasen}},
  journal = "Bayreuther Math. Schriften",
  volume = "31",
  pages = "1-9",
  year = "1989"
}
```

———-

1.39 To Be Classified

— **axiom.bib** —

```
@article{Bozm84,
  author = "Bozman, Gerald and Buco, William and Daly, Timothy and
            Tetzlaff, William",
  title = {{Analysis of Free Storage Algorithms - Revisited}},
  journal = "IBM Systems Journal",
  volume = "23",
  number = "1",
  year = "1984",
  abstract =
    "Most research in free-storage management has centered around
    strategies that search a linked list and strategies that partition
    storage into predetermined sizes. Such algorithms are analyzed in
    terms of CPU efficiency and storage efficiency. The subject of this
    study is the free-storage management in the Virtual Machine/System
    Product (VM/SP) system control program. As a part of this study,
    simulations were done of established, and proposed, dynamic storage
    algorithms for the VM/SP operating system. Empirical evidence is given
    that simplifying statistical assumptions about the distribution of
    interarrival times and holding times has high predictive
    ability. Algorithms such as first-fit, modified first-fit, and
    best-fit are found to be CPU-inefficient. Buddy systems are found to
    be very fast but suffer from a high degree of internal
    fragmentation. A form of extended subpooling is shown to be as fast as
    buddy systems with improved storage efficiency. This algorithm was
    implemented for VM/SP, and then measured. Results for this algorithm
    are given for several production VM/SP systems.",
  paper = "Bozm84.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Calm92,
  author = "Calmet, J. and Campbell, J.A.",
  title = {{Artificial Intelligence and Symbolic Mathematical
           Computations}},
  journal = "LNCS",
  volume = "737",
  pages = "1-19",
  year = "1992",
  abstract =
    "This introductory paper summarizes the picture of the territory
    common to AI and SMC that has evolved from discussions following the
    presentation of papers given at the 1992 Karlsruhe conference.  Its
    main objective is to highlight some patterns that can be used to guide
    both sketches of the state of the art in this territory and
    suggestions for future research activities.",
  paper = "Calm92.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Davexx,
  author = "Davenport, J.H.",
  title = {{Computer algebra -- past, present and future}},
  abstract =
    "Computer algebra started in 1953, and there were several systems in
    existence in the 1960's. Those inspired by physical applications
    largely implemented ``high school'' algebra and, from the point of
    view of today's much larger machines and more sophisticated
    programming languages, the miracle is that they worked at all, or as
    efficiently as they did.

    By the end of the 1960's it was clear that more sophisticated
    algorithms were necessary, either to solve problems for which the
    ``high school'' algorithms were inefficient on large data (e.g. gcd or
    factorization), or problems for which the ``high school'' techniques
    were not really algorithms at all (e.g. integration, solution of sets
    of equations).

    Hence the 1970's (and indeed much of the 1980's) were the ``age of
    algorithms''. It rapidly became obvious that these algorithms required
    more complex data structures and mathematical objects: finite fields,
    ideals, algebraic curves, divisor class groups to name but a few. This
    led to the growth of new systems, such as Axiom (formerly Scratchpad),
    Maple, Mathematica and Reduce 3. It is currently the case that many
    more algorithms are known than are implemented, and certainly that few
    systems implement even a reasonable cross-section of the known
    algorithms.

    At the present, there are two main trends. One is the rush to
    implement, which is causing a lot of duplication of work, but there is
    also a realisation that these systems need to be able to communicate,
```

```
    and that it is inherently impossible to have all the best algorithms
    in one system. To take an example, why implement from scratch enough
    group theory to analyse blocks of imprimitivity in a permutation
    group, when Cayley has all this and much more? However, parts of
    integration theory require this analysis.

    The other trend is the tendency to more ``structure-oriented''
    algorithms, i.e. algorithms which take account of the structure of the
    problem. To name two, there is Gatemann's work on polynomial equation
    systems with symmetry, and Richardson's work on roots of polynomials
    which can be written as $p(x,x^n)$ with $p$ of low degree.

    The paper concludes with some speculations on the future of computer
    algebra.",
  paper = "Davexx.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Fate72,
  author = "Fateman, Richard J.",
  title = {{Essays in Algebraic Simplification}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/tr-95.pdf}",
  school = "MIT",
  comment = "MAC TR-95 technical report",
  year = "1972",
  abstract =
    "This thesis consists of essays on several aspects of the problem
    of algebraic simplification by computer. We first discuss a pattern
    matching system intended to recognize non-obvious occurrences of
    patterns within Algebraic expression. A user of such a system can
    ``teach'' the computer new simplification rules. Then we report on
    new applications of canonical simplification of rational functions.
    These applications include techniques for picking out coefficients,
    and for substituting for summs, products, quotients, etc. Our final
    essay is on a new, practical, canonical simplification algorithms
    for radical expressions (i.e. algebraic expressions including roots
    of polynomials). The effectiveness of the procedure is assured
    through proofs of appropriate properties of the simplified forms.
    Two appendices describe MACSYMA, a computer system for algebraic
    manipulations, which served as the basis for this work.",
  paper = "Fate72.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate08b,
```

```
  author = "Fateman, Richard J.",
  title = {{Applications and Methods for Recognition of (Anti)-Symmetric
           Functions}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/symmetry.pdf}",
  year = "2008",
  abstract =
    "One of the important advantages held by computer algebra systems (CAS)
    over purely-numerical computational frameworks is that the CAS can
    provide a higher-level ``symbolic'' viewpoint for problem
    solving. Sometimes this can convert apparently impossible problems to
    trivial ones. Sometimes the symbolic perspective can provide
    information about questions which cannot be directly answered, or
    questions which might be hard to pose. For example, we might be able
    to analyze the asymptotic behavior of a solution to a differential
    equation even though we cannot solve the equation. One route to
    implicitly solving problems is the use of symmetry arguments. In this
    paper we suggest how, through symmetry, one can solve a large class of
    definite integration problems, including some that we found could not
    be solved by computer algebra systems. One case of symmetry provides
    for recognition of periodicity, and this solves additional problems,
    since removal of periodic components can be important in integration
    and in asymptotic expansions.",
  paper = "Fate08b.pdf"
}
```

———-

— axiom.bib —

```
@misc{Fate03b,
  author = "Fateman, Richard J.",
  title = {{Manipulation of Matrices Symbolically}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/symmat2.pdf}",
  year = "2003",
  abstract =
    "Traditionally, matrix algebra in computer algebra systems is
    ``implemented'' in three ways:
    \begin{itemize}
    \item numeric explicit computation in a special arithmetic domain:
    exact rational or integer, high-precision software floating-point,
    interval, or conventional hardware floating-point.
    \item symbolic explicit computation with polynomial or other
    expression entries,
    \item (implicit) matrix computation with symbols defined over a
    (non-commuting) ring.
    \end{itemize}
    Manipulations which involve matrices of indefinite size (n  m) or
    perhaps have components which are block submatrices of indefinite size
    have little or no support in general-purpose computer algebra systems,
    in spite of their importance in theorems, proofs, and generation of
    programs. We describe some efforts to design and implement tools for
    this mode of thinking about matrices in computer systems.",
  paper = "Fate03b.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Fate09a,
  author = "Fateman, Richard J.",
  title = {{Simplifying RootSum Expressions}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/rootsum.pdf}",
  year = "2009",
  abstract =
    "It's useful to sum an expression with a parameter varying over all
    the roots of a given polynomial. Here's a defense of that statement
    and a method to do the task.",
  paper = "Fate09a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate09,
  author = "Fateman, Richard J.",
  title = {{Rational Integration, Simplified}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/root-integ.pdf}",
  year = "2009",
  abstract =
    "After all this computer algebra stuff, and several PhD theses in
    the last few decades, what more could we say about symbolic
    rational function integration?

    How about a closed formula for the result, subject to a few algebraic
    side-conditions, which works even with parameters in the denominator?",
  paper = "Fate09.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate99b,
  author = "Fateman, Richard J.",
  title = {{Generation and Optimization of Numerical Programs by
           Symbolic Mathematical Methods}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/RIMS.pdf}",
  year = "1999",
  abstract =
    "Symbolic mathematical methods and systems
    \begin{itemize}
    \item support scientific and engineering ``problem solving environments''
```

```
    (PSEs),
    \item The specific manipulation of mathematical models as a precursor
    to the coding of algorithms
    \item Expert system selection of modules from numerical libraries and
    other facilities
    \item The production of custom numerical software such as derivatives
    or non-standard arithmetic code-generation packages,
    \item The complete solution of certain classes of mathematical problems
    that simply cannot be handled solely by conventional floating-point
    computation.
    \end{itemize}

    Viewing computational objects and algorithms from a symbolic
    perspective and then specializing them to numerical or graphical views
    provides substantial additional flexibility over a more conventional view.

    We also consider interactive symbolic computing as a tool to provide
    an organizing principle or glue among otherwise dissimilar components.",
  paper = "Fate99b.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fate07,
  author = "Fateman, Richard J.",
  title = {{Rational Function Computing with Poles and Residues}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/qd.pdf}",
  year = "2007",
  abstract =
    "In a numerical calculation sometimes we need higher-than
    double-precision floating-point arithmetic to allow us to be confident
    of a result. One alternative is to rewrite the program to use a
    software package implementing arbitrary-precision extended
    floating-point arithmetic such as ARPREC or MPFR, and try to choose a
    suitable precision.

    Such an arithmetic scheme, in spite of helpful tools, may be
    inconvenient to write. There are also facilities in computer algebra
    systems (CAS) for such software-implemented ``bigfloats.'' These
    facilities are convenient if one is already using the CAS. In any of
    these situations the bigfloats may be rather slow, a cost of its
    generality.

    There are possibilities intermediate between the largest hardware
    floating-point format and the general arbitrary-precision software
    which combine a considerable (but not arbitrary) amount of extra
    precision with a (relatively speaking) modest factor loss in
    speed. Sometimes merely doubling the number of bits in a
    double-floating-point fraction is enough, in which case arithmetic on
    double-double (DD) operands would suffice. Another possibility is to
```

```
    go for yet another doubling to quad-double (QD) arithmetic: instead of
    using the machine double-floats to give about 16 decimal digits of
    precision, QD supplies about 64 digits. DD and QD as used here provide
    the same exponent range as ordinary double.

    Here we describe how we incorporated QD arithmetic implemented in a
    library into a Common Lisp system, providing a smooth interface while
    adding only modest overhead to the run-time costs (compared to
    accessing the library from C or C++). One advantage is that we keep
    the program text almost untouched while switching from double to
    quad-double. Another is that the programs can be written, debugged,
    and run in an interactive environment. Most of the lessons from QD can
    be used for other versions of arithmetic which can be embedded in
    Lisp, including MPFR, for indefinite (arbitrary) precision, should QD
    provide inadequate precision or range.",
  paper = "Fate07.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fate13a,
  author = "Fateman, Richard J.",
  title = {{Rational Function Computing with Poles and Residues}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/openmathcrit.pdf}",
  year = "2013",
  abstract =
    "Computer algebra systems (CAS) usually support computation with exact
    or approximate rational functions as ratios of polynomials in
    ‘‘expanded form’’ with explicit coefficients. We examine the
    consequences of introducing a partial-fraction type of form in which
    some of the usual rational operations can be implemented in
    substantially faster times. In this form an expression in one
    variable, say $x$, is expressed as a polynomial in $x$ plus a sum of
    terms each of which has a denominator $x-c$ perhaps to an integer
    power, where $c$ is in general a complex constant. We show that some
    common operations including rational function addition,
    multiplication, and matrix determinant calculation can be performed
    many times faster than in the conventional representation. Polynomial
    GCD operations, the costliest part of rational additions, are entirely
    eliminated. Applicaiton of Cauchy's integral theorem allow for trivial
    integration of an expression around a closed contour. In some cases
    the approximate evaluation of transcendental functions can be
    accelerated, especially in parallel, by evaluation of a formula in
    pole+residue form.",
  paper = "Fate13a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate01a,
  author = "Fateman, Richard J.",
  title = {{A Critique of OpenMath and Thoughts on Encoding Mathematics}},
  year = "2001",
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/openmathcrit.pdf}",
  algebra =
    "\newline\refto{category OM OpenMath}
     \newline\refto{domain COMPLEX Complex}
     \newline\refto{domain DFLOAT DoubleFloat}
     \newline\refto{domain FLOAT Float}
     \newline\refto{domain FRAC Fraction}
     \newline\refto{domain INT Integer}
     \newline\refto{domain LIST List}
     \newline\refto{domain SINT SingleInteger}
     \newline\refto{domain STRING String}
     \newline\refto{domain SYMBOL Symbol}
     \newline\refto{package OMEXPR ExpressionToOpenMath}
     \newline\refto{package OMSERVER OpenMathServerPackage}",
  abstract =
    "The OpenMath project, as portrayed in the Special Issue of the SIGSAM
    Bulletin (volume 34 no. 2), seems to have a number of problems to
    face. One of them is the (apparently implicit) assumption that
    OpenMath designers, through dint of mathematical thought and the
    advice of the members of the Open Math Society, have solved, in their
    domain, one of the most pressing problems of software engineering
    today, namely software re-use. After six years there is insufficient
    evidence on which to base any claims of success and it appears that
    most substantive practical issues of mathematical representation and
    communication have yet to be addressed. We also raise questions about
    related computational mathematical goals and mathematical encodings.",
  paper = "Fate01a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate08a,
  author = "Fateman, Richard J.",
  title = {{Verbs, Nouns, and Computer Algebra, or What's Grammar Got
           to do with Math?}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/nounverbmac.pdf}",
  year = "2008",
  abstract =
    "Computer algebra systems (CAS) are supposed to do mathematics.
    Unfortunately, much of human mathematics presentation and
    manipulation depends on humans to understand what is going on more
    generally. This context is often not made available to the computer
    system, and so it will sometimes fail to make the right guess at the
    meaning of constructions, unless they are made unambiguous. Among many
    issues, one that has been raised and resolved, is the need for
    attributing properties to operators that distinguish between ‘‘the name
```

```
    of the operation'' and ``the operation itself''.

    As a simple example, referential transparency requires that if $x = y$,
    then any true statement $P(x)$ about $x$ must be true about $y$, namely
    $P(y)$ is also true.

    Consider now the well-known equation $sin(0) = 0$, and the statement
    $P(r) :=$ ``r = 0 is a rule to simplify the sin() function''.  The
    statement $P(sin(0))$ is clearly true. Therefore $P(0)$ should be true
    as well. It is not, since $0 = 0$ is not a rule to simplify $sin()$.

    The point here is that $sin(0)$ is not equal to 0 if sin is treated as
    the name of an operation. Perhaps it is a character string, or a data
    structure of some sort: a noun phrase. The corresponding expression
    including the verb form of sin is one in which $sin(0)$ can be
    simplified to something equivalent, but in some sense preferable. Here
    it is reasonable to replace it by 0.",
  paper = "Fate08a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate97c,
  author = "Fateman, Richard J.",
  title = {{Excerpts from a proposal to the National Science Foundation
           on Programming Environments and Tools for Advanced Scientific
           Computation}},
  year = "1997",
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/newpse.pdf}",
  abstract =
    "We propose to concentrate on investigations of technology in support
    of advanced scientific computation, based on a higher-level view of the
    relationship of mathematical modeling to computation. This is based on
    the following considerations:
    \begin{enumerate}
    \item Symbolic mathematical manipulation and computer algebra systems:
    These are tools and techniques that support problem solving, code
    development, comprehension, debugging and partial evaluation
    (the principle behind many optimization techniques).
    \item Data modeling: in particular, the use of IEEE floating-point
    number representations and operations.
    \item Closed form or exact solutions: For some sub-problems of
    scientific interest, numerical solutions are not the only paradigm.
    The availability of exact solutions, or symbolic-approximation solutions
    has advanced along with computing technology too.
    \item The use of scripting languages: the linkage of high-performance
    library routines to graphical interfaces for data analysis and model setup
    can be enhanced by symbolic support and high-level scripting languages.
    \item Extending the use of remote computation. We are now using standard
    network connections for the acquisition of complex data (in this case,
    symbolic solution of definite integrals).
```

```
    \end{enumerate}",
  paper = "Fate97c.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fate98,
  author = "Fateman, Richard J.",
  title = {{Open Problems in Human-Computer Interaction with Specific
           Application to Computer-Aided Mathematics}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/open4.ps}",
  year = "1998",
  abstract =
    "It is our intention, in this short paper, to indicate some of the
    problems of human-computer interactions at the boundaries of current
    research is algebraic manipulation as we see the field now, some 35
    years from the development of its first software systems (the
    now-ancient Macsyma was announced in 1971, and it had precurors.)

    Because computers, algorithms, and software systems have advanced so
    substantially, there is a tendency for computer scientists to think
    that all of mathematics can be done by computer, or at least that we
    are on track for the continued ``arithmetization of mathematics.''
    Working mathematicians tend to disagree with the characterization that
    we are at all close now, nor that we are headed toward any goal of
    being able to do or understand mathematics.

    Most of the problems which we indicate below are actually quite
    general problems in computer-human communication. For specificity (and
    the possibility that one could qunatify a solution!) we will count our
    discussion of in terms of applications in the area of algebraic
    manipulation. If we cannot handle allegedly well-formed mathematical
    issues, how can we expect to handle ill-formed ambigious
    ``natural''communication?

    Some of the problems are representative of strategic mathemaical
    manipulation procedures which lack totally algorithmic approaches at
    this time, and so are more of an interactive problem-solving protocol
    than a fixed recipe.",
  paper = "Fate98.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fate00a,
  author = "Fateman, Richard J.",
```

```
  title = {{Improving Exact Integral From Symbolic Algebra Systems}},
  year = "2000",
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/nform2col.pdf}",
  abstract =
    "Programs in symbolic algebraic manipulation systems can compute
    certain classes of symbolic indefinite integrals in closed form. Although
    these answers are ordinarily formally correct algebraic anti-derivatives,
    their form is often unsuitable for further numerical or even analytical
    processing. In particular, we address cases in which such ‘‘exact answers’’
    when numerically evaluated may give less-accurate answers than numerical
    approximations from first principles! The symbolic formulas may also
    behave inappropriately near singularities. We discuss techniques, based
    in part on the calculus of divided differences, for improving the form of
    results of symbolic mathematics systems. In particular, computer alge
    algebra systems must take explicit account of the possibility that they are
    producing not mathematics but templates of programs consisting of se
    sequences of arithmetic operations. In brief, mathematical correctness is
    not enough. Forms produced by rational integration programs are used
    for examples.",
  paper = "Fate00a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate03,
  author = "Fateman, Richard J.",
  title = {{Continuity and Limits of Programs}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/limit.pdf}",
  year = "2003",
  abstract =
    "In demonstrations of proofs [3] of correctness for programs that are
    designed to compute mathematical functions we attempted to show that
    programs have the right properties, namely the same properties as the
    mathematical functions they are alleged to compute. In the cited paper
    we were forced to hand-wave (our excuse was the need for brevity) in
    at least two places, trying to side-step sticky issues. Here we try to
    address these issues by clarifying two concepts: (a) computational
    continuity and (b) equality in a domain where floating-point
    computations can be done to variable (presumably high) precision. We
    introduce notations p-representable and p-negligible where p denotes
    precision, and show how this helps in our applications.",
  paper = "Fate03.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Fate15,
```

```
  author = "Fateman, Richard J.",
  title = {{Interval Arithmetic, Extended Numbers and Computer Algebra
           Systems}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/interval.pdf}",
  year = "2015",
  abstract =
    "Many ambitious computer algebra systems were initially designed in a
    flush of enthusiasm, with the goal of automating any symbolic
    mathematical manipulation correctly. Historically, this approach
    resulted in programs that implicitly used certain identities to
    simplify expressions. These identities, which very likely seemed
    universally true to the programmers in the heat of writing the CAS,
    (and often were true in well-known abstract algebraic domains) later
    needed re-examination when such systems were extended for dealing with
    kinds of objects unanticipated in the original design. These new
    objects are generally introduced to the CAS by extending generically
    the arithmetic or other operations.  For example, approximate floats
    do not have the mathematical properties of exact integers or
    rationals.  Complex numbers may strain a system designed for
    real-valued variables. In the situation examined here, we consider two
    categories of extended numbers:  or undefined, and real
    intervals. We comment on issues raised by these two troublesome
    notions, how their introduction into a computer algebra system may
    require a (sometimes painful) reconsideration and redesign of parts of
    the program, and how they are related. An alternative (followed most
    notably by the Axiom system is to essentially envision a meta CAS
    defined in terms of categories and inheritance with only the most
    fundamental built-in concepts; from these one can build many variants
    of specific CAS features. This approach is appealing but can fails to
    accommodate extensions that violate some mathematical tenets in the
    cause of practicality.",
  paper = "Fate15.pdf"
}
```

———-

— axiom.bib —

```
@misc{Fate14,
  author = "Fateman, Richard J.",
  title = {{Fun with Filon Quadrature - a Computer Algebra Perspective}},
  year = "2014",
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/fun-filon.pdf}",
  abstract =
    "Filon quadrature, useful for oscillatory integrals, is easily
    implemented in a computer algebra system (CAS) using exact or
    approximate arithmetic. Although quadrature is almost always used in
    the context of approximation, we see that various properties are
    exhibited by running an exact algorithm on exact {\sl symbolic}
    inputs. This experimental approach to the mathematics allows us to
    prove the implementation of the algorithm has the expected
    mathematical correctness properties, and is therefore more likely to
    be, itself, a correct implementation.",
```

```
  paper = "Fate14.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate97a,
  author = "Fateman, Richard J.",
  title = {{The Fast Fourier Transform}},
  course = "Berkeley CS 292, Fall 1997, Handout 2",
  year = "1997",
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/fftnotes.pdf}",
  paper = "Fate97a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate97b,
  author = "Fateman, Richard J.",
  title = {{More Versatile Scientific Documents Over-Extended Abstract,
           working paper}},
  year = "1997",
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/mvsd.ps}",
  abstract =
    "The electronic representation of scientific documents: journals,
    technical reports, program documentation, laboratory notebooks,
    etc. present challenges in several distinct communities. We see five
    distinct groups concerned with electronic versions of scientific
    documents:
    \begin{itemize}
    \item Publishers of journals, texts, reference works, and their authors.
    \item Software publishers for OCR/document analysis, document formatting
    \item Software publishers whose products access ``contents semantics''
    from documents, including library keyword search programs, natural language
    search programs, data-base systems, visual presentation systems,
    mathematical computation systems, etc.
    \item Institutions maintain access to electronic libraries, which must
    be broadly construed to include data and programs of all sorts.
    \item Individuals and their programs acting as their agents who need to
    use these libraries, to identify, locate, and retrieive relevant
    documents
    \end{itemize}

    We need to have a convergence in design and standards for encoding new
    or pre-existing (typically paper-based) documents in order to most
    efficiently meet the needs of all these groups. Various effort, some
    loosely coordinated, but just as often competing, are trying to set
    standards and build tools.
```

```
    We approach this by first dividing the task into visual, syntactic,
    and semantic components. These specifications can focus attention on
    the addressing requirements of the different groups. Additionally,
    these components rely on a structure for documents that incorporates
    existing library models and extends them to new modes of operation.
    Berkeley's MVD is one plausible structure which will allow prototype
    development and explorations.",
  paper = "Fate97b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Gent74a,
  author = "Gentleman, W. Morven",
  title = {{Experience with Truncated Power Series}},
  journal = "ACM SIGSAM",
  volume = "8",
  number = "3",
  year = "1974",
  pages = "61-62",
  abstract =
    "The truncated power series package in ALTRAN has been available for
    over a year now, and it has proved itself to be a useful and exciting
    addition to the armoury of symbolic algebra. A wide variety of
    problems have been attacked with this tool: moreover, through use in
    the classroom, we have had the opportunity to observe how a large
    number of people react to the availability of this tool.",
  paper = "Gent74a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Harr79,
  author = "Harrington, Steven J.",
  title = {{A Symbolic Limit Evaluation Program in REDUCE}},
  journal = "ACM SIGSAM",
  volume = "13",
  number = "1",
  year = "1979",
  pages = "27-31",
  abstract =
    "A method for the automatic evaluation of algebraic limits is
    described. It combines many of the techniques previously employed,
    including top-down recursive evaluation, power series expansion, and
    L'Hpital's rule. It introduces the concept of a special algebraic
    form for limits. The method has been implemented in MODE-REDUCE.",
  paper = "Harr79.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Heck97,
  author = "Heckmann, Reinhold and Wilhelm, Reinhard",
  title = {{A Functional Description of TeX's Formula Layout}},
  journal = "J. Functional Programming",
  volume = "7",
  number = "5",
  pages = "451-485",
  year = "1997",
  link = "\url{http://http.cs.berkeley.edu/~fateman/temp/neuform.pdf}",
  comment = "ftp://ftp.cs.uni-sb.de/formulae/formulae.tar.gz",
  abstract =
    "While the quality of the results of TeX's mathematica formula layout
    algorithm is convincing, its original description is hard to
    understand since it is presented as an imperative program with complex
    control flow and destructive manipulations of the data structures
    representing formulae. In this paper, we present a re-implementation
    of TeX's formula layout algorithm in the functional language SML,
    thereby providing a more readable description of the algorithm,
    extracted from the monolithical TeX system.",
  paper = "Heck97.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Jeffxx,
  author = "Jeffrey, David",
  title = {{Real Integration on Domains of Maximum Extent}},
  abstract =
    "General purpose computer algebra systems are used by people with
    widely varying backgrounds. Amongst the many difficulties that face
    the developer because of this, one that is particularly relevant to
    the subject of this talk is the fact that different users attach
    different meanings, or definitions, to the same symbols. When a user
    asks a CAS to integrate a function, it is not clear which definition
    of integral should be used. Some of the disagreements over the
    ``correct'' value of an integral reduce to the fact that the different
    parties are using different defintions. This talk therefore starts by
    defining my version of integration. According to this definition,
    functions returned as integrals should not only differentiate to the
    function supplied by the user, they should also satisfy global
    continuity properties. In order to achieve these properties, the idea
    of a rectifying transform is intrduced. For the problem of integrating
    a rational trigonometric function, a new rectifying transform is
    described."
}
```

———-

— **axiom.bib** —

```
@misc{Jeffxxa,
  author = "Jeffrey, D. J.",
  title = {{The Integration of Functions Containing Fractional Powers}},
  abstract =
    "An algorithm is developed for integrating functions that contain
    fractional powers. The algorithm addresses the following points. The
    integral must be valid for all possible values of the variable,
    including those values of the variable that make the integrand, and
    hence the integral, take complex values. The algorithm must allow for
    the fact that there are two possible interpretations of the cube root
    as a real number (in fact of any odd root), and produce correct
    integrals for both interpretations (it is shown that it is possible
    for the functiona form of the integral to change with the
    interpretation). Finally, all simplifications, especially of complex
    quantities, must follow correct rules, what are here derived using the
    concept of the unwinding number."
}
```

———-

— **axiom.bib** —

```
@misc{Jeffxxb,
  author = "Jeffrey, D.J. and Corless, R.M.",
  title = {Explorations of uses of the unwinding number $\Kappa$},
}
```

———-

— **axiom.bib** —

```
@misc{Kohl08,
  author = "Kohlhase, Michael",
  title = {{Using Latex as a Semantic Markup Format}},
  year = "2008",
  link = "\url{https://kwarc.info/kohlhase/papers/mcs08-stex.pdf}",
  abstract =
    "One of the great problems of Mathematical Knowledge Management (MKM)
    systems is to obtain access to a sufficiently large corpus of
    mathematical knowledge to allow the management / search / naviation
    techniques developed by the community to display their strength. Such
    systems usually expect the mathematical knowledge they operate on in
    the form of semantically enhanced documents, but mathematicians and
    publishers in Mathematics have heavily invested into the Tex/Latex
    format and workflow.
```

```
    We analyze the current practice of semi-semantic markup in Latex
    documents and extend it by a markup infrastructure that allows to
    embed semantic annotations into latex documents without changing their
    visual appearance. This collection of tex macro packages is called
    stex (semantic tex) as it allows to markup latex documents
    semantically without leaving the time-tried tex/latex workflow,
    essentially turning latex into an MKM format. At the heart of stex is
    a definition mechanism for semantic macros for mathematical objects
    and a non-standard scoping construct for them, which is oriented at
    the semantic dependency relation rather than the document structure.

    We evaulate the stex macro collection on a large case study: the
    course materials of a two-semester course in Computer Science was
    annotated semantically and coverted to the OMDOC MKM format by Bruce
    Miller's LatexML system.",
  paper = "Kohl08.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Miol91,
  author = "Miola, Alfonso",
  title = {{Symbolic Computation and Artificial Intelligence}},
  journal = "LNCS",
  volume = "535",
  pages = "243-255",
  year = "1991",
  abstract =
    "The paper presents an overview of the research achievements on issues
    of common interest for Symbolic Computation and Artificial
    Intelligence.  Common methods and techniques of non-numerical
    information processing and of automated problem solving are underlined
    together with specific applications.  A qualitative analysis of the
    symbolic computation systems currently available is presented in
    view of the design and implementation of a new system.  This system
    allows both formal algebraic and analytical computations and automated
    deduction to prove properties of the computation. ",
  paper = "Miol91.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Nore08,
  author = "Norell, Ulf and Chapman, James",
  title = {{Dependently Typed Programming in Agda}},
  link = "\url{http://www.cse.chalmers.se/~ulfn/papers/afp08/tutorial.pdf}",
  year = "2008",
  paper = "Nore08.pdf",
```

```
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Paul14,
  author = "Paule, Peter",
  title = {{Complex Variables Visualized}},
  school = "RISC Linz",
  year = "2014",
  link =
  "\url{http://www.risc.jku.at/publications/download/risc_5011/DiplomaThesisPonweiser.pdf}",
  abstract =
    "The aim of this diploma thesis is the visualization of some
    fundamental results in the context of the theory of the modular group
    and modular functions. For this purpose the computer algebra software
    Mathematica is utilized.

    The thesis is structured in three parts. In
    Chapter 1, we summarize some important basic concepts of group theory
    which are relevant to this work.  Moreover, we introduce obius
    transformations and study their geometric mapping properties.

    Chapter 2 is devoted to the study of the modular group from an
    algebraic and geometric point of view. We introduce the canonical
    fundamental region which gives rise to the modular tessellation of
    the upper half-plane. Additionally, we present a general method
    for nding fundamental regions with respect to subgroups of the
    modular group based on the concepts of 2-dimensional hyperbolic
    geometry.

    In Chapter 3 we give some concrete examples how the developed results and
    methods can be exploited for the visualization of certain mathematical
    results. Besides the visualization of function graphs of modular
    functions, a particularly nice result is the connection between
    modular transformations and continued fraction expansions.",
  paper = "Paul14.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Stou79,
  author = "Stoutemyer, David R.",
  title = {{LISP Based Symbolic Math Systems}},
  journal = "Byte Magazine",
  volume = "8",
  pages = "176-192",
  year = "1979",
```

```
link = "\url{https://ia902603.us.archive.org/30/items/byte-magazine-1979-08/1979_08_BYTE_04-08_LISP.pdf}",
comment =
  "SCRATCHPAD is a very large computer-algebra system implemented by the
  IBM Thomas J. Watson Research Center. It is available there on an IBM
  370, and it is available from other IBM corporate sites via
  telephone. Regrettably, this fine system has not yet been released to
  the public, but it is discussed here because of its novel features.

  In its entirety, the system occupies about 1600K bytes on an IBM 370
  with virtual storage, for which an additional minimum of 100 K bytes
  is recommeded for workspace. The variety of built-in transformations
  currently lies between that of REDUCE and MACSYMA. However, each of
  the three systems has features that none of the others possess, and
  one of these features may be a decisive advantage for a particular
  application. Here are some highlights of the SCRATCHPAD system:
  \begin{itemize}
  \item The system provides single-precision floating-point arithmetic
  as well as indefinite-precision rational arithmetic
  \item The built-in unavoidable and optional algebraic transformations
  are approximately similar to those of MACSYMA.
  \item The built-in exponential, logarithmic, and trigonometric
  transformations are approximately similar to those of REDUCE.
  \item Besides built-in symbolic matrix algebra, APL like array
  operations are included, and they are even further generalized to
  permit symbolic operations of nonhomogeneous arrays and on arrays
  of indefinite or infinite size.
  \item Symbolic differentiation and integration are built-in, with
  the latter employing the powerful Risch-Normal algorithm.
  \item There is a particularly elegant built-in facility for
  determining Taylor series expansions.
  \item There is a built-in SOLVE function capable of determining the
  exact solution to a system of linear equations.
  \item There is a powerful pattern-matching facility which serves as
  the primary mechanism for user level extensions. The associated syntax
  is at a very high level, being the closest of all computer algebra
  systems to the declarative, nonprocedural notation of mathematics.
  To implement the trigonometric multiple-angle expansions, we can
  merely enter the rewrite rules:
  \[cos(n*x) == 2*cos(x)*cos((n-1)*x)-cos((n-2)*x), n{\rm\ in\ }
  (2,3,...), x{\rm\ arb}\]
  \[sin(n*x) == 2*cos(x)*sin((n-1)*x)-sin((n-2)*x), n{\rm\ in\ }
  (2,3,...),x{\rm\ arb}\]
  Then, whenever we subsequently enter an expression such as
  $cos(4*b)$, the response will be a corresponding expanded
  expression such as
  \[8*cos(B) - 8*cos^2(B)+1\]
  Thus, programs resemble a collection of math formulae, much as they
  would appear in a book or article.
  \item SCRATCHPAD has a particularly powerful yet easily used mechanism
  for controlling the output format of expressions.. For example, the user
  can specify that an expression be displayed as a power series in x,
  with coefficients which are factored rational functions in b and c,
  etc. For large expressions, such fine control over the output may
  mean the difference between an important new discovery and an
```

```
    inconprehensible mess.
    \end{itemize}

    This generalized recursive format idea is so natural and effective
    that SCRATCHPAD is now absorbing the idea into an internal
    representation. A study of the polynomial additional algorithm in
    the previous section reveals that it is written to be applicable
    to any coefficient domain which has the algebraic properties of a
    {\sl ring}. The coefficients could be matrices, power-series, etc.
    That coefficient domain could in turn have yet another coefficient
    domain, and so on. With a careful modular design, packages to treat
    each of these domains can be dynamically linked together so that
    code can be shared and combined in new ways without extensive
    rewriting and duplication. Then not only the output, but also the
    internal computations can be selected most suitably for a particular
    application.

    For further information about SCRATCHPAD, contact Richard Jenks
    at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY
    10598",
  paper = "Stou79.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Vaki98,
  author = "Vakil, Ravi",
  title = {{A Beginner's Guide to Jet Bundles fromthe Point of View of
           Algebraic Geometry}},
  link = "\url{http://math.stanford.edu/~vakil/files/jets.pdf}",
  year = "1998",
  paper = "Vaki98.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wang74,
  author = "Wang, Paul S.",
  title = {{The Undecidability of the Existence of Zeros of Real Elementary
           Functions}},
  journal = "J. ACM",
  volume = "21",
  number = "4",
  pages = "586-589",
  year = "1974",
  abstract =
    "From Richardson's undecidability results, it is shown that the predicate
```

```
    ``there exists a real number $r$ such that $G(r)=0$'' is recursively
    undecidable for $G(x)$ in a class of functions which involves polynomials
    and the sine function. The deduction follows that the convergence of a
    class of improper integrals is recursively undecidable.",
  paper = "Wang74.djvu"
}
```

———-

— **axiom.bib** —

```
@article{Lang02,
  author = "Langley, Simon and Richardson, Daniel",
  title = {{What can we do with a Solution?}},
  journal = "Electronic Notes in Theoretical Computer Science",
  volume = "66",
  number = "1",
  year = "2002",
  link = "\url{http://www.elsevier.nl/locate/entcs/volume66.html}",
  abstract =
    "If $S=0$ is a system of $n$ equations and unknowns over $\mathbb{C}$
    and $S(\alpha)=0$ to what extent can we compute with the point $\alpha$?
    In particular, can we decide whether or not a polynomial expressions
    in the components of $\alpha$ with integral coefficients is zero?
    This question is considered for both algebraic and elementary systems
    of equations.",
  paper = "Lang02.pdf"
}
```

———-

— **axiom.bib** —

```
@article{John71,
  author = "Johnson, S.C.",
  title = {{On the Problem of Recognizing Zero}},
  journal = "J. ACM",
  volume = "18",
  number = "4",
  year = "1971",
  pages = "559-565",
  paper = "John71.djvu"
}
```

———-

— **axiom.bib** —

```
@article{Rich07,
  author = "Richardson, Daniel",
  title = {{How to Recognize Zero}},
```

```
  journal = "J. Symbolic Computation",
  volume = "24",
  number = "6",
  year = "2007",
  pages = "627-645",
  abstract =
    "An elementary point is a point in complex $n$ space, which is an
    isolated, non-singular solution of $n$ equations in $n$ variables,
    each equation being either or the form $p=0$, where $p$ is a polynomial
    in $\mathbb{Q}[x_1,\ldots,x_n]$, or of the form $x_j=e^{x_i}=0$. An
    elementary number is the polynomial image of an elementary point. In
    this article a semi-algorithm is given to decide whether or not a
    given elementary number is zero. It is proved that this semi-algorithm
    is an algorithm, i.e. that it always terminates, unless it is given
    a problem containing a counter example to Schanuel's conjecture.",
  paper = "Rich07.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Hurx00,
  author = "Hur, Namhyun and Davenport, James H.",
  title = {{An exact real algebraic arithmetic with equality determination}},
  booktitle = "Proc. ISSAC 2000",
  series = "ISSAC '00",
  pages = "169-174",
  year = "2000",
  abstract =
    "We describe a new arithmetic model for real algebraic numbers with
    an exact equality determination. The model represents a real algebraic
    number as a pair of an arbitrary precision numerical value and a
    symbolic expression. For the numerical part we currently (another
    representation could be used) use the dyadic exact real number and
    for the symbolic part we use a square-free polynomial for the real
    algebraic number. In this model we show that we can decide exactly
    the equality of real algebraic numbers.",
  paper = "Hurx00.djvu"
}
```

———-

— **axiom.bib** —

```
@article{Chow99,
  author = "Chow, Timothy Y.",
  title = {{What is a closed-form number?}},
  journal = "The American Mathematical Monthly",
  volume = "106",
  number = "5",
  pages = "440-448",
```

```
  year = "1999",
  paper = "Chowxx.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Yapx00,
  author = "Yap, Chee-Keng",
  title = {{Fundamental Problems in Algorithmic Algebra}},
  publisher = "John Wiley and Sons",
  year = "2000",
  link = "\url{http://www.csie.nuk.edu.tw/~cychen/gcd/Fundamental%20Problems%20in%20Algorithmic%20Algebra.pdf}",
  abstract =
    "The author shows an interesting way to write arithmetic operations.
    Given
    \[z(x,y)=\frac{axy+bx+cy+d}
                  {a^{\prime}xy+b^{\prime}x+c^{\prime}y+d^{\prime}}\]
    we call the numerical constants $a,b,\ldots,c^{\prime},d^{\prime}$
    {\sl state variables} and use the compact notation
    \[z(x,y)=\frac{(a,b,c,d)}{(a^{\prime},b^{\prime},c^{\prime},d^{\prime})}
    {x \choose y}\]
    The arithmetic operations can be recovered by suitable choices for
    the state variables:
    \[x+y=\frac{(0,1,1,0)}{(0,0,0,1)}{x \choose y}\]
    \[x-y=\frac{(0,1,-1,0)}{(0,0,0,1)}{x \choose y}\]
    \[xy=\frac{(1,0,0,0)}{(0,0,0,1)}{x \choose y}\]
    \[x/y=\frac{(0,1,0,0)}{(0,0,1,0)}{x \choose y}\]
    \[\frac{ax+b}{cx+d}=\frac{(0,a,0,b)}{(0,c,0,d)}{x \choose y}\]
    and if
    \[x=q+\frac{p}{x^{\prime}}\]
    then
    \[\begin{array}{ccc}
    z(x,y)&=&\displaystyle\frac{a(q+p/x^{\prime})y+b(q+p/x^{\prime})+cy+d}
                  {a^{\prime}(q+p/x^{\prime})y+
                   b^{\prime}(q+p/x^{\prime})+
                   c^{\prime}y+d^{\prime}}\\
    &=&\displaystyle\frac{(aq+c,bq+d,ap,bp)}
             {(a^{\prime}q+c^{\prime},b^{\prime}q+d^{\prime},a^{\prime}p,
               b^{\prime}p)}{x^{\prime} \choose {y}}
    \end{array}\]",
  paper = "Yapx00.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Yap02a,
  author = "Yap, Chee-Keng",
  title = {{Problem of Algebra Lecture 0: Introduction}},
```

```
  year = "2002",
  link =
    "\url{https://people.eecs.berkeley.edu/~fateman/282/readings/yap-0.pdf}",
  paper = "Yap02a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Yap02b,
  author = "Yap, Chee-Keng",
  title = {{Lecture II: The GCD}},
  year = "2002",
  link =
    "\url{https://people.eecs.berkeley.edu/~fateman/282/readings/yap-2.pdf}",
  paper = "Yap02b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Burn00,
  author = "Burnikel, C. and Fleischer, R. and Mehlhom, K. and Schirra, S.",
  title = {{A Strong and Easily Computable Separation Bound for Arithmetic
           Expressions Involving Radicals}},
  journal = "Algorithmica",
  volume = "27",
  pages = "87-99",
  year = "2000",
  abstract =
    "We consider arithmetic expressions over operators $+$, $-$, $*$, $/$,
    and $\sqrt{k}$, with integer operands. For an expression $E$ having
    value $\eta$, a separation bound $sep(E)$ is a positive real number with
    the property that $\eta \ne 0$ implies $\vert\eta\vert \ge sep(E)$. We
    propose a new separation bound that is easy to compute and stronger
    than previous bounds.",
  paper = "Burn00.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Corl97a,
  author = "Corless, Robert M. and Jeffrey, David J.",
  title = {{The Turing Factorization of a Rectangular Matrix}},
  journal = "ACM SIGSAM Bulletin",
  volume = "31",
  number = "3",
```

```
  pages = "28-35",
  year = "1997",
  abstract =
    "The Turing factorization is a generalization of the standard LU
    factoring of a square matrix. Among other advantages, it allows us to
    meet demands that arise in a symbolic context. For a rectangular
    matrix A, the generalized factors are written PA = LDU R, where R is
    the row-echelon form of A. For matrices with symbolic entries, the LDU
    R factoring is superior to the standard reduction to row-echelon form,
    because special case information can be recorded in a natural
    way. Special interest attaches to the continuity properties of the
    factors, and it is shown that conditions for discontinuous behaviour
    can be given using the factor D. We show that this is important, for
    example, in computing the Moore-Penrose inverse of a matrix containing
    symbolic entries.We also give a separate generalization of LU
    factoring to fraction-free Gaussian elimination.",
  paper = "Corl97a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Corl97,
  author = "Corless, Robert M. and Jeffrey, David J. and Knuth, Donald E.",
  title = {{A Sequence of Series for The Lambert W Function}},
  booktitle = "Proc. ISSAC 1997",
  series = "ISSAC '97",
  pages = "197-204",
  year = "1997",
  abstract =
    "We give a uniform treatment of several series expansions for the
    Lambert $W$ function, leading to an infinite family of new series.
    We also discuss standardization, complex branches, a family of
    arbitrary-order iterative methods for computation of $W_i$, and
    give a theorem showing how to correctly solve another simple and
    frequently occurring nonlinear equation in terms of $W$ and the
    unwinding number",
  paper = "Corl97.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Corl96,
  author = "Corless, Robert M. and Jeffrey, David J.",
  title = {{Editor's Corner: The Unwinding Number}},
  journal = "SIGSAM Bulletin",
  volume = "30",
  number = "1",
  issue = "115",
```

```
  pages = "28-35",
  year = "1996",
  abstract =
    "From the Oxford English Dictionary we find that {\sl to unwind}
    can mean ``to become free from a convoluted state''. Further down
    we find the quationation ``The solutoin of all knots, and unwinding
    of all intricacies'', from H. Brooke (The Fool of Quality, 1809).
    While we do not promise that the unwinding number, defined below,
    will solve {\sl all} intricacies, we do show that it may help for
    quite a few problems.",
  paper = "Corl96.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Corl98,
  author = "Corless, Robert M. and Jeffrey, David J.",
  title = {{Graphing Elementary Riemann Surfaces}},
  journal = "SIGSAM Bulletin",
  volume = "32",
  number = "1",
  pages = "11-17",
  year = "1998",
  abstract =
    "This paper discusses one of the prettiest pieces of elementary
    mathematics or computer algebra, that we have ever had the pleasure
    to learn. The tricks that we discuss here are certainly ``well-known''
    (that is, in the literature), but we didn't know them until recently,
    and none of our immediate colleagues knew them either. Therefore we
    believe that it is useful to publicize them further. We hope that
    you find these ideas as pleasant and useful as we do.",
  paper = "Corl98.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wang93,
  author = "Wang, Dongming",
  title = {{An Elimination Method for Polynomial Systems}},
  journal = "J. Symbolic Computation",
  volume = "16",
  number = "2",
  pages = "83-114",
  year = "1993",
  abstract =
    "We present an elimination method for polynomial systems, in the form
    of three main algorithms. For any given system [$\mathbb{P}$,$\mathbb{Q}$]
    of two sets of multivariate polynomials, one of the algorithms computes a
```

```
    sequence of triangular forms $\mathbb{T}_1,\ldots,\mathbb{T}_e$ and
    polynomial sets $\mathbb{U}_1,\ldots,\mathbb{U}_e$ such that
    Zero($\mathbb{P}$/$\mathbb{Q}$)
    $= \cup_{i=1}^e {\rm\ Zero}(\mathbb{T}_i/\mathbb{U}_i)$,
    where Zero($\mathbb{P}$/$\mathbb{Q}$) denotes the set of common zeros of
    the polynomials in $\mathbb{P}$ which are not zeros of any polynomial in
    $\mathbb{Q}$, and similarly for Zero($\mathbb{T}_i$/$\mathbb{U}_i$).
    The two other algorithms compute the same zero decomposition but with nicer
    properties such as Zero$(\mathbb{T}_i/\mathbb{U}_i) \ne 0$ for each $i$.
    One of them, for which the computed triangular systems
    [$\mathbb{T}_i$, $\mathbb{U}_i$] possess the projection property, provides
    a quantifier elimination procedure for algebraically closed fields.
    For the other, the computed triangular forms $\mathbb{T}_i$ are
    irreducible. The relationship between our method and some existing
    elimination methods is explained. Experimental data for a set of test
    examples by a draft implementation of the method are provided, and show
    that the efficiency of our method is comparable with that of some
    well-known methods. A few encouraging examples are given in detail for
    illustration.",
  paper = "Wang93.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wang94,
  author = "Wang, Dongming",
  title = {{Differentiation and Integration of Indefinite Summations with
           Respect to Indexed Variables - Some Rules and Applications}},
  journal = "J. Symbolic Computation",
  volume = "18",
  number = "3",
  pages = "249-263",
  year = "1994",
  abstract =
    "In this paper we present some rules for the differentiation and
    integration of expressions involving indefinite summations with
    respect to indexed variables which have not yet been taken into
    account of current computer algebra systems. These rules, together
    with several others, have been implemented in MACSYMA and MAPLE as a
    toolkit for manipulating indefinite summations. We discuss some
    implementation issues and report our experiments with a set of typical
    examples. The present work is motivated by our investigation in the
    computer-aided analysis and derivation of artificial neural systems.
    The application of our rules to this subject is briefly explained.",
  paper = "Wang94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wang95a,
  author = "Wang, Dongming",
  title = {{A Method for Proving Theorems in Differential Geometry and
           Mechanics}},
  journal = "J. Universal Computer Science",
  volume = "1",
  number = "9",
  pages = "658-673",
  year = "1995",
  link = "\url{http://www.jucs.org/jucs\_1\_9/a\_method\_for\_proving}",
  abstract =
    "A zero decomposition algorithm is presented and used to devise a
    method for proving theorems automatically in differential geometry and
    mechanics. The method has been implemented and its practical
    efficiency is demonstrated by several non-trivial examples including
    Bertrand s theorem, Schell s theorem and Kepler-Newton s laws.",
  paper = "Wang95a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Wang98,
  author = "Wang, Dongming",
  title = {{Decomposing Polynomial Systems into Simple Systems}},
  volume = "25",
  number = "3",
  pages = "295-314",
  year = "1998",
  abstract =
    "A simple system is a pair of multivariate polynomial sets (one set
    for equations and the other for inequations) ordered in triangular
    form, in which every polynomial is squarefree and has non-vanishing
    leading coefficient with respect to its leading variable. This paper
    presents a method that decomposes any pair of polynomial sets into
    finitely many simple systems with an associated zero decomposition.
    The method employs top-down elimination with splitting and the
    formation of subresultant regular subchains as basic operation.",
  paper = "Wang98.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wang99,
  author = "Wang, Dongming",
  title = {{Polynomial Systems from Certain Differential Equations}},
  journal = "J. Symbolic Computation",
  volume = "28",
  number = "1-2",
```

```
  pages = "303-315",
  year = "1999",
  abstract =
    "In this paper, combined elimination techniques are applied to
    establish relations among center conditions for certain cubic
    differential systems initially investigated by Kukles in 1944. The
    obtained relations clarify recent rediscoveries of some known
    conditions of Cherkas. The computational difficulties of establishing
    the complete center conditions for Kukles system, a problem that is
    still open, are illustrated by interactive elimination. Some generated
    polynomial systems that need be solved are made available
    electronically for other developers to test elimination algorithms and
    their implementations.",
  paper = "Wang99.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wang00,
  author = "Wang, Dongming",
  title = {{Computing Triangular Systems and Regular Systems}},
  journal = "J. Symbolic Computation",
  volume = "30",
  number = "2",
  pages = "221-236",
  year = "2000",
  abstract =
    "A previous algorithm of computing simple systems is modified and
    extended to compute triangular systems and regular systems from any
    given polynomial system. The resulting algorithms, based on the
    computation of subresultant regular subchains, have a simple structure
    and are efficient in practice. Preliminary experiments indicate that
    they perform at least as well as some of the known algorithms. Several
    properties about regular systems are also proved.",
  paper = "Wang00.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Wang04,
  author = "Wang, Dongming",
  title = {{A simple method for implicitizing rational curves and surfaces}},
  journal = "J. Symbolic Computation",
  volume = "38",
  number = "1",
  pages = "899-914",
  year = "2004",
  abstract =
```

```
    "This paper presents a simple method for converting rational
    parametric equations of curves and surfaces into implicit
    equations. The method proceeds via writing out the implicit polynomial
    $F$ of estimated degree with indeterminate coefficients $u_i$,
    substituting the rational expressions for the given parametric curve
    or surface into $F$ to yield a rational expression $g/h$ in the
    parameter $s$ (or $s$ and $t$), equating the coefficients of $g$ in
    terms of $s$ (and $t$) to 0 to generate a sparse, partially triangular
    system of linear equations in $u_i$ with constant coefficients, and
    finally solving the linear system for ui. If a nontrivial solution is
    found, then an implicit polynomial is obtained; otherwise, one repeats
    the same process, increasing the degree of $F$. Our experiments show
    that this simple method is efficient. It performs particularly well in
    the presence of base points and may detect the dependency of
    parameters incidentally.",
  paper = "Wang04.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Brad92,
  author = "Bradford, Russell",
  title = {{Algebraic Simplification of Multiple-Valued Functions}},
  booktitle = "Proc. DISCO 92",
  series = "Lecture Notes in Computer Science 721",
  year = "1992",
  abstract =
    "Many current algebra systems have a lax attitude to the
    simplification of expressions involving functions like log and
    $\sqrt{}$, leading the the ability to ‘‘prove’’ equalities like $-1=1$
    in such systems. In fact, only a little elementary arithmetic is
    needed to devise what the correct simplification should be. We detail
    some of these simplification rules, and outline a method for their
    incorporation into an algebra system.",
  paper = "Brad92.djvu"
}
```

———-

— **axiom.bib** —

```
@misc{Kais09,
  author = "Kaisler, Stephen H. and Madey, Gregory",
  title = {{Complex Adaptive Systems: Emergence and Self-Organization}},
  year = "2009",
  institution = "University of Notre Dame",
  comment = "source for Sweeney.eps",
  link = "\url{http://www3.nd.edu/~gmadey/Activities/CAS-Briefing.pdf}"
}
```

———-

— **axiom.bib** —

```
@article{Kalm01,
  author = "Kalman, Dan",
  title = {{A Generalized Logarithm for Exponential-Linear Equations}},
  journal = "The College Mathematics Journal",
  volume = "32",
  number = "1",
  year = "2001",
  abstract =
    "How do you solve the equation
    \[1.6^x = 5054.4 - 122.35*x\]
    We will refer to equations of this type, with an exponential
    expression on one side and a linear one on the other, as
    {\sl exponential-linear} equations. Numerical approaches such as Newtons
    method or bisection quickly lead to accurate approximate solutions of
    exponential-linear equations. But in terms of the elementary functions
    of calculus and college algebra, there is no analytic solution.

    One approach to remedying this situation is to introduce a special
    function designed to solve exponential-linear equations. Quadratic
    equations, by way of analogy, are solvable in terms of the special
    function $\sqrt{x}$ , which in turn is simply the inverse of a very
    special and simple quadratic function. Similarly, exponential
    equations are solvable in terms of the natural logarithm {\sl log},
    and that too is the inverse of a very special function. So it is
    reasonable to ask whether there is a special function in terms of
    which exponential-linear equations might be solved. Furthermore, an
    obvious strategy for finding such a function is to invert some simple
    function connected with exponential-linear equations.

    This line of thinking proves to be immediately successful, and leads
    to a function I call {\sl glog} (pronounced {\sl gee-log}) which is a
    kind of generalized logarithm. As intended, glog can be used to solve
    exponential-linear equations. But that is by no means all it is good
    for. For example, with glog you can write a closed-form expression for
    the iterated exponential ($x^{x^{x^.}}$), and solve $x + y = x^y$ for
    $y$. The glog function is also closely related to another special
    function, called the Lambert $W$ function in [3] and [6], whose study
    dates to work of Lambert in 1758 and of Euler in 1777. Interesting
    questions about glog arise at every turn, from symbolic integration,
    to inequalities and estimation, to numerical computation. Elaborating
    these points is the goal of this paper.",
  paper = "Kalm01.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Dewa91,
```

```
  author = "Dewar, Michael C.",
  title = {{Interfacing algebraic and numeric computation}},
  year = "1991",
  school = "University of Bath, UK, England",
  paper = "Dewa91.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Fate92,
  author = "Fateman, Richard J. and Einwohner, Theodore H.",
  title = {{A Proposal for Automated Integral Tables (Work in Progress)}},
  year = "1992",
  link = "\url{http://people.eecs.berkeley.edu/~fateman/papers/intable.pdf}",
  abstract =
    "One of the long-term general goals of algebraic manipulation systems
    has been the automation of difficult or tedious, yet common, symbolic
    mathematical operations. Prominent amount these has been symbolic
    integration. Although some effective algorithms have been devised for
    integration expecially for those problems solvable in terms of
    elementary functions and a few additional special functions, the vast
    majority of entires in large tables of indefinite and definite
    integrals remain out of reach of current machine algorithms. We
    propose techniques for introducing the information in such tables to
    computers, extending such tables, and measuring the success of such
    automation.

    Similar tabular data concerning simplifications, summation identities,
    and similar formulas could also be treated by some of the same
    techniques.",
  paper = "Fate92.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Atha03,
  author = "Athale, Rahul Ramesh and Diaz, Glauco Alfredo Lopez",
  title = {{Explaining Schwarz's LODEF algorithm with examples}},
  link = "\url{http://www.risc.jku.at/publications/download/risc\_1539/03-11.ps.gz}",
  year = "2003",
  abstract =
    "To every linear homogeneous ordinary differential equation
    \[L(y)=y^{(n)}+a_1y^{(n-1)}+\cdots+a_{n-1}y^{\prime}+a_ny=0\]
    one can associate the linear operator:
    \[L(D)[y]=(D^n+a_1D^{n-1}+\cdots+a_{n-1}D+a_n)[y]\]
    Here, $D^i$ is another notation for the $i$-th derivative of $y$,
    and the coefficients $a_i$ belong to a differential field $K$.
    Such an operator is called a linear homogeneous differential operator
```

```
    either over $K$, or with coefficients in $K$. Linear homogeneous
    differential operators over $K$ form a ring under the usual addition
    of operators and composition as multiplication.

    {\bf Problem}: Decompose $L$ as a product of operators of lower degree
    in $D$. We allow expressions algebraic over $K$.",
  paper = "Atha03.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Chou89,
  author = "Chou, Shang-Ching and Gao, Xiao-Shan",
  title = {{A Collection of 120 Computer Solved Geometry Problems in
           Mechanical Formula Derivation}},
  institution = "University of Texas, Austin",
  link = "\url{http://www.cs.utexas.edu/ftp/techreports/tr89-22.pdf}",
  type = "technical report",
  number = "tr-89-22",
  year = "1989",
  abstract =
    "This is a collection of 120 geometric problems mechanically solved by
    a program based on the methods introduced by us. Researchers can use
    this collection to experiment with their methods/programs similar to
    ours. It consists of two parts: the exact specification of the input
    to our program and a collection of 120 examples. A typical example
    consists of an informal description of the geometric problem, the
    input to the program which is the exact specification of the problem,
    the result of the problem, and a diagram.",
  paper = "Chou89.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Chou94,
  author = "Chou, Shang-Ching and Gao, Xiao-Shan and Zhang, Jing-Zhong",
  title = {{Machine Proofs in Geometry: Automated Production of Readable
           Proofs for Geometry Theorems}},
  publisher = "World Scientific",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.374.11778}",
  year = "1994",
  paper = "Chou94.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Stur00,
  author = "Sturmfels, Bernd",
  title = {{Solving Systems of Polynomial Equations}},
  link = "\url{https://math.berkeley.edu/~bernd/cbms.pdf}",
  year = "2000",
  abstract =
    "One of the most classical problems of mathematics is to solve systems
    of polynomial equations in several unknowns. Today, polynomial
    models are ubiquitous and widely applied across the sciences. They
    arise in robotics, coding theory, optimization, mathematical
    biology, computer vision, game theory, statistics, machine learning,
    control theory, and numerous other areas.  The set of solutions to a
    system of polynomial equations is an algebraic variety, the basic
    object of algebraic geometry. The algorithmic study of algebraic
    varieties is the central theme of computational algebraic
    geometry. Exciting recent developments in symbolic algebra and
    numerical software for geometric calculations have revolutionized
    the field, making formerly inaccessible problems tractable, and
    providing fertile ground for experimentation and conjecture.

    The first half of this book furnishes an introduction and represents a
    snapshot of the state of the art regarding systems of polynomial
    equations.  Afficionados of the well-known text books by Cox, Little,
    and OShea will find familiar themes in the first five chapters:
    polynomials in one variable, Groebner bases of zero-dimensional
    ideals, Newton polytopes and Bernsteins Theorem, multidimensional
    resultants, and primary decomposition.

    The second half of this book explores polynomial equations from a
    variety of novel and perhaps unexpected angles. Interdisciplinary
    connections are introduced, highlights of current research are
    discussed, and the authors hopes for future algorithms are
    outlined. The topics in these chapters include computation of Nash
    equilibria in game theory, semidefinite programming and the real
    Nullstellensatz, the algebraic geometry of statistical models, the
    piecewise-linear geometry of valuations and amoebas, and the
    Ehrenpreis-Palamodov theorem on linear partial differential equations
    with constant coefficients.

    Throughout the text, there are many hands-on examples and exercises,
    including short but complete sessions in the software systems maple,
    matlab, Macaulay 2, Singular, PHC, and SOStools . These examples
    will be particularly useful for readers with zero background in
    algebraic geometry or commutative algebra. Within minutes, anyone can
    learn how to type in polynomial equations and actually see some
    meaningful results on the computer screen.",
  paper = "Stur00.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{NTCI16,
  author = "NTCIR",
  title = {{Axiom (computer algebra system)}},
  link = "\url{http://ntcir11-wmc.nii.ac.jp/index.php/Axiom\_(computer_algebra_system)}",
  year = "2016",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Vers16,
  author = "Verstraete, Jacques",
  title = {{Combinatorial Calculus of Formal Power Series}},
  comment = "264A Lecture B",
  link = "\url{http://www.math.ucsd.edu/~jverstra/264A-LECTUREB.pdf}",
  paper = "Vers16.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Koep92,
  author = "Koepf, Wolfram",
  title = {{Power Series in Computer Algebra}},
  journal = "J. Symbolic Computation",
  volume = "13",
  pages = "581-603",
  year = "1992",
  abstract =
    "Formal power series (FPS) of the form
    $\sum_{k=0}^{\infty}{a_k(x-x_0)^k}$ are important in calculus and
    complex analysis.  In some Computer Algebra Systems (CASs) it is
    possible to define an FPS by direct or recursive definition of its
    coefficients.  Since some operations cannot be directly supported
    within the FPS domain, some systems generally convert FPS to finite
    truncated power series (TPS) for operations such as addition,
    multiplication, division, inversion and formal substitution.  This
    results in a substantial loss of information.  Since a goal of
    Computer Algebra is - in contrast to numerical programming - to work
    with formal objects and preserve such symbolic information, CAS should
    be able to use FPS when possible.

    There is a one-to-one correspondence between FPS with positive radius
    of convergence and corresponding analytic functions.  It should be
    possible to automate conversion between these forms.  Among CASs
    only MACSYMA provides a procedure {\tt powerseries} to calculate FPS from
    analytic expressions in certain special cases, but this is rather
    limited.
```

```
    Here we give an algorithmic approach for computing an FPS for a
    function from a very rich family of functions including all of the
    most prominent ones that can be found in mathematical dictionaries
    except those where the general coefficient depends on the Bernoulli,
    Euler, or Eulerian numbers.  The algorithm has been implemented by the
    author and A.  Rennoch in the CAS MATHEMATICA, and by D.  Gruntz in
    MAPLE.

    Moreover, the same algorithm can sometimes be reversed to calculate a
    function that corresponds to a given FPS, in those cases when a
    certain type of ordinary differential equation can be solved.",
  paper = "Koep92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Asla96,
  author = "Aslaksen, Helmer",
  title = {{Multiple-valued complex functions and computer algebra}},
  journal = "SIGSAM Bulletin",
  volume = "30",
  number = "2",
  year = "1996",
  pages = "12-20",
  link = "\url{http://www.math.nus.edu.sg/aslaksen/papers/cacas.pdf}",
  abstract =
    "I recently taught a course on complex analysis. That forced me to
    think more carefully about branches. Being interested in computer
    algebra, it was only natural that I wanted to see how such programs
    dealt with these problems. I was also inspired by a paper by
    Stoutemyer.

    While programs like Derive, Maple, Mathematica and Reduce are very
    powerful, they also have their fair share of problems. In particular,
    branches are somewhat of an Achilles' heel for them. As is well-known,
    the complex logarithm function is properly defined as a
    multiple-valued function. And since the general power and exponential
    functions are defined in terms of the logarithm function, they are
    also multiple-valued. But for actual computations, we need to make
    them single valued, which we do by choosing a branch. In Section 2, we
    will consider some transformation rules for branches of
    multiple-valued complex functions in painstaking detail.

    The purpose of this short article is not to do a comprehensive
    comparative study of different computer algebra systems. My goal is
    simply to make the readers aware of some of the problems, and to
    encourage the readers to sit down and experiment with their favourite
    programs.",
  paper = "Asla96.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Tonixx,
  author = "Tonisson, Eno",
  title = {{Branch Completeness in School Mathematics and in Computer Algebra
           Systems}},
  journal = "The Electronic Journal of Mathematics and Technology",
  volume = "1",
  number = "1",
  year = "2000",
  issn = "1933-2823",
  link = "\url{https://php.radford.edu/~ejmt/deliveryBoy.php?paper=eJMT_v1n3p5}",
  abstract =
    "In many cases when solving school algebra problems (e.g. simplifying
    an expression, solving an equation), the solution is separable into
    branches in some manner. The paper describes some approaches to
    branches that are used in school textbooks and computer algebra
    systems and compares them with mathematically branch-complete
    solutions. It tries to identify possible reasons behind different
    approaches and also indicate some ideas how such differences could be
    explained to the students.",
  paper = "Tonixx.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Stou91,
  author = "Stoutemyer, David R.",
  title = {{Crimes and misdemeanors in the computer algebra trade}},
  journal = "Notices of the American Mathematical Society",
  volume = "38",
  number = "7",
  pages = "778-785",
  year = "1991"
}
```

———-

— **axiom.bib** —

```
@misc{Sang10a,
  author = "Sangwin, Chris",
  title = {{Intriguing Integrals: Part I and II}},
  year = "2010",
  link = "\url{https://plus.maths.org/issue54/features/sangwin/2pdf/index.html/op.pdf}",
  paper = "Sang10a.pdf",
}
```

———-

— **axiom.bib** —

```
@misc{Sang10b,
  author = "Sangwin, Chris",
  title = {{Intriguing Integrals: Part I and II}},
  year = "2010",
  link = "\url{https://plus.maths.org/issue54/features/sangwin2/2pdf/index.html/op.pdf}",
  paper = "Sang10b.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Evanxx,
  author = "Evans, Brian",
  title = {{History of CA Systems}},
  link = "\url{http://felix.unife.it/Root/d-Mathematics/d-The-mathematician/d-History-of-mathematics/t-History-o
  paper = "Evanxx.txt"
}
```

———-

— **axiom.bib** —

```
@misc{Apel94,
  author = "Apel, Joachim and Klaus, Uwe",
  title = {{Representing Polynomials in Computer Algebra Systems}},
  year = "1994",
  abstract =
    "There are discussed implementational aspects of the special-purpose
    computer algebra system FELIX designed for computations in
    constructive algebra. In particular, data types developed for the
    representation of and computation with commutative and non-commuative
    polynomials are described. Furthermore, comparison of time and memory
    requirements of different polynomial representations are reported.",
  paper = "Apel94.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Juds15,
  author = "Judson, Thomas W.",
  title = {{Abstract Algebra: Theory and Applications}},
  year = "2015",
  link = "\url{http://abstract.ups.edu/aata/colophon-1.html}",
  publisher = "Website"
```

```
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt83,
  author = "Kaltofen, E.",
  title = {{On the complexity of finding short vectors in integer lattices}},
  booktitle = "Proc. EUROCAL '83",
  series = "Lect. Notes Comput. Sci.",
  year = "1983",
  volume = "162",
  pages = "236--244",
  publisher = "Springer-Verlag",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/83/Ka83_eurocal.pdf}",
  paper = "Kalt83.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt85,
  author = "Kaltofen, E.",
  title = {{Effective {Hilbert} Irreducibility}},
  booktitle = "Proc. EUROSAM '84",
  pages = "275--284",
  year = "1985",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/85/Ka85_infcontr.ps.gz}",
  paper = "Kalt85.ps"
}
```

———-

— **axiom.bib** —

```
@TechReport{Kalt85c,
  author = "Kaltofen, E.",
  title = {{Sparse Hensel lifting}},
  institution = "RPI",
  address = "Dept. Comput. Sci., Troy, N. Y.",
  year = "1985",
  number = "85-12",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/85/Ka85_techrep.pdf}",
  paper = "Kalt85c.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt85d,
  author = "Kaltofen, E.",
  title = {{Sparse Hensel lifting}},
  booktitle = "EUROCAL 85 European Conf. Comput. Algebra Proc. Vol. 2",
  pages = "4--17",
  year = "1985",
  link =
  "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/85/Ka85_eurocal.pdf}",
  abstract =
    "A new algorithm is introduced which computes the multivariate
    leading coefficients of polynomial factors from their univariate
    images. This algorithm is incorporated into a sparse Hensel
    lifting scheme and only requires the factorization of a single
    univariate image. The algorithm also provides the content of the
    input polynomial in the main variable as a by-product. We show how
    we can take advantage of this property when coputing the GCD of
    multivariate polynomials by sparse Hensel lifting.",
  paper = "Kalt85d.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Mill88,
  author = "Miller, G.L. and Ramachandran, V. and Kaltofen, E.",
  title = {{Efficient parallel evaluation of straight-line code and
           arithmetic circuits}},
  journal = "SIAM J. Comput.",
  year = "1988",
  volume = "17",
  number = "4",
  pages = "687--695",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/88/MRK88.pdf}",
  paper = "Mill88.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Greg88,
  author = "Gregory, B.; Kaltofen, E.",
  title = {{Analysis of the binary complexity of asymptotically fast
           algorithms for linear system solving}},
  journal = "SIGSAM Bulletin",
  year = "1988",
  month = "April",
  volume = "22",
  number = "2",
  pages = "41--49",
```

```
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/88/GrKa88.pdf}",
  paper = "Grey88.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt89a,
  author = "Kaltofen, E.; Rolletschek, H.",
  title = {{Computing greatest common divisors and factorizations in
           quadratic number fields}},
  journal = "Math. Comput.",
  year = "1989",
  volume = "53",
  number = "188",
  pages = "697--720",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/89/KaRo89.pdf}",
  paper = "Kalt89a.pdf"
}
```

———-

— **axiom.bib** —

```
@Unpublished{Kalt89b,
  author = "Kaltofen, E.",
  title = {{Processor efficient parallel computation of polynomial greatest
           common divisors}},
  note = "unknown",
  year = "1989",
  month = "July",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/89/Ka89_gcd.ps.gz}",
  paper = "Kalt89b.ps"
}
```

———-

— **axiom.bib** —

```
@TechReport{Kalt89c,
  author = "Kaltofen, E.",
  title = {{Parallel Algebraic Algorithm Design}},
  institution = "RPI",
  address = "Dept. Comput. Sci., Troy, New York",
  year = "1989",
  month = "July",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/89/Ka89_parallel.ps.gz}",
  paper = "Kalt89c.ps"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Cann89,
  author = "Canny, J. and Kaltofen, E. and Yagati, Lakshman",
  title = {{Solving systems of non-linear polynomial equations faster}},
  booktitle = "Proc. 1989 Internat. Symp. Symbolic Algebraic Comput.",
  pages = "121--128",
  year = "1989",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/89/CKL89.pdf}",
  paper = "Cann89.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt90b,
  author = "Kaltofen, E.",
  title = {{Computing the irreducible real factors and components of an
           algebraic curve}},
  journal = "Applic. Algebra Engin. Commun. Comput.",
  year = "1990",
  volume = "1",
  number = "2",
  pages = "135--148",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/90/Ka90_aaecc.pdf}",
  paper = "Kalt90b.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt90d,
  author = "Kaltofen, E.; Trager, B.",
  title = {{Computing with polynomials given by black boxes for their
    evaluations: Greatest common divisors, factorization, separation of
    numerators and denominators}},
  journal = "J. Symbolic Comput.",
  year = "1990",
  volume = "9",
  number = "3",
  pages = "301--320",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/90/KaTr90.pdf}",
  paper = "Kalt90d.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt91a,
  author = "Kaltofen, E. and Singer, M.F.",
  editor = "D. V. Shirkov and V. A. Rostovtsev and V. P. Gerdt",
  title = {{Size efficient parallel algebraic circuits for partial
           derivatives}},
  booktitle =
    "IV International Conference on Computer Algebra in Physical Research",
  pages = "133--145",
  publisher = "World Scientific Publ. Co.",
  year = "1991",
  address = "Singapore",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/91/KaSi91.pdf}",
  paper = "Kalt91a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt93,
  author = "Kaltofen, E.",
  title = {{Computational Differentiation and Algebraic Complexity Theory}},
  booktitle = "Workshop Report on First Theory Institute on Computational
               Differentiation",
  editor = "C. H. Bischof and A. Griewank and P. M. Khademi",
  publisher = "Argonne National Laboratory",
  address = "Argonne, Illinois",
  series = "Tech. Rep.",
  volume = "ANL/MCS-TM-183",
  month = "December",
  year = "1993",
  pages = "28--30",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/93/Ka93_diff.pdf}",
  paper = "Kalt93.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt93b,
  author = "Kaltofen, E.",
  title = {{Direct proof of a theorem by Kalkbrener, Sweedler, and Taylor}},
  journal = "SIGSAM Bulletin",
  year = "1993",
  volume = "27",
  number = "4",
  pages = "2",
  link =
"\url{http://www.math.ncsu.edu/~kaltofen/bibliography/93/Ka93_sambull.ps.gz}",
  paper = "Kalt93b.ps"
```

```
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt94,
  author = "Kaltofen, E. and Pan, V.",
  title = {{Parallel solution of Toeplitz and Toeplitz-like linear
           systems over fields of small positive characteristic}},
  booktitle = "Proc. First Internat. Symp. Parallel Symbolic Comput.",
  pages = "225--233",
  year = "1994",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/94/KaPa94.pdf}",
  paper = "Kalt94.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Raab13,
  author = "Raab, C.G.",
  title = {{Generalization of Risch's Algorithm to Special Functions}},
  journal = "CoRR",
  volume = "abs/1305.1481",
  year = "2013",
  link = "\url{http://arxiv.org/pdf/1305.1481v1.pdf}",
  abstract =
    "Symbolic integration deals with the evaluation of integrals in closed
    form. We present an overview of Risch's algorithm including recent
    developments. The algorithms discussed are suited for both indefinite
    and definite integration. They can also be used to compute linear
    relations among integrals and to find identities for special functions
    given by parameter integrals. The aim of this presentation is twofold:
    to introduce the reader to some basic ideas of differential algebra in
    the context of integration and to raise awareness in the physics
    community of computer algebra algorithms for indefinite and definite
    integration. ",
  paper = "Raab13.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Sama95,
  author = "Samadani, M. and Kaltofen, E.",
  title = {{Prediction based task scheduling in distributed computing}},
  booktitle = "Languages, Compilers and Run-Time Systems for Scalable
               Computers",
```

```
  editor = "B. K. Szymanski and B. Sinharoy",
  publisher = "Kluwer Academic Publ.",
  address = "Boston",
  pages = "317--320",
  year = "1996",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/95/SaKa95_poster.ps.gz}",
  paper = "Sama95.ps"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Erli96,
  author = "Erlingsson, U. and Kaltofen, E. and Musser, D.",
  title = {{Generic {Gram}-{Schmidt} Orthogonalization by Exact Division}},
  booktitle = "Proc. 1996 Internat. Symp. Symbolic Algebraic Comput.",
  year = "1996",
  pages = "275--282",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/96/EKM96.pdf}",
  paper = "Erli96.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt96,
 author = "Kaltofen, E. and Lobo, A.",
 title = {{On rank properties of {Toeplitz} matrices over finite fields}},
  booktitle = "Proc. 1996 Internat. Symp. Symbolic Algebraic Comput.",
  year = "1996",
  pages = "241--249",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/96/KaLo96_issac.pdf}",
  paper = "Kalt96.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt97,
  author = "Kaltofen, E.",
  title = {{Teaching Computational Abstract Algebra}},
  journal = "Journal of Symbolic Computation",
  volume = "23",
  number = "5-6",
  pages = "503--515",
  year = "1997",
  note = "Special issue on education, L. Lambe, editor.",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/97/Ka97_jsc.pdf}",
```

```
  abstract = "
    We report on the contents and pedagogy of a course in abstract algebra
    that was taught with the aid of educational software developed within
    the Mathematica system. We describe the topics covered and the
    didactical use of the corresponding Mathematica packages, as well as
    draw conclusions for future such courses from the students' comments
    and our own experience.",
  paper = "Kalt97.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@Unpublished{Hitz97,
  author = "Hitz, M. A. and Kaltofen, E.",
  title = {{The {Kharitonov} theorem and its applications in symbolic
           mathematical computation}},
  note = "unknown",
  year = "1997",
  month = "May",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/97/HiKa97_kharit.pdf}",
  paper = "Hitz97.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Bern99,
  author = "Bernardin, L. and Char, B. and Kaltofen, E.",
  title = {{Symbolic Computation in {Java}: an Appraisement}},
  booktitle = "Proc. 1999 Internat. Symp. Symbolic Algebraic Comput.",
  year = "1999",
  pages = "237--244",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/99/BCK99.pdf}",
  paper = "Bern99.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt02,
  author = "Kaltofen, Erich and McLean, Michael and Norris, Larry",
  title = {{`{Using} {Maple} to Grade {Maple}' Assessment Software from
            {North Carolina State University}}},
  booktitle = "Proceedings 2002 Maple Workshop",
  year = "2002",
  publisher = "Waterloo Maple Inc.",
```

```
  address = "Waterloo, Canada",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/02/KMN02.pdf}",
  paper = "Kalt02.pdf"
}
```

———-

— **axiom.bib** —

```
@Book{Grab03,
  editor = "Grabmeier, J. and Kaltofen, E. and Weispfenning, V.",
  title = {{Computer Algebra Handbook}},
  publisher = "Springer-Verlag",
  year = "2003",
  note =  "637 + xx~pages + CD-ROM. Includes E. Kaltofen and V. Weispfenning
    \S1.4 Computer algebra -- impact on research, pages 4--6;
    E. Kaltofen
    \S2.2.3 Absolute factorization of polynomials, page 26;
    E. Kaltofen and B. D. Saunders
    \S2.3.1 Linear systems, pages 36--38;
    R. M. Corless, E. Kaltofen and S. M. Watt
    \S2.12.3 Hybrid methods, pages 112--125;
    E. Kaltofen
    \S4.2.17 FoxBox and other blackbox systems, pages 383--385.",
  isbn = "3-540-65466-6",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/01/symnum.pdf}",
  paper = "Grab03.pdf",
  keywords = "axiomref",
  beebe = "Grabmeier:2003:CAH"
}
```

———-

— **Grabmeier:2003:CAH** —

```
@Book{Grabmeier:2003:CAH,
  editor =       "Johannes Grabmeier and Erich Kaltofen and Volker
                 Weispfenning",
  title =        {{Computer algebra handbook: foundations, applications,
                 systems}},
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "xx + 637",
  year =         "2003",
  ISBN =         "3-540-65466-6",
  ISBN-13 =      "978-3-540-65466-7",
  LCCN =         "QA155.7.E4 C64954 2003",
  MRclass =      "68W30, 00B15, 68-06",
  bibdate =      "Tue Nov 22 06:00:25 MST 2005",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib;
                 z3950.loc.gov:7090/Voyager",
  note =         "Includes CD-ROM.",
  link =          "\url{http://www.springer.com/sgw/cda/frontpage/0,11855,1-102-22-1477871-0,00.html}",
```

```
  acknowledgement = ack-nhfb,
  keywords =      "Aldor; AXIOM; Derive; exact arithmetic; Macsyma;
                  Magma; Maple Mathematica; MuPAD; REDUCE; TI-92",
  subject =       "Algebra; Data processing",
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt07,
  author = "Kaltofen, Erich and Li, Bin and Sivaramakrishnan, Kartik and
            Yang, Zhengfeng and Zhi, Lihong",
  title = {{Lower bounds for approximate factorizations via semidefinite
           programming (extended abstract)}},
  year = "2007",
  booktitle =
    "SNC'07 Proc. 2007 Internat. Workshop on Symbolic-Numeric Comput.",
  pages = "203--204",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/07/KLSYZ07.pdf}",
  paper = "Kalt07.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Borw07,
  author = "Borwein, Peter and Kaltofen, Erich and Mossinghoff, Michael J.",
  title = {{Irreducible Polynomials and {Barker} Sequences}},
  journal = "{ACM} Communications in Computer Algebra",
  volume = "162",
  number = "4",
  year = "2007",
  pages = "118--121",
  month = "December",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/07/BKM07.pdf}",
  paper = "Borw07.pdf"
}
```

———-

— **axiom.bib** —

```
@Article{Kalt10,
  author = "Kaltofen, Erich and Lavin, Mark",
  title = {{Efficiently Certifying Non-Integer Powers}},
  journal = "Computational Complexity",
  year = "2010",
  volume = "19",
  number = "3",
```

```
  month = "September",
  pages = "355--366",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/09/KaLa09.pdf}",
  paper = "Kalt10.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt11,
  author = "Kaltofen, Erich L. and Nehring, Michael",
  title = {{Supersparse black box rational function interpolation}},
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'11",
  month = "June",
  year = "2011",
  pages = "177--185",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/11/KaNe11.pdf}",
  paper = "Kalt11.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Gren11a,
  author = "Grenet, Bruno and Kaltofen, Erich L. and Koiran, Pascal
            and Portier, Natacha",
  title = {{Symmetric Determinantal Representation of Weakly Skew Circuits}},
  booktitle = "Proc. 28th Internat. Symp. on Theoretical Aspects of Computer
               Science, STACS 2011",
  pages = "543--554",
  year = "2011",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/11/GKKP11.pdf}",
  paper = "Gren11a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt11a,
  author = "Kaltofen, Erich L. and Nehring, Michael and Saunders, David B.",
  title = {{Quadratic-Time Certificates in Linear Algebra}},
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'11",
  month = "June",
  year = "2011",
  pages = "171--176",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/11/KNS11.pdf}",
  paper = "Kalt11a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt11b,
  author = "Kaltofen, Erich L. and Lee, Wen-shin and Yang, Zhengfeng",
  title = {{Fast estimates of {Hankel} matrix condition numbers
           and numeric sparse interpolation}},
  booktitle = "Proc. 2011 Internat. Workshop on Symbolic-Numeric Comput.",
  month = "June",
  year = "2011",
  pages = "130--136",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/11/KLY11.pdf}",
  paper = "Kalt11b.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Guo12,
  author = "Guo, Feng and Kaltofen, Erich L. and Zhi, Lihong",
  title = {{Certificates of Impossibility of {Hilbert}-{Artin} Representations
           of a Given Degree for Definite Polynomials and Functions}},
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'12",
  month = "July",
  year = "2012",
  pages = "195--202",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/12/GKZ12.pdf}",
  paper = "Guo12.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Come12a,
  author = "Comer, Matthew T. and Kaltofen, Erich L. and Pernet, Cl{\'e}ment",
  title = {{Sparse Polynomial Interpolation and {Berlekamp}/\allowbreak
           {Massey} Algorithms That Correct Outlier Errors in Input Values}},
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'12",
  month = "July",
  year = "2012",
  pages = "138--145",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/12/CKP12.pdf}",
  paper = "Come12a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Boye13,
  author = "Boyer, Brice and Comer, Matthew T. and Kaltofen, Erich L.",
  title =  {{Sparse Polynomial Interpolation by Variable Shift in
            the Presence of Noise and Outliers in the Evaluations}},
  booktitle =
    "Proc. Tenth Asian Symposium on Computer Mathematics (ASCM 2012)",
  year = "2013",
  month = "October",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/13/BCK13.pdf}",
  paper = "Boye13.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt13b,
  author = "Kaltofen, Erich and Yang, Zhengfeng",
  title = {{Sparse multivariate function recovery from values with noise and
           outlier errors}},
  year = "2013",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'13",
  pages = "219--226",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/13/KaYa13.pdf}",
  paper = "Kalt13b.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt13c,
  author = "Kaltofen, Erich L.",
  title = {{Symbolic Computation and Complexity Theory Transcript of My Talk}},
  booktitle =
    "Proc. Tenth Asian Symposium on Computer Mathematics (ASCM 2012)",
  year = "2013",
  month = "October",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/13/Ka13.pdf}",
  paper = "Kalt13c.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt14,
  author = "Kaltofen, Erich L. and Yang, Zhengfeng",
  title = {{Sparse Multivariate Function Recovery With a High Error Rate
```

```
          in Evaluations}},
  year = "2014",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'14",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/14/KaYa14.pdf}",
  paper = "Kalt14.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kalt14a,
  author = "Kaltofen, Erich L. and Pernet, Cl{\'e}ment",
  title = {{Sparse Polynomial Interpolation Codes and Their Decoding
          Beyond Half the Minimal Distance}},
  year = "2014",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'14",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/14/KaPe14.pdf}",
  paper = "Kalt14a.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Duma14,
  author = "Dumas, Jean-Guillaume and Kaltofen, Erich L.",
  title = {{Essentially Optimal Interactive Certificates In Linear Algebra}},
  year = "2014",
  booktitle = "Internat. Symp. Symbolic Algebraic Comput. ISSAC'14",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/14/DuKa14.pdf}",
  paper = "Duma14.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Boye14,
  author = "Boyer, Brice B. and Kaltofen, Erich L.",
  title = {{Numerical Linear System Solving With Parametric Entries By
          Error Correction}},
  year = "2014",
  booktitle = "SNC'14 Proc. 2014 Int. Workshop on Symbolic-Numeric Comput.",
  link = "\url{http://www.math.ncsu.edu/~kaltofen/bibliography/14/BoKa14.pdf}",
  paper = "Boye14.pdf"
}
```

———-

1.40 Axiom Citations in the Literature

1.40.1 A

— **axiom.bib** —

```
@book{Abla96,
  author = "Ablamowicz, Rafal and Lounesto, Pertti and Para, Josep M.",
  title = {{Clifford algebras with numeric and symbolic computations}},
  publisher = "Birkhauser",
  year = "1996",
  abstract =
    "The 20 articles of this volume will be reviewed individually. They
    have been grouped into the following four sections: I) Verifying and
    falsifying conjectures (1); II) Differential geometry quantum
    mechanics, spinors and conformal group (9); III) Generalized Clifford
    algebras and number systems, projective geometry and crystallography
    (7); IV) Numerical methods in Clifford algebras (3).

    Selected electronic materials submitted by our contributors can be
    found at a Web site:
    http://www.birkhauser.com/books/ISBN/0-8176-3907-1

    These materials contain original packages, worksheets, notebooks,
    computer programs, etc., that were used in deriving results presented
    in this book.",
  keyword = "axiomref"
}
```

————-

— **axiom.bib** —

```
@misc{ACA00,
  author = "Kokol-voljc, Vlasta and Kutzler, Bernhard",
  title = {{Computer Algebra Meets Education}},
  conference = "Sessions of ACA2000",
  year = "2000",
  abstract =
    "Education has become one of the fastest growing application areas for
    computers in general and computer algebra in particular. Computer
    algebra tools such as TI-92/89, Derive, Mathematica, Maple, Axiom,
    Reduce, Macsyma, or Mupad make powerful teaching tools in mathematics,
    physics, chemistry, biology, economy.

    The goal of this session is to exchange ideas and experiences, to hear
    about classroom experiments, and to discuss all issues related to the
    use of computer algebra tools in the classroom (such as assessment,
    change of curricula, new support material, ...)",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{ACA15,
  author = "Martinez-Moro, Edgar Kotsireas, Ilias",
  title = {{21st Conference on Applications of Computer Algebra}},
  conference = "Sessions of ACA2015",
  location = "Kalamata, Greece",
  year = "2015",
  link = "\url{http://www.singacom.uva.es/ACA2015/latex/ACAproc.pdf}",
  paper = "ACA15.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{ACM89,
  author = "ACM",
  title = {{Proceedings of the ACM-SIGSAM 1989 International
           Symposium on Symbolic and Algebraic Computation, ISSAC '89}},
  year = "1989",
  isbn = "0-89791-325-6",
  link = "\url{http://doi.acm.org/10.1145/74540.74567}",
  doi = "10.1145/74540.74567",
  acmid = "74567",
  publisher = "ACM Press",
  address = "New York, NY, USA",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{ACM94,
  author = "ACM",
  title = {{Proceedings of the ACM-SIGSAM 1989 International
           Symposium on Symbolic and Algebraic Computation, ISSAC '94}},
  year = "1994",
  isbn = "0-89791-638-7",
  publisher = "ACM Press",
  address = "New York, NY, USA",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Adam99a,
  author = "Adams, A.A. and Gottliebsen, H. and Linton, S.A. and Martin, U.",
  title = {{VSDITLU: A verifiable symbolic definite integral table look-up}},
  booktitle = "Automated Deduction",
  series = "CADE 16",
  year = "1999",
  location = "Trento, Italy",
  pages = "112-126",
  link = "\url{http://www.a-cubed.info/Publications/CADE99.pdf}",
  abstract =
    "We present a verifiable symbolic definite integral table lookup: a
    system which matches a query, comprising a definite integral with
    parameters and side conditions, against an entry in a verifiable table
    and uses a call to a library of facts about the reals in the theorem
    prover PVS to aid in the transformation of the table entry into an
    answer. Our system is able to obtain correct answers in cases where
    standard techniques implemented in computer algebra systems fail. We
    present the full model of such a system as well as a description of
    our prototype implementation showing the efficacy of such a system:
    for example, the prototype is able to obtain correct answers in cases
    where computer algebra systems do not. We extend upon Fatemans
    web-based table by including parametric limits of integration and
    queries with side conditions.",
  paper = "Adam99a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Akba08,
  author = "Akbar Hussain, D.M. and Haq, Shaiq A. and Khan, Zafar Ullah
            and Ahmed, Zaki",
  title = {{Simple object oriented designed computer algebra system}},
  journal = "J. Comput. Mathods Sci. Eng.",
  volume = "8",
  number = "3",
  pages = "195-211",
  year = "2008",
  abstract =
    "Computer Algebra System (CAS) is a software program that facilitates
    symbolic mathematics. The core functionality of a typical CAS is
    manipulation of mathematical expressions in symbolic forms. The
    expressions manipulated by CAS normally include polynomials in
    multiple variables, standart trigonometric and exponential functions,
    various special functions for example gamma, zeta, Bessel, etc. and
    also arbitrary functions like derivatives, integrals, sums, and
    products of expressions. Our implementation of a CAS tool provides an
    object orienged design framework. The system is portable to other
    platforms and highly scalable. The other key features include a very
    simple and interactive user GUI support for a formula editor, making
```

```
    it a self contained system, additionally the formula editor provides a
    real-time syntax checking for expressions.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Aitk99a,
  author = "Aitken, William E. and Constable, Robert L. and
            Underwood, Judith L.",
  title = {{Metalogical frameworks. II: Developing a reflected decision
           procedure}},
  journal = "J. Autom. Reasoning",
  volume = "22",
  number = "2",
  pages = "171-221",
  year = "1999",
  link = "\url{http://www.nuprl.org/documents/Constable/MetalogicalFrameworksII.pdf}",
  abstract =
    "Proving theorems is a creative act demanding new combinations of ideas
    and on occasion new methods of argument. For this reason, theorem
    proving systems need to be extensible. The provers should also remain
    correct under extension, so there must be a secure mechanism for doing
    this. The tactic-style provers pioneered by Edinburgh LCF provide a
    very effective way to achieve secure extensions, but in such systems,
    all new methods must be reduced to tactics. This is a drawback because
    there are other useful proof generating tools such as decision
    procedures; these include, for example, algorithms which reduce a
    deduction problem, such as arithmetic provability, to a computation on
    graphs.

    The Nuprl system pioneered the combination of fixed decision
    procedures with tactics, but the issue of securely adding new ones was
    not solved. In this paper, we show how to safely include user-defined
    decision procedures in theorem provers. The idea is to prove
    properties of the procedure inside the provers logic and then invoke
    a reflection rule to connect the procedure to the system. We also show
    that using a rich underlying logic permits an abstract account of the
    approach so that the results carry over to different implementations
    and other logics.",
  paper = "Aitk99a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{America,
  author = "america.pink",
```

```
  title = {{Axiom (computer algebra system)}},
  year = "2016",
  link = "\url{http://america.pink/axiom-computer-algebra-system_526647.html}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Alad03,
  author = "Aladjev, Victor",
  title = {{Computer Algebra System Maple: A New Software Library}},
  journal = "LNCS",
  year = "2003",
  pages = "711-717",
  abstract =
    "The paper represents Maple library containing more than 400 procedures
    expanding possibilities of the Maple package of releases 6,7 and 8.
    The library is structurally organized similarly to the main Maple
    library. The process of the library installing is simple enough as a
    result of which the above library will be logically linked with the
    main Maple library, supporting access to software located in it
    equally with standard Maple software. The demo library is delivered
    free of charge at request to addresses mentioned above.",
  paper = "Alad03.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Andr87,
  author = "Andrews, George",
  title = {{Applications of Scratchpad to Problems in Special
           Functions and Combinatorics}},
  journal = "Lecture Notes in Computer Science",
  volume = "296",
  year = "1987",
  pages = "158-166",
  abstract =
    "Within the last few years, there have been numerous applications of
    computer algebra to special functions. G. Gasper (Norwestern
    University) has studied classical hypergeometric functions, and
    W. Gosper (Symbolics Inc.) has developed a large variety of
    spectacular transformation and summation techniques for MACSYMA. The
    purpose of this note is to explore some of the interface between
    computer algebra and special functions. In Section 2 we examine an
    application of MACSYMA which inadequately relied, in my opinion, on
    what was readily available in the literature on hypergeometric
    series. In Section 3 we consider classical observations on sums of
```

```
    powers of binomial coefficients. In Section 4 we consider a problem of
    D.M. Jackson wherein SCRATCHPAD and classical hypergeometric series
    interact nicely. We close with a problem inspired by work in
    statistical mechanics which leads us to questions about algorithms
    that would be useful in computer algebra applications.",
  paper = "Andr87.pdf",
  keywrods = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Andr90,
  author = "Andrews, George and Baxter, R.J.",
  title = {{SCRATCHPAD explorations for elliptic theta functions}},
  booktitle = "Computers in Mathematics",
  series = "Lecture Notes in Pure and Appl. Math 125",
  pages = "17-33",
  year = "1990",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Andr09,
  author = "Andrews, George",
  title = {{The Meaning of Ramanujan Now and for the Future}},
  journal = "Ramanujan Journal",
  volume = "20",
  pages = "257-273",
  year = "2009",
  link = "\url{http://www.personal.psu.edu/gea1/pdf/274.pdf}",
  abstract =
    "December 22, 2010 marks the 123th anniversary of Ramanujan's
    birth. In this paper we pay homage to this towering figure whose
    mathematical discoveries so affected mathematics throughout the
    twentieth century and into the twenty-first.",
  paper = "Andr09.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Anto01,
  author = "Antoine, Xavier",
  title = {{Microlocal diagonalization of strictly hyperbolic
```

```
         pseudo-differential systems and application to the design of
         radiation conditions in electromagnetism}},
  journal = "SIAM J. Appl. Math",
  volume = "61",
  number = "6",
  pages = "1877-1905",
  year = "2001",
  abstract =
    "In [Commun. Pure Appl. Math. 28, 457-478 (1975; Zbl 0332.35058)],
    M. E. Taylor described a constructive diagonalization method to
    investigate the reflection of singularities of the solution to
    first-order hyperbolic systems. According to further works initiated
    by Engquist and Majda, this approach proved to be well adapted to the
    construction of artificial boundary conditions. However, in the case
    of systems governed by pseudodifferential operators with variable
    coefficients, Taylors method involves very elaborate calculations of
    the symbols of the operators. Hence, a direct approach may be
    difficult when dealing with high-order conditions.

    This motivates the first part of this paper, where a recursive
    explicit formulation of Taylors method is stated for microlocally
    strictly hyperbolic systems. Consequently, it provides an algorithm
    leading to symbolic calculations which can be handled by a computer
    algebra system. In the second part, an application of the method is
    investigated for the construction of local radiation boundary
    conditions on arbitrarily shaped surfaces for the transverse electric
    Maxwell system. It is proved that they are of complete order, provided
    the introduction of a new symbols class well adapted to the Maxwell
    system. Next, a complete second-order condition is designed, and when
    used as an on-surface radiation condition [G. A. Kriegsmann,
    A. Taflove, and K. R. Umashankar, IEEE Trans. Antennas Propag. 35,
    153-161 (1987; Zbl 0947.78571)], it can give rise to an ultrafast
    numerical approximate solution of the scattered field.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Arna95a,
  author = "Arnault, Francois",
  title = {{Rabin-Miller primality test: Composite numbers which pass it}},
  journal = "Mathematics of  Computation",
  volume = "64",
  number = "209",
  pages = "355-361",
  year = "1995",
  link = "\url{https://www.jointmathematicsmeetings.org/mcom/1995-64-209/S0025-5718-1995-1260124-2/S0025-5718-19
  abstract =
    "The Rabin-Miller primality test is a probabilistic test which can be
    found in several algebraic computing systems (such as Pari, Maple,
    ScratchPad) because it is very easy to implement and, with a
```

```
    reasonable amount of computing, indicates whether a number is
    composite or ``probably prime'' with a very low probability of error. In
    this paper, we compute composite numbers which are strong pseudoprimes
    to several chosen bases. Because these bases are those used by the
    ScratchPad implementation of the test, we obtain, by a method which
    differs from a recent one by G. Jaeschke [ibid. 61, 915-926 (1993; Zbl
    0802.11001)], composite numbers which are found to be ``probably prime''
    by this test.",
  paper = "Arna95a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Atki95,
  author = "Atkinson, M. D. and Linton, S. A. and Walker, L. A.",
  title = {{Priority queues and multisets}},
  journal = "J. Comb",
  volume = "2",
  pages = "385-402",
  year = "1995",
  link = "\url{http://www.combinatorics.org/ojs/index.php/eljc/article/download/v2i1r24.pdf}",
  abstract =
    "A priority queue, a container data structure equipped with the
    operations insert and delete-minimum, can re-order its input in
    various ways, depending both on the input and on the sequence of
    operations used. If a given input $\sigma$ can produce a particular
    output $\tau$ then $(\sigma,\tau)$ is said to be an allowable pair. It
    is shown that allowable pairs on a fixed multiset are in one-to-one
    correspondence with certain $k$-way trees and, consequently, the
    allowable pairs can be enumerated. Algorithms are presented for
    determining the number of allowable pairs with a fixed input
    component, or with a fixed output component. Finally, generating
    functions are used to study the maximum number of output components
    with a fixed input component, and a symmetry result is derived.",
  paper = "Atki95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Augo91,
  author = "Augot, D. and  Charpin, P. and Sendrier, N.",
  title = {{The miniumum distance of some binary codes via the
           Newton's identities}},
  booktitle = "Int. Symp. on Coding Theory and Applications",
  year = "1991",
  pages = "65-73",
```

```
  isbn = "0-387-54303-1",
  abstract =
    "In this paper, we give a natural way of deciding whether a given
    cyclic code contains a word of given weight. The method is based on
    the manipulation of the locators and of the locator polynomial of a
    codeword $x$.

    Because of the dimensions of the problem, we need to use a symbolic
    computation software, like Maple or Scratchpad II. The method can be
    ineffective when the length is too large.

    The paper contains two parts: In the first part we will present the main
    definitions and properties we need.

    In the second part, we will explain how to use these properties, and, as
    illustration, we will prove the following facts:
    \begin{itemize}
    \item The dual of the BCH code of length 63 and designed distance 9
    has true minimum distance 14 (which was already known).
    \item The BCH code of length 1023 and designed distance of 253 has
    minimum distance 253.
    \item The cyclic codes of length $2^11-1$, $2^13-1$, $2^17-1$, with
    generator polynomial $m_1(x)$ and $m_7(x)$ have minimum distance 4.
    \end{itemize}",
  paper = "Augo91.pdf",
  keywords = "axiomref",
  beebe = "Augot:1991:MDS"
}
```

———-

— Augot:1991:MDS —

```
@InProceedings{Augot:1991:MDS,
  author =       "D. Augot and P. Charpin and N. Sendrier",
  title =        {{The minimum distance of some binary codes via the
                 {Newton}'s identities}},
  crossref =     "Cohen:1991:EIS",
  pages =        "65--73",
  month =        "",
  year =         "1991",
  bibdate =      "Tue Sep 17 06:41:20 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The authors propose a natural way of deciding whether
                 a given cyclic code contains a word of given weight.
                 The method is based on the manipulation of the locators
                 and of the locator polynomial of a codeword x. Because
                 of the dimensions of the problem, one needs to use
                 symbolic computation software, like Maple or Scratchpad
                 II. The method can be ineffective when the length is
                 too large. The paper is in two parts: In the first
                 part, they present the main definitions and properties
                 needed. In the second part, they explain how to use
```

```
                these properties, and, as illustration, prove the three
                following facts: the dual of the BCH code of length 63
                and designed distance 9 has true minimum distance 14
                (which was already known). The BCH code of length 1023
                and designed distance 253 has minimum distance 253. The
                cyclic codes of length 2/sup 11/, 2/sup 13/, 2/sup 17/,
                with generator polynomial m/sub 1/(x) and m/sub 7/(x)
                have minimum distance 4.",
  acknowledgement = ack-nhfb,
  affiliation =  "Paris 6 Univ., France",
  classification = "B6120B (Codes)",
  keywords =     "BCH code; Binary codes; Codeword; Cyclic codes;
                Generator polynomial; Locator polynomial; Minimum
                distance; Newton identities; Symbolic computation",
  language =     "English",
  thesaurus =    "Codes",
}
```

———-

— **ignore** —

```
\bibitem[Adams 94]{AL94}
  author = "Adams, William W. and Loustaunau, Philippe",
  title = {{An Introduction to Gr\"obner Bases}},
  year = "1994",
American Mathematical Society (1994)
  isbn = "0-8218-3804-0",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@InProceedings{Andr84,
  author = "Andrews, George E.",
  title = {{Ramanujan and SCRATCHPAD}},
  booktitle = "Proc. of 1984 MACSYM Users' Conference, July 1984",
  location = "General Electric, Schenectady, NY",
  year = "1984",
  pages = "383-408",
  keywords = "axiomref"
}
```

———-

— **Andrews:1984:RS** —

```
@InProceedings{Andrews:1984:RS,
  author =       "George E. Andrews",
  title =        {{{Ramanujan} and {SCRATCHPAD}}},
  crossref =     "Golden:1984:PMU",
```

```
  pages =        "383--??",
  year =         "1984",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.1.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— axiom.bib —

```
@InProceedings{Andr88,
  author = "Andrews, G. E.",
  title = {{Application of Scratchpad to problems in special functions and
           combinatorics}},
  booktitle = "Trends in Computer Algebra",
  publisher = "Springer",
  series = "Lecture Notes in Comp. Sci. 296",
  year = "1988",
  isbn = "3-540-18928-9",
  pages = "159-166",
  abstract =
    "Within the last few years, there have been numerous applications of
    computer algebra to special functions. G. Gasper (Northwestern
    University) has studied classical hypergeometric functions, and
    W. Gosper (Symbolics, Inc.) has developed a large variety of
    spectacular transformation and summation techniques for MACSYMA. The
    purpose of this note is to explore some of the interface between
    computer algebra and special functions. In section 2 we examine an
    application of MACSYMA which inadequately relied, in my opinion, on
    what was readily available in the literature on hypergeometric
    series. In Section 3 we consider classical observations on sums of
    powers of binomial coefficients. In Section 4 we consdier a problem of
    D.M. Jackson wherein SCRATCHPAD and classical hypergeometric series
    interact nicely. We close with a problem inspired by work in
    statistical mechanics which leads us to question about algorithsm that
    would be useful in computer algebra applications.

    In this brief survey, we have illustrated some of the uses of computer
    algebra. It might be objected that our work could well be carried out
    in almost any computer language; so why bother with SCRATCHPAD? The
    answer, of course, lies in the naturalness and simplicity of computer
    algebra approaches to these problems. Expressions like (2.2), (3.1)
    and (4.1) can be coded in SCRATCHPAD in one line exactly as they are
    written. They can then be studied with minimal thought about the
    computer and maximal concentration on what is happening. Often
    mathematical research consists of sifting low grade ore, and when such
    sifting requires ingenious programming skills it is likely not to be
    carried out.",
  keywords = "axiomref",
  beebe = "Andrews:1988:ASP"
}
```

———-

— **Andrews:1988:ASP** —

```
@InProceedings{Andrews:1988:ASP,
  author =       "G. E. Andrews",
  title =        {{Application of {Scratchpad} to problems in special
                 functions and combinatorics}},
  crossref =     "Janssen:1988:TCA",
  pages =        "158--??",
  year =         "1988",
  bibdate =      "Fri Dec 29 18:28:25 1995",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.1.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— **axiom.bib** —

```
@book{Anon91,
  author = "Anonymous",
  title = {{Challenges of a Changing World (2 Volumes)}},
  publisher = "American Society for Engineering Education",
  year = "1991",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Anon95,
  author = "Anonymous",
  title = {{Zeitschrift fur Angewandte Mathematik und Physik, 75 (suppl. 2)}},
  publisher = "Unknown",
  year = "1995",
  issn = "0044-2267",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@Article{Arna95,
  author = "Arnault, Francois",
  title = {{Constructing Carmichael numbers which are strong pseudoprimes to
           several bases}},
  year = "1995",
  journal = "Journal of Symbolic Computation",
  volume  = "20",
  number = "2",
```

```
  pages = "151-161",
  link = "\url{http://ac.els-cdn.com/S0747717185710425/1-s2.0-S0747717185710425-main.pdf}",
  algebra = "\newline\refto{package PRIMES IntegerPrimesPackage}",
  abstract =
    "We describe here a method of constructing Carmichael numbers which
    are strong pseudoprimes to some set of prime bases. We apply it to
    find composite numbers which are found to be prime by the Rabin-Miller
    test of packages as Axiom or Maple. We also use a variation of this
    method to construct strong Lucas pseudoprimes with respect to several
    pairs of parameters.",
  paper = "Arna95.pdf",
  keywords = "axiomref",
  beebe = "Arnault:1995:CCN"
}
```

———-

— Arnault:1995:CCN —

```
@Article{Arnault:1995:CCN,
  author =       "Fran{\c{c}}ois Arnault",
  title =        {{Constructing {Carmichael} Numbers Which are Strong
                 Pseudoprimes to Several Bases}},
  journal =      j-J-SYMBOLIC-COMP,
  volume =       "20",
  number =       "2",
  pages =        "151--162 (or 151--161??)",
  month =        aug,
  year =         "1995",
  CODEN =        "JSYCEH",
  ISSN =         "0747-7171 (print), 1095-855X (electronic)",
  ISSN-L =       "0747-7171",
  MRclass =      "11Y11 (11A51)",
  MRnumber =     "96k:11153",
  MRreviewer =   "Andrew Granville",
  bibdate =      "Sat May 10 15:54:09 MDT 1997",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib;
                 http://www.math.utah.edu/pub/tex/bib/jsymcomp.bib",
  acknowledgement = ack-nhfb,
  classcodes =   "C7310 (Mathematics computing); C1160 (Combinatorial
                 mathematics)",
  corpsource =   "Fac. des Sci., Limoges Univ., France",
  fjournal =     "Journal of Symbolic Computation",
  link =  "\url{http://www.sciencedirect.com/science/journal/07477171}",
  keywords =     "Axiom; Carmichael numbers; composite numbers; Lucas;
                 Maple; number theory; pseudoprimes; Rabin-Miller test;
                 symbol manipulation",
  treatment =    "T Theoretical or Mathematical",
}
```

———-

— **axiom.bib** —

```
@article{Augo97,
  author = "Augot, Daniel and Camion, Paul",
  title = {{On the computation of minimal polynomials, cyclic vectors,
           and Frobenius forms}},
  journal = "Linear Algebra Appl.",
  volume = "260",
  pages = "61-94",
  year = "1997",
  abstract =
    "Algorithms related to the computation of the minimal polynomial of an
    $x\times n$ matrix over a field $K$ are introduced. The complexity of
    the first algorithm, where the complete factorization of the
    characteristic polynomial is needed, is $O(\sqrt{n}\cdot n^3)$. An
    iterative algorithm for finding the minimal polynomial has complexity
    $O(n^3+n^2m^2)$, where $m$ is a parameter of the shift Hessenberg
    matrix used. The method does not require the knowlege of the
    characteristic polynomial. The average value of $m$ is $O(log n)$.

    Next methods are discussed for finding a cyclic vector for a matrix.
    The authors first consider the case when its characteristic polynomial
    is squarefree. Using the shift Hessenberg form leads to an algorithm
    at cost $O(n^3 + n^2m^2)$. A more sophisticated recurrent procedure
    gives the result in $O(n^3)$ steps. In particular, a normal basis for
    an extended finite field of size $q^n$ will be obtained with complexity
    $O(n^3+n^2 log q)$.

    Finally, the Frobenius form is obtained with asymptotic average
    complexity $O(n^3 log n)$.",
  paper = "Augo97.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{AxiomPressRelease,
  author = "Numerical Algorithms Group",
  title = {{Axiom Press Release}},
  link = "\url{http://www.dorn.org/uni/sls/kap05/e08\_01.htm}",
  year = "1996",
  month = "July",
  day = "24",
  paper = "AxiomPressRelease.tgz",
  keywords = "axiomref"
}
```

———-

1.40.2 B

— axiom.bib —

```
@inproceedings{Babo16,
  author = "Babovsky, Hans and Grabmeier, Johannes",
  title = {{Calculus and design of discrete velocity models using
           computer algebra}},
  booktitle = "AIP Conference Proceedings",
  volume = "1786",
  isbn = "978-0-7354-1448-8",
  link = "\url{http://aip.scitation.org/doi/pdf/10.1063/1.4967672}",
  year = "2016",
  abstract =
    "In [2,3], a framework for a calculus with Discrete Velocity Models
    (DVM) has been derived. The rotational symmetry of the discrete
    velocities can be modelled algebraically by the action of the cyclic
    group $C_4$ -- or including reflections of the dihedral group $D_4$.
    Taking this point of view, the linearized collision operator can be
    represented in a compact form as a matrix of elements in the group
    algebra. Or in other words, by choosing a special numbering it
    exhibits a certain block structure which lets it appear as a matrix
    with entries in a certain polynomial ring. A convenient way for
    approaching such a structure is the use of a computer algebra system
    able to treat these (predefined) algebraic structures. We use the
    computer algebra system FriCAS/AXIOM [4,5] for the generation of the
    velocity and the collision sets and for the analysis of the structure
    of the collision operator. Concerning the fluid dynamic limit, the
    system provides the characterization of sets of collisions and their
    contribution to the flow parameters. It allows the design of
    rotationally invariant symmetric models for prescribed Prandtl
    numbers.  The implementation in FriCAS/AXIOM is explained and its
    results for a 25-velocity model are presented.",
  paper = "Babo16.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Bacl14,
  author = "Baclawski, Krystian",
  title = {{SPAD language type checker}},
  journal = "unknown",
  year = "2014",
  link = "\url{http://github.com/cahirwpz/phd}",
  abstract = "
   The project aims to deliver a new type checker for SPAD language.
   Several improvements over current type checker are planned.
   \begin{itemize}
   \item introduce better type inference
   \item introduce modern language constructs
```

```
   \item produce understandable diagnostic messages
   \item eliminate well known bugs in the type system
   \item find new type errors
   \end{itemize}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Bake16,
  author = "Baker, Martin",
  title = {{Axiom Maths Program}},
  year = "2016",
  link = "\url{http://www.euclideanspace.com/prog/scratchpad/axiom/index.htm}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Bake16a,
  author = "Baker, Martin",
  title = {{Axioms in Axiom}},
  year = "2016",
  link = "\url{http://www.euclideanspace.com/prog/scratchpad/axiomsinAxiom}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Ball14,
  author = "Ballarin, Clemens",
  title = {{Locales: a module system for mathematical theories}},
  journal = "J. Autom. Reasoning",
  volume = "52",
  number = "2",
  pages = "123-153",
  year = "2014",
  link = "\url{http://www21.in.tum.de/~ballarin/publications/jar2013.pdf}",
  abstract =
    "Locales are a module system for managing theory hierarchies in a
    theorem prover through theory interpretation. They are available for
    the theorem prover Isabelle. In this paper, their semantics is defined
    in terms of local theories and morphisms. Locales aim at providing
    flexible means of extension and reuse. Theory modules (which are
    called locales) may be extended by definitions and
```

```
    theorems. Interpretation of Isabelle's global theories and proof
    contexts is possible via morphisms. Even the locale hierarchy may be
    changed if declared relations between locales do not adequately
    reflect logical relations, which are implied by the locales'
    specifications. By discussing their design and relating it to more
    commonly known structuring mechanisms of programming languages and
    provers, locales are made accessible to a wider audience beyond the
    users of Isabelle. The discussed mechanisms include ML-style functors,
    type classes and mixins (the latter are found in modern
    object-oriented languages).",
  paper = "Ball14.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Barn02,
  author = "Barnett, Michael P.",
  title = {{Computer algebra in the life sciences}},
  journal = "SIGSAM Bulletin",
  volume = "36",
  number = "4",
  pages = "5-31",
  year = "2002",
  link = "\url{https://notendur.hi.is/vae11/\%C3\%9Eekking/Systems\%20Biology/Biological\%20Algebra.PDF}",
  abstract =
    "This note (1) provides references to recent work that applies computer
    algebra (CA) to the life sciences, (2) cites literature that explains
    the biological background of each application, (3) states the
    mathematical methods that are used, (4) mentions the benefits of CA,
    and (5) suggests some topics for future work.",
  paper = "Barn02.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Baru19,
  author = "Baruch, Robert",
  title = {{Very Basic Introduction to Formal Verification}},
  year = "2019",
  link = "\url{https://www.youtube.com/watch?v=9e7F1XhjhKw}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Baru19a,
  author = "Baruch, Robert",
  title = {{Cmod A7 Reference Manual}},
  year = "2019",
  link = "\url{https://reference.digilentinc.com/_media/reference/programmable-logic/cmod-a7/cmod_a7_rm.pdf}",
  paper = "Baru19a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Baru19b,
  author = "Baruch, Robert",
  title = {{Cmod A7 Schematic}},
  year = "2019",
  link = "\url{https://reference.digilentinc.com/_media/cmod-a7/cmod_a7_sch.pdf}",
  paper = "Baru19b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Beeb14,
  author = "Beebe, Nelson H. F.",
  title = {{A Bibliography of Publications about the AXIOM
           (formerly, Scratchpad) Symbolic Algebra Language}},
  year = "2014",
  link = "\url{http://www.netlib.org/bibnet/journals/axiom.ps.gz}",
  paper = "Beeb14.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Benk98,
  author = "Benker, Hans",
  title = {{Engineering mathematics with computer algebra systems}},
  publisher = "Unknown",
  year = "1998",
  comment = "German",
  keywords = "axiomref",
  beebe = "Benker:1998:ICS"
}
```

———-

— **Benker:1998:ICS** —

```
@Book{Benker:1998:ICS,
  author =       "Hans Benker",
  title =        "{Ingenieurmathematik mit Computeralgebra-Systemen.
                 AXIOM, DERIVE, MACSYMA, MAPLE, MATHCAD, MATHEMATICA,
                 MATLAB und MuPAD in der Anwendung}. (German)
                 {Engineering mathematics with computer algebra systems.
                 The applications: AXIOM, DERIVE, MACSYMA, MAPLE,
                 MATHCAD, MATHEMATICA, MATLAB UND MuPAD.}",
  publisher =    pub-VIEWEG,
  address =      pub-VIEWEG:adr,
  pages =        "xiii + 439",
  year =         "1998",
  MRclass =      "68W30 (Symbolic computation and algebraic computation)
                 68-01 (Textbooks (computer science)) 00A06 (Mathematics
                 for non-mathematicians) 68-04 (Machine computation,
                 programs (computer science)) 65D18 (Computer graphics
                 and computational geometry)",
  bibdate =      "Tue Mar 30 18:49:35 2010",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  ZMnumber =     "0909.68109",
  acknowledgement = ack-nhfb,
  keywords =     "calculus; differential equations; Fourier transform;
                 Laplace transform; linear algebra; optimization;
                 probability theory; statistics; textbook",
  language =     "German",
  reviewer =     "Helmut K{\"o}cher (Dresden)",
}
```

———-

— **axiom.bib** —

```
@misc{Bern14,
  author = "Bernard, Joey",
  title = {{Open Axiom}},
  link = "\url{http://www.linuxjournal.com/content/open-axiom}",
  year = "2014",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Bett99,
  author = "Betten, Anton and Kohnert, Axel and Laue, Reinhard and
            Wassermann, Alfred",
  title = {{Algebraic Combinatorics and Applications}},
  year = "1999",
  publisher = "Springer",
```

```
  isbn = "978-3-540-41110-9",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Blai70a,
  author = "Blair, Fred W. and Griesmer, James H. and Jenks, Richard D.",
  title = {{An interactive facility for symbolic mathematics}},
  journal = "ACM SIGSAM",
  volume = "14",
  year = "1970",
  pages = "17-18",
  abstract =
    "This paper describes a system designed to provide an interactive
    symbolic computational facility for the mathematician user. To carry
    out this objective, an experimental LISP system has been implemented
    for IBM System/360 computers. Using this LISP system as a base,
    portions of several systems have been combined and augmented to
    provide the following facilities to a user:(1) rational function
    manipulation and simplification; symbolic differentiation (Anthony
    Hearn's REDUCE)(2) symbolic integration (Joel Moses' SIN)(3)
    polynomial factorization, solution of linear differential equations,
    direct and inverse symbolic Laplace transforms (Carl Engelman's
    MATHLAB, including Knut Korsvold's simplification system)(4) unlimited
    precision integer arithmetic(5) manipulation of arrays containing
    symbolic entries(6) two-dimensional output on IBM 2741 terminals or
    IBM 2250 displays (William Martin's Symbolic Mathematical Laboratory,
    and Jonathan Millen's CHARYBDIS program from MATHLAB)(7)
    self-extending language facility (META/LISP).The user language created
    for the system incorporates a subset of ``customary'' mathematical
    notation. Data objects include sequences (both finite and infinite)
    and arrays of arbitrary rank. Assignment statements are the
    fundamental commands in the user language; they may contain
    ``for''-clauses which restrict the domain for which the assignment is
    valid and permit ``piecewise'' and recursive definition of new operators
    and functions. The user may also enter syntax definition statements in
    order to introduce new notations into the system.Expressions appearing
    in assignment statements may include ``where''-clauses which allow user
    control over the ``environment'' used in evaluation. Otherwise,
    evaluation of expressions occurs in the current environment created by
    the successive user commands, with certain operations such as
    integration, differentiation, and simplification performed
    automatically.Translators for the user language and for a resident
    higher-level procedural language facility are written in META/LISP, a
    new self-compiling translator-writing system. As a result, all input
    language facilities as well as the underlying manipulation routines
    may be interactively extended by an experienced user.",
  paper = "Blai70a.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **ignore** —

```
@techreport{Blai70,
  author = "Blair, Fred W. and Jenks, Richard D.",
  title = {{LPL: LISP programming language}},
  type = "technical report",
  institution = "IBM",
  number = "RC3062",
  year = "1970",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Bosm94,
  author = "Bosma, Wieb and Cannon, John and Matthews, Graham",
  title = {{Programming with algebraic structures: Design of the Magma
           language}},
  booktitle = "Proc. ISSAC 94",
  series = "ISSAC 94",
  year = "1994",
  publisher = "ACM Press",
  location = "Baltimore, MD",
  pages = "52-57",
  link = "\url{http://www.math.ru.nl/~bosma/pubs/ISSAC94.pdf}",
  abstract =
    "MAGMA is a new software system for computational algebra, number
    theory and geometry whose design is centred on the concept of
    algebraic structure (magma). The use of algebraic structure as a
    design paradigm provides a natural strong typing mechanism. Further,
    structures and their morphisms appear in the language as first class
    objects. Standard mathematical notions are used for the basic data
    types. The result is a powerful, clean language which deals with
    objects in a mathematically rigorous manner. The conceptual and
    implementation ideas behind MAGMA will be examined in this paper. This
    conceptual base differs significantly from those underlying other
    computer algebra systems.",
  paper = "Bosm94.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Bosm97,
```

```
  author = "Bosma, Wieb and Cannon, John and Playoust, Catherine",
  title = {{The Magma Algebra System I: The User Language}},
  journal = "J. Symbolic Computation",
  volume = "24",
  pages = "235-265",
  year = "1997",
  link = "\url{http://lib.org.by/_djvu/_Papers/Computer_algebra/CAS%20systems/}",
  abstract =
    "In the first of two papers on MAGMA, a new system for computational
    algebra, we present the MAGMA language, outline the design principles
    and theoretical background, and indicate its scope and use. Particular
    attention is given to the constructors for structures, maps, and sets.",
  paper = "Bosm97.djvu",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Boyl88,
  author = "Boyle, Ann and Caviness, B.F. and Hearn, Anthony C.",
  title = {{Future Directions for Research in Symbolic Computation}},
  publisher = "Soc. for Industrial and Applied Mathematics",
  year = "1988",
  link = "\url{http://www.eecis.udel.edu/~caviness/wsreport.pdf}",
  paper = "Boyl88.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Brad09,
  author = "Bradford, Russell and Davenport, James H. and
            Sangwin, Christopher J.",
  title = {{A comparison of equality in computer algebra and correctness
           in mathematical pedagogy}},
  year = "2009",
  booktitle = "Intelligent Computer Mathematics, 16th symposium",
  series = "Calculemus 2009",
  pages = "75-89",
  isbn = "978-3-642-02613-3",
  link = "\url{http://opus.bath.ac.uk/15188/1/RJBJHDCJSv2.pdf}",
  abstract =
    "How do we recognize when an answer is ``right''? This is a question
    that has bedevilled the use of computer systems in mathematics (as
    opposed to arithmetic) ever since their introduction. A computer
    system can certainly say that some answers are definitely wrong, in
    the sense that they are provably not an answer to the question
    posed. However, an answer can be mathematically right without being
```

```
    pedagogically right. Here we explore the differences and show that,
    despite the apparent distinction, it is possible to make many of the
    differences amenable to formal treatment, by asking ‘‘under which
    congruence is the pupil’s answer equal to the teacher’s?’’.",
  paper = "Brad09.pdf",
  keywords = "axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Brem08,
  author = "Bremner, Murray R. and Murakami, Lucia I. and Shestakov, Ivan P.",
  title = {{Nonassociative Algebras}},
  year = "2008",
  algebra = "\newline\refto{category NARNG NonAssociativeRng}",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.146.2665}",
  abstract =
    "One of the earliest surveys on nonassociative algebras is the article
    by Shirshov which introduced the phrase ‘‘rings that are nearly
    associative’’. The first book in the English language devoted to a
    systematic study of nonassociative algebras is Schafer (Scha66). A
    comprehensive exposition of the work of the Russian School is
    Zhevlakov, Slinko, Shestakov and Shirshov. A collection of open
    research problems in algebra, including many problems on
    nonassociative algebra, is the {\sl Dniester Notebook}; the survey
    article by Kuzmin and Shetakov is from the same period. Three books on
    Jordan algebras which contain substantial material on general
    nonassociative algebras are Braun and Koecher, Jacobson, and
    McCrimmon. Recent research appears in the Proceedings of the
    International Conferences on Nonassociative Algebras and its
    Applications. The present section provides very limited information on
    Lie algebras, since they are the subject of Section 16.4. The last
    part (section 9) of the present section presents three applications of
    computational linear algebra to the study of polynomial identiies for
    nonassociative algebras: pseudorandom vectors in a nonassociative
    algebra, the expansion matrix for a nonassociative operation, and the
    representation theory of the symmetric group."
}
```

———-

— **axiom.bib** —

```
@InProceedings{Breu98,
  author = "Breuer, Thomas and Linton, Steve",
  title = {{The GAP 4 type system organising algebraic algorithms}},
  booktitle = "Proc. ISSAC 98",
  series = "ISSAC 98",
  year = "1998",
  publisher = "ACM Press",
```

```
  location = "Rostock, Germany",
  pages = "13-15",
  link = "\url{http://www.gap-system.org/Doc/Talks/paper.ps}",
  abstract =
    "Version 4 of the GAP (Groups, Algorithms, Programming) system for
    computational discrete mathematics has a number of novel features. In
    this paper, we describe the type system, and the way in which it is
    used for method selection. This system is central to the organization
    of the library which is the main part of the GAP system. Unlike
    simpler object-oriented systems, GAP allows method selection based on
    the types of all arguments and on certain aspects of the relationship
    between the arguments. In addition, the type of an object can change,
    in a controlled way, during its life. This reflects information about
    the object which has been computed and stored. Individual methods can
    be written and installed independently. Furthermore, most checking of
    the arguments is done in a uniform way by the method selection system,
    making individual methods simpler and less prone to error. The methods
    are combined automatically to produce a powerful and usable system for
    interactive use or programming.",
  paper = "Breu98.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Broa95,
  author = "Broadbery, P. A. and G{\'o}mez-D{\'\i}az, T. and Watt, S. M.",
  title = {{On the Implementation of Dynamic Evaluation}},
  booktitle = "Unknown",
  publisher = "Unknown",
  year = "1995",
  pages = "77-84",
  isbn = "0-89791-699-9",
  link = "\url{http://pdf.aminer.org/000/449/014/on_the_implementation_of_dynamic_evaluation.pdf}",
  abstract = "
    Dynamic evaluation is a technique for producing multiple results
    according to a decision tree which evolves with program execution.
    Sometimes it is desired to produce results for all possible branches
    in the decision tree, while on other occasions, it may be sufficient
    to compute a single result which satisfies certain properties. This
    techinique finds use in computer algebra where computing the correct
    result depends on recognizing and properly handling special cases of
    parameters. In previous work, programs using dynamic evaluation have
    explored all branches of decision trees by repeating the computations
    prior to decision points.

    This paper presents two new implementations of dynamic evaluation
    which avoid recomputing intermediate results. The first approach uses
    Scheme ‘‘continuations’’ to record state for resuming program
    execution. The second implementation uses the Unix ‘‘fork’’ operation
    to form new processes to explore alternative branches in parallel.",
```

```
  paper = "Broa95.pdf",
  keywords = "axiomref",
  beebe = "Broadbery:1995:IDE"
}
```

— Broadbery:1995:IDE —

```
@InProceedings{Broadbery:1995:IDE,
  author =       "P. A. Broadbery and T. G{\'o}mez-D{\'\i}az and S. M.
                 Watt",
  title =        {{On the Implementation of Dynamic Evaluation}},
  crossref =     "Levelt:1995:IPI",
  pages =        "77--84",
  year =         "1995",
  bibdate =      "Thu Mar 12 08:42:30 MST 1998",
  bibsource =    "http://www.acm.org/pubs/toc/;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://www.acm.org:80/pubs/citations/proceedings/issac/220346/p77-broadbery/}",
  abstract =     "Dynamic evaluation is a technique for producing
                 multiple results according to a decision tree which
                 evolves with program execution. Sometimes we need to
                 produce results for all possible branches in the
                 decision tree, while on other occasions it may be
                 sufficient to compute a single result which satisfies
                 certain properties. This technique finds use in
                 computer algebra where computing the correct result
                 depends on recognising and properly handling special
                 cases of parameters. In previous work, programs using
                 dynamic evaluation have explored all branches of
                 decision trees by repeating the computations prior to
                 decision points. The paper presents two new
                 implementations of dynamic evaluation which avoid
                 recomputing intermediate results. The first approach
                 uses Scheme `continuations' to record the state for
                 resuming program execution. The second implementation
                 uses the Unix `fork' operation to form new processes to
                 explore alternative branches in parallel. These
                 implementations are based on modifications to Lisp- and
                 C-based run-time systems for the Axiom Version 2
                 extension language (previously known as $ A^{\mbox
                 {Hash}} $ ). This allows the same high-level source
                 code to be compared using the `re-evaluation', the
                 `continuation', and the `fork' implementations.",
  acknowledgement = ack-nhfb,
  affiliation =  "Numerical Algorithms Group Ltd., Oxford, UK",
  classification = "C1140E (Game theory); C1160 (Combinatorial
                 mathematics); C6130 (Data handling techniques); C6150G
                 (Diagnostic, testing, debugging and evaluating
                 systems); C6150J (Operating systems)",
  keywords =     "algebraic computation, Dynamic evaluation; algorithms;
                 Axiom Version 2 extension language; C-based run-time
```

```
                systems; Computer algebra; Decision points; Decision
                tree; High-level source code; ISSAC; languages;
                Lisp-based run-time systems; Multiple results; Program
                execution; Re-evaluation; Scheme continuations; State
                recording; symbolic computation; Unix fork operation",
  subject =     "{\bf I.1.2} Computing Methodologies, SYMBOLIC AND
                ALGEBRAIC MANIPULATION, Algorithms, Algebraic
                algorithms. {\bf D.3.2} Software, PROGRAMMING
                LANGUAGES, Language Classifications, SCHEME. {\bf
                F.2.1} Theory of Computation, ANALYSIS OF ALGORITHMS
                AND PROBLEM COMPLEXITY, Numerical Algorithms and
                Problems, Computations on polynomials. {\bf D.3.2}
                Software, PROGRAMMING LANGUAGES, Language
                Classifications, C.",
  thesaurus =   "Decision theory; Symbol manipulation; System
                monitoring; Trees [mathematics]; Unix",
}
```

———-

— **axiom.bib** —

```
@misc{Bronxx,
  author = "Bronstein, Manuel",
  title = {{Symbolic Integration in Computer Algebra}},
  link = "\url{http://www.iaea.org/inis/collection/NCLCollectionStore/_Public/26/042/26042580.pdf}",
  year = "1990",
  abstract =
    "One major goal of symbolic integrators is to determine under what
    circumstances the integral of the elementary functions of calculus can
    themselves be expressed as elementary functions.  While using tables
    and the ad hoc tricks taught in calculus courses can have some limited
    success, a decision procedure is necessary in all but the most trivial
    cases.  The first complete algorithm for solving this problem was
    presented by Risch in 1969, but its complexity, specially when
    algebraic functions are present in the integrand, has prevented it
    from being fully implemented.  Over the past 20 years, the Risch
    integration algorithm has been completed, extended, and improved to
    such a point that recent computer algebra systems can integrate
    elementary functions without using any of the heuristics traditionally
    taught in calculus courses and used by older systems.  In this talk,
    we give an overview and description of the algorithms used in the
    Scratchpad symbolic integrator, and illustrate them with integrals
    drawn from the physical sciences.",
  paper = "Bron90.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
\article{Brun04,
  author = "Brunelli, J.C.",
  title = {{PSEUDO: applications of streams and lazy evaluation to integrable
           models}},
  journal = "Comput. Phys. Commun.",
  volume = "163",
  number = "1",
  pages = "22-40",
  year = "2004",
  abstract =
    "Procedures to manipulate pseudo-differential operators in MAPLE are
    implemented in the program PSEUDO to perform calculations with
    integrable models. We use lazy evaluation and streams to represent and
    operate with pseudo-differential operators. No order of truncation is
    needed since terms are produced on demand. We give a series of
    concrete examples.",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Boeh89,
  author = "Boehm, Hans-J.",
  title = {{Type Inference in the Presence of Type Abstraction}},
  journal = "ACM SIGPLAN Notices",
  volume = "24",
  number = "7",
  month = "July",
  year = "1989",
  pages = "192-206",
  link = "\url{http://www.acm.org/pubs/citations/proceedings/pldi/73141/p192-boehm}",
  abstract = "
    A number of recent programming language designs incorporate a type
    checking system based on the Girard-Reynolds polymorphic
    $\lambda$-calculus.  This allows the construction of general purpose,
    reusable software without sacrificing compile-time type checking. A
    major factor constraining the implementation of these languages is the
    difficulty of automatically inferring the lengthy type information
    that is otherwise required if full use is made of these
    languages. There is no known algorithm to solve any natural and fully
    general formulation of the ``type inference'' problem. One very
    reasonable formulation of the problem is known to be undecidable.

    Here we define a restricted version of the type inference problem and
    present an efficient algorithm for its solution. We argue that the
    restriction is sufficiently weak to be unobtrusive in practice.",
  paper = "Boeh89.pdf",
  keywords = "axiomref",
  beebe = "Boehm:1989:TIP"
}
```

———-

— **Boehm:1989:TIP** —

```
@Article{Boehm:1989:TIP,
  author =       "Hans-J. Boehm",
  title =        {{Type inference in the presence of type abstraction}},
  journal =      j-SIGPLAN,
  volume =       "24",
  number =       "7",
  pages =        "192--206",
  month =        jul,
  year =         "1989",
  CODEN =        "SINODQ",
  ISSN =         "0362-1340 (print), 1523-2867 (print), 1558-1160
                 (electronic)",
  ISSN-L =       "0362-1340",
  bibdate =      "Thu May 13 12:31:07 MDT 1999",
  bibsource =    "http://www.acm.org/pubs/contents/proceedings/pldi/73141/index.html;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://www.acm.org:80/pubs/citations/proceedings/pldi/73141/p192-boehm/}",
  abstract =     "A number of recent programming language designs
                 incorporate a type checking system based on the
                 Girard--Reynolds polymorphic \$lambda@-calculus. This
                 allows the construction of general purpose, reusable
                 software without sacrificing compile-time type
                 checking. A major factor constraining the
                 implementation of these languages is the difficulty of
                 automatically inferring the lengthy type information
                 that is otherwise required if full use is made of these
                 languages. There is no known algorithm to solve any
                 natural and fully general formulation of this `type
                 inference' problem. One very reasonable formulation of
                 the problem is known to be undecidable. Here we define
                 a restricted version of the type inference problem and
                 present an efficient algorithm for its solution. We
                 argue that the restriction is sufficiently weak to be
                 unobtrusive in practice.",
  acknowledgement = ack-nhfb,
  affiliationaddress = "Houston, TX, USA",
  annote =       "Published as part of the Proceedings of PLDI'89.",
  classification = "723",
  conference =   "Proceedings of the SIGPLAN '89 Conference on
                 Programming Language Design and Implementation",
  fjournal =     "ACM SIGPLAN Notices",
  link =  "\url{http://portal.acm.org/browse_dl.cfm?idx=J706}",
  journalabr =   "SIGPLAN Not",
  keywords =     "Abstract Data Types (ADT); algorithms; Computer
                 Programming Languages--Design; Data Processing; Data
                 Structures; design; languages; Scratchpad; theory",
  meetingaddress = "Portland, OR, USA",
  meetingdate =  "Jun 21--23 1989",
  meetingdate2 = "06/21--23/89",
  sponsor =      "ACM, Special Interest Group on Programming Languages,
```

```
                New York; SS NY, USA",
  subject =     "{\bf F.4.1} Theory of Computation, MATHEMATICAL LOGIC
                AND FORMAL LANGUAGES, Mathematical Logic. {\bf F.3.3}
                Theory of Computation, LOGICS AND MEANINGS OF PROGRAMS,
                Studies of Program Constructs, Type structure.",
}
```

———-

— **axiom.bib** —

```
@inproceedings{BHGM04,
  author = "Boulton, Richard and Hardy, Ruth and Gottliebsen, Hanne
            and Martin, Ursula",
  title = {{Design verification for control engineering}},
  year = "2004",
  month = "April",
  booktitle = "Proc 4th Int. Conf. on Integrated Formal Methods",
  abstract = "
    We introduce control engineering as a new domain of application for
    formal methods. We discuss design verification, drawing attention to
    the role played by diagrammatic evaluation criteria involving numeric
    plots of a design, such as Nichols and Bode plots. We show that
    symbolic computation and computational logic can be used to discharge
    these criteria and provide symbolic, automated, and very general
    alternatives to these standard numeric tests. We illustrate our work
    with reference to a standard reference model drawn from military
    avionics.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Boul91,
  author = "Boulanger, Jean-Louis",
  title = {{Etude de la compilation de scratchpad 2}},
  year = "1991",
  month = "September",
  publisher = "Rapport de DEA Universite dl lille 1",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Bou93a,
  author = "Boulanger, Jean-Louis",
  title = {{Axiom, language fonctionnel \‘a d\’evelopement objet}},
```

```
  year = "1993",
  month = "October",
  paper = "Bou93a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Boul93b,
  author = "Boulanger, Jean-Louis",
  title = {{AXIOM, A Functional Language with Object Oriented Development}},
  year = "1993",
  abstract =
    "We present in this paper, a study about the computer algebra system
    Axiom, which gives us many very interesting Software engineering
    concepts.  This language is a functional language with an Object
    Oriented Development.  This feature is very important for modeling the
    mathematical world (Hierarchy) and provides a running with
    mathematical sense. (All objects are functions). We present many
    problems of running and development in Axiom. We can note that Aiom is
    the only system of this category.",
  paper = "Boul93b.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Boul95,
  author = "Boulanger, J.L.",
  title = {{Object Oriented Method for Axiom}},
  year = "1995",
  month = "February",
  pages = "33-41",
  publisher = "ACM SIGPLAN Notices, 30(2) CODEN SINODQ ISSN 0362-1340",
  abstract = "
    Axiom is a very powerful computer algebra system which combines two
    language paradigms (functional and OOP). Mathematical world is complex
    and mathematicians use abstraction to design it. This paper presents
    some aspects of the object oriented development in Axiom. The Axiom
    programming is based on several new tools for object oriented
    development, it uses two levels of class and some operations such that
    {\sl coerce}, {\sl retract}, or {\sl convert} which permit the type
    evolution. These notions introduce the concept of multi-view.",
  paper = "Boul95.pdf",
  keywords = "axiomref",
  beebe = "Boulanger:1995:OOM"
}
```

———-

— Boulanger:1995:OOM —

```
@Article{Boulanger:1995:OOM,
  author =       "J.-L. Boulanger",
  title =        {{Object oriented method for Axiom}},
  journal =      j-SIGPLAN,
  volume =       "30",
  number =       "2",
  pages =        "33--41",
  month =        feb,
  year =         "1995",
  CODEN =        "SINODQ",
  ISSN =         "0362-1340 (print), 1523-2867 (print), 1558-1160
                 (electronic)",
  ISSN-L =       "0362-1340",
  bibdate =      "Tue Sep 17 06:32:41 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Axiom is a very powerful computer algebra system which
                 combines two language paradigms (functional and OOP).
                 The mathematical world is complex and mathematicians
                 use abstraction to design it. The paper presents some
                 aspects of object oriented development in Axiom. Axiom
                 programming is based on several new tools for object
                 oriented development, it uses two levels of class and
                 some operations such as coerce, retract or convert
                 which permit the type evolution. These notions
                 introduce the concept of multi-view.",
  acknowledgement = ack-nhfb,
  affiliation =  "Lab. d'Inf. Fondamentale de Lille, Lille I Univ.,
                 Villeneuve d'Ascq, France",
  classification = "C4240 (Programming and algorithm theory); C6110J
                 (Object-oriented programming); C6120 (File
                 organisation); C6130 (Data handling techniques); C6140D
                 (High level languages); C7310 (Mathematics computing)",
  fjournal =     "ACM SIGPLAN Notices",
  link =  "\url{http://portal.acm.org/browse_dl.cfm?idx=J706}",
  keywords =     "Abstraction; Axiom; Axiom programming; Class; Coerce;
                 Computer algebra system; Convert; Functional language;
                 Multiview concept; Object oriented development; Object
                 oriented method; Object-oriented language; Retract;
                 Tools; Type evolution",
  language =     "English",
  pubcountry =   "USA",
  thesaurus =    "Abstract data types; Functional languages; Functional
                 programming; Mathematics computing; Object-oriented
                 languages; Object-oriented programming; Symbol
                 manipulation; Type theory",
}
```

———-

— **ignore** —

```
\bibitem[Bronstein 87]{Bro87}
  author = "Bronstein, Manuel",
  title = {{Integration of Algebraic and Mixed Functions}},
  year = "1987",
in [Wit87], p18
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Bron89,
  author = "Bronstein, Manuel",
  title = {{Simplification of real elementary functions}},
  booktitle = "Proc. ISSAC 1989",
  series = "ISSAC 1989",
  year = "1989",
  pages = "207-211",
  isbn = "0-89791-325-6",
  doi = "https://dx.doi.org/10.1145/74540.74566",
  abstract = "
    We describe an algorithm, based on Risch's real structure theorem, that
    determines explicitly all the algebraic relations among a given set of
    real elementary functions. We also provide examples from its
    implementation that illustrate the advantages over the use of complex
    logarithms and exponentials.",
  paper = "Bron89.djvu",
  keywords = "axiomref",
  beebe = "Bronstein:1989:SRE"
}
```

———-

— **Bronstein:1989:SRE** —

```
@InProceedings{Bronstein:1989:SRE,
  author =       "M. Bronstein",
  title =        {{Simplification of real elementary functions}},
  crossref =     "ACM:1989:PAI",
  pages =        "207--211",
  month =        "",
  year =         "1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The author describes an algorithm, based on Risch's
                 real structure theorem, that determines explicitly all
                 the algebraic relations among a given set of real
                 elementary functions. He provides examples from its
                 implementation in the Scratchpad computer algebra
                 system that illustrate the advantages over the use of
                 complex logarithms and exponentials.",
```

```
  acknowledgement = ack-nhfb,
  affiliation =  "IBM Res. Div., T. J. Watson Res. Center, Yorktown
                 Heights, NY, USA",
  classification = "C1110 (Algebra); C7310 (Mathematics)",
  keywords =     "Computer algebra system; Real elementary functions;
                 Real structure theorem; Scratchpad",
  language =     "English",
  thesaurus =    "Functions; Mathematics computing; Symbol
                 manipulation",
}
```

———-

— axiom.bib —

```
@inproceedings{Bron91a,
  author = "Bronstein, M.",
  title = {{The Risch Differential Equation on an Algebraic Curve}},
  booktitle = "Proc. 1991 Int. Symp. on Symbolic and Algebraic Computation",
  series = "ISSAC'91",
  year = "1991",
  pages = "241-246",
  isbn = "0-89791-437-6",
  publisher = "ACM, NY",
  abstract =
    "We present a new rational algorithm for solving Risch differential
    equations over algebraic curves. This algorithm can also be used to
    solve $n^{th}$-order linear ordinary differential equations with
    coefficients in an algebraic extension of the rational functions. In
    the general (``mixed function'') case, this algorithm finds the
    denominator of any solution of the equation.",
  paper = "Bro91a.pdf",
  keywords = "axiomref",
  beebe = "Bronstein:1991:RDE"
}
```

———-

— Bronstein:1991:RDE —

```
@InProceedings{Bronstein:1991:RDE,
  author =       "M. Bronstein",
  title =        {{The {Risch} differential equation on an algebraic
                 curve}},
  crossref =     "Watt:1991:PIS",
  pages =        "241--246",
  month =        "",
  year =         "1991",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The author presents a new rational algorithm for
                 solving Risch differential equations over algebraic
                 curves. This algorithm can also be used to solve n/sup
```

```
                th/-order linear ordinary differential equations with
                coefficients in an algebraic extension of the rational
                functions. In the general ('mixed function') case, this
                algorithm finds the denominator of any solution of the
                equation. The algorithm has been implemented in the
                Maple and Scratchpad computer algebra systems.",
  acknowledgement = ack-nhfb,
  affiliation =  "Inf. ETH-Zentrum, Zurich, Switzerland",
  classification = "C4170 (Differential equations); C7310
                (Mathematics)",
  keywords =     "Algebraic curve; Computer algebra systems; Maple;
                N/sup th/-order linear ordinary differential equations;
                Rational algorithm; Rational functions; Risch
                differential equation; Scratchpad",
  language =     "English",
  thesaurus =    "Differential equations; Symbol manipulation",
}
```

———-

— ignore —

```
\bibitem[Bronstein 91c]{Bro91c}
  author = "Bronstein, Manuel",
  title = {{Computer Algebra and Indefinite Integrals}},
  year = "1991",
in Computer Aided Proofs in Analysis, K.R. Meyers et al. (eds)
Springer-Verlag, NY (1991)
  abstract = "
    We give an overview, from an analytical point of view, of decision
    procedures for determining whether an elementary function has an
    elementary function has an elementary antiderivative. We give examples
    of algebraic functions which are integrable and non-integrable in
    closed form, and mention the current implementation of various computer
    algebra systems.",
  paper = "Bro91c.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@misc{Bron92,
  author = "Bronstein, Manuel",
  title = {{Linear Ordinary Differential Equations: Breaking Through the
           Order 2 Barrier}},
  booktitle = "Proc. ISSAC 1992",
  series = "ISSAC 92",
  publisher = "ACM",
  pages = "42-48",
  year = "1992",
```

```
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/issac92.ps.gz}",
  algebra =
   "\newline\refto{package ODECONST ConstantLODE}
    \newline\refto{package LODEEF ElementaryFunctionLODESolver}
    \newline\refto{package ODEPAL PureAlgebraicLODE}
    \newline\refto{package ODERAT RationalLODE}
    \newline\refto{package ODERED ReduceLODE}",
  abstract = "
    A major subproblem for algorithms that either factor ordinary linear
    differential equations or compute their closed form solutions is to
    find their solutions $y$ which satisfy $y^{'}/y \in \overline{K}(x)$
    where $K$ is the constant field for the coefficients of the equation.
    While a decision procedure for this subproblem was known in the
    $19^{th}$ century, it requires factoring polynomials over
    $\overline{K}$ and has not been implemented in full generality. We
    present here an efficient algorithm for this subproblem, which has
    been implemented in the AXIOM computer algebra system for equations of
    arbitrary order over arbitrary fields of characteristic 0. This
    algorithm never needs to compute with the individual complex
    singularities of the equation, and algebraic numbers are added only
    when they appear in the potential solutions.  Implementation of the
    complete Singer algorithm for $n=2,3$ based on this building block is
    in progress.",
  paper = "Bron92.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Bron92a,
  author = "Bronstein, Manuel",
  title = {{Formulas for Series Computation}},
  journal = "Applied Algebra in Engineering, Communication and Computing",
  volume = "2",
  pages = "195-206",
  year = "1992",
  abstract =
    "We describe an algorithm that computes polynomials whose roots are
    the coefficients of any specific order of the Laurent series of a
    rational function. The algorithm uses only rational operations in the
    coefficient field of the function. This allows us to compute with the
    principal parts of the Laurent series, in particular with the residues
    of the function, without factoring or splitting its denominator. As
    applications, we get a generalisation of the residue formulas used in
    symbolic integration algorithms to $n^{th}$-order formulas. We also
    get an algorithm that computes explicitly the principlal parts at all
    the poles simultaneously, while computing in the field generated by
    the coefficients of the series instead of the one generated by the
    poles of the function. This yields an improved version of the
    necessary conditions for the various cases of Kovacic's algorithm,
    that expresses those conditions in the smallest possible extension
```

```
    field."
}
```

———-

— **ignore** —

```
\bibitem[Bronstein 93]{Bro93}
  author = "Bronstein, Manuel (ed)",
  year = "1993",
  month = "July"
  isbn = "0-89791-604-2",
ISSAC'93: proceedings of the 1993 International Symposium on Symbolic
and Algebraic Computation, Kiev, Ukraine,
ACM Press New York, NY 10036, USA, ISBN
LCCN QA76.95 I59 1993 ACM order number 505930
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Brunelli 08]{Brun08}
  author = "Brunelli, J.C.",
  title = {{Streams and Lazy Evaluation Applied to Integrable Models}},
  year = "2008",
  link = "\url{http://arxiv.org/PS_cache/nlin/pdf/0408/0408058v1.pdf}",
  abstract = "
    Computer algebra procedures to manipulate pseudo-differential
    operators are implemented to perform calculations with integrable
    models. We use lazy evaluation and streams to represent and operate
    with pseudo-differential operators. No order of truncation is needed
    since terms are produced on demand. We give a series of concrete
    examples using the computer algebra language MAPLE.",
  paper = "Brun08.pdf",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Bronstein 93]{BS93}
  author = "Bronstein, Manuel and Salvy, Bruno",
  title = {{Full Partial Fraction Decomposition of Rational Functions}},
  year = "1993",
  pages = "157-160",
  isbn = "0-89791-604-2",
In Bronstein [Bro93]   LCCN QA76.95 I59 1993
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Bro92a,
  author = "Bronstein, Manuel",
  title = {{Integration and Differential Equations in Computer Algebra}},
  year = "1992",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.576}",
  abstract = "
    We describe in this paper how the problems of computing indefinite
    integrals and solving linear ordinary differential equations in closed
    form are now solved by computer algebra systems. After a brief review
    of the mathematical history of those problems, we outline the two
    major algorithms for them (respectively the Risch and Singer
    algorithms) and the recent improvements on those algorithms which has
    allowed them to be implemented.",
  paper = "Bro92a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Bene94,
  author = "Beneke, T. and Schwippert, W.",
  title = {{Double-track into the future: MathCAD will gain new users with
           Standard and Plus versions}},
  journal = "Elektronik",
  volume = "43",
  number = "15",
  pages = "107-110",
  year = "1994",
  month = "July",
  abstract =
   "MathCAD software is a type of 'intelligent scratchpad with a pocket
    calculator function'. Hitherto it has been suitable only to a limited
    extent for engineering mathematics. The new Version 5.0 is now offered
    in two implementations: as an inexpensive basic package and in a
    considerably more costly Plus version. The authors question whether
    MathCAD can catch up with the classical Maple and Mathematica products.",
  keywords = "axiomref",
  beebe = "Beneke:1994:DFM",
}
```

———-

— **Beneke:1994:DFM** —

```
@Article{Beneke:1994:DFM,
  author =       "T. Beneke and W. Schwippert",
  title =        {{Double-track into the future: {MathCAD} will gain new
                 users with {Standard} and {Plus} versions}},
```

```
  journal =      j-ELECTRONIK,
  volume =       "43",
  number =       "15",
  pages =        "107--110",
  month =        jul,
  year =         "1994",
  CODEN =        "EKRKAR",
  ISSN =         "0013-5658",
  bibdate =      "Tue Sep 17 06:35:39 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "MathCAD software is a type of 'intelligent scratchpad
                 with a pocket calculator function'. Hitherto it has
                 been suitable only to a limited extent for engineering
                 mathematics. The new Version 5.0 is now offered in two
                 implementations: as an inexpensive basic package and in
                 a considerably more costly Plus version. The authors
                 question whether MathCAD can catch up with the
                 classical Maple and Mathematica products.",
  acknowledgement = ack-nhfb,
  classification = "C7310 (Mathematics computing)",
  fjournal =     "Elektronik",
  keywords =     "Engineering mathematics; Intelligent scratchpad;
                 MathCAD software",
  language =     "German",
  pubcountry =   "Germany",
  thesaurus =    "CAD; Mathematics computing; Software packages",
}
```

———-

— axiom.bib —

```
@inproceedings{Bron97a,
  author = "Bronstein, Manuel and Weil, Jacques-Arthur",
  title = {{On Symmetric Powers of Differential Operators}},
  booktitle = "Proc. of ISSAC 1997",
  series = "ISSAC'97",
  year = "1997",
  pages = "156-163",
  link = "\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/mb_papers.html}",
  publisher = "ACM, NY",
  abstract = "
    We present alternative algorithms for computing symmetric powers of
    linear ordinary differential operators. Our algorithms are applicable
    to operators with coefficients in arbitrary integral domains and
    become faster than the traditional methods for symmetric powers of
    sufficiently large order, or over sufficiently complicated coefficient
    domains. The basic ideas are also applicable to other computations
    involving cyclic vector techniques, such as exterior powers of
    differential or difference operators.",
  paper = "Bro97a.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Borwein 00]{Bor00}
  author = "Borwein, Jonathan",
  title = {{Multimedia tools for communicating mathematics}},
  year = "2000",
  pages = "58",
  isbn = "3-540-42450-4",
  publisher = "Springer-Verlag",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Brow94,
  author = "Brown, Ronald and Tonks, Andrew",
  title = {{Calculations with simplicial and cubical groups in AXIOM}},
  journal = "Journal of Symbolic Computation",
  volume =  "17",
  number = "2",
  pages = "159-179",
  year = "1994",
  month = "February",
  misc = "CODEN JSYCEH ISSN 0747-7171",
  abstract =
    "Work on calculations with simplicial and cubical groups in AXIOM was
    carried out using loan equipment and software from IBM UK and guidance
    from L. A. Lambe. We report on the results of this work, and present
    the AXIOM code written by the second author during this period. This
    includes an implementation of the monoids which model cubes and
    simplices, together with a new AXIOM category of near-rings with which
    to carry out non-abelian calculations. Examples of the use of this
    code in interactive AXIOM sessions are also given.",
  paper = "Brow94.pdf",
  keywords = "axiomref"
}
```

———-

— **Brown:1994:CSC** —

```
@Article{Brown:1994:CSC,
  author =       "R. Brown and A. Tonks",
  title =        {{Calculations with simplicial and cubical groups in
                 AXIOM}},
  journal =      j-J-SYMBOLIC-COMP,
  volume =       "17",
```

```
  number =       "2",
  pages =        "159--179",
  month =        feb,
  year =         "1994",
  CODEN =        "JSYCEH",
  ISSN =         "0747-7171 (print), 1095-855X (electronic)",
  ISSN-L =       "0747-7171",
  bibdate =      "Tue Sep 17 06:35:39 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Work on calculations with simplicial and cubical
                 groups in AXIOM was carried out using loan equipment
                 and software from IBM UK and guidance from L. A. Lambe.
                 We report on the results of this work, and present the
                 AXIOM code written by the second author during this
                 period. This includes an implementation of the monoids
                 which model cubes and simplices, together with a new
                 AXIOM category of near-rings with which to carry out
                 non-abelian calculations. Examples of the use of this
                 code in interactive AXIOM sessions are also given.",
  acknowledgement = ack-nhfb,
  affiliation =  "Sch. of Math., Univ. of Wales, Bangor, UK",
  classification = "C6130 (Data handling techniques); C7310
                 (Mathematics)",
  fjournal =     "Journal of Symbolic Computation",
  link =  "\url{http://www.sciencedirect.com/science/journal/07477171}",
  keywords =     "AXIOM; Cubical groups; Monoids; Nonabelian
                 calculations; Simplicial groups; Software",
  language =     "English",
  pubcountry =   "UK",
  thesaurus =    "Mathematics computing; Symbol manipulation",
}
```

———-

— axiom.bib —

```
@misc{Brow95,
  author = "Brown, Ronald and Dreckmann, Winfried",
  title = {{Domains of data and domains of terms in AXIOM}},
  year = "1995",
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/brown-free-c-g.pdf}",
  abstract = "
    The main new concept we wish to illustrate in this paper is a
    distinction between ‘‘domains of data’’ and ‘‘domains of terms’’, and
    its use in the programming of certain mathematical structures.
    Although this distinction is implicit in much of the programming work
    that has gone into the construction of Axiom categories and domains,
    we believe that a formalisation of this is new, that standards and
    conventions are necessary and will be useful in various other
    contexts. We shall show how this concept may be used for the coding of
    free categories and groupoids on directed graphs.",
  paper = "Brow95.pdf",
  keywords = "axiomref"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Buch11,
  author = "Buchberger, Bruno",
  title = {{Groebner Bases: A Short Introduction for System Theorists}},
  year = "2011",
  abstract =
    "In this paper, we give a brief overview on Groebner bases theory,
    addressed to novices without prior knowledge in the field. After
    explaining the general strategy for solving problems via the Groebner
    approach, we develop the concept of Groebner bases by studying
    uniqueness of polynomial division (``reduction''). For explicitly
    constructing Groebner bases, the crucial notion of S-polynomials is
    introduced, leading to the complete algorithmic solution of the
    construction problem. The algorithm is applied to examples from
    polynomial equation solving and algebraic relations. After a short
    discussion of complexity issues, we conclude the paper with some
    historical remarks and references."
}
```

———-

— **axiom.bib** —

```
@book{Buch85,
  author = "Buchberger, Bruno and Caviness, Bob F. (eds)",
  title = {{Lecture Notes in Computer Science, Vol 204}},
  isbn = "0-387-15983-5 (vol 1), 0-387-15984-3 (vol 2)",
  year = "1985",
  month = "April",
  publisher = "Springer-Verlag, Berlin, Germany",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Buh05,
  author = "Buhl, Soren L.",
  title = {{Some Reflections on Integrating a Computer Algebra System in R}},
  year = "2005",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Burg91,
  author = "Burge, W.H.",
  title = {{Scratchpad and the Rogers-Ramanujan identities}},
  booktitle = "Proc. ISSAC 1991",
  year = "1991",
  pages = "189-190",
  abstract =
    "This note sketches the part played by Scratchpad in obtaining new
    proofs of Euler's theorem and the Rogers-Ramanujan Identities.",
  paper = "Burg91.pdf",
  keywords = "axiomref",
  beebe = "Burge:1991:SRI"
}
```

———-

— **Burge:1991:SRI** —

```
@InProceedings{Burge:1991:SRI,
  author =       "W. H. Burge",
  title =        {{Scratchpad and the Rogers--Ramanujan identities}},
  crossref =     "Watt:1991:PIS",
  pages =        "189--190",
  month =        "",
  year =         "1991",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "This note sketches the part played by Scratchpad in
                 obtaining new proofs of Euler's theorem and the
                 Rogers--Ramanujan Identities.",
  acknowledgement = ack-nhfb,
  affiliation =  "IBM Thomas J. Watson Res. Center, Yorktown Heights,
                 NY, USA",
  classification = "C1160 (Combinatorial mathematics); C7310
                 (Mathematics)",
  keywords =     "Euler theorem; Infinite series; Restricted partition
                 pairs; Rogers--Ramanujan identities; Scratchpad",
  language =     "English",
  thesaurus =    "Mathematics computing; Number theory; Symbol
                 manipulation",
}
```

———-

— **axiom.bib** —

```
@techreport{Burg87,
  author = "Burge, William and Watt, Stephen",
  title = {{Infinite structures in SCRATCHPAD II}},
  year = "1987",
```

```
  institution = "IBM Research",
  type = "Technical Report",
  number = "RC 12794 (\#57573)",
  link = "\url{http://www.csd.uwo.ca/~watt/pub/reprints/1987-eurocal-infinite.pdf}",
  abstract =
    "An {\sl infinite structure} is a data structure which cannot be fully
    constructed in any fixed amount of space. Several varieties of
    infinite structures are currently supported in Scratchpad II: infinite
    sequences, radix expansions, power series and continued fractions. Two
    basic methods are employed to represent infinite structures:
    self-referential data structures and lazy evaluation. These may be
    employed separately or in conjunction.

    This paper presents recently developed facilities in Scratchpad II for
    manipulating infinite structures. General techniques for manipulating
    infinite structures are covered, as well as the higher level
    manipulations on the various types of mathematical objects represented
    by infinite structures.",
  paper = "Burg87.pdf",
  keywords = "axiomref"
}
```

———-

— **Burge:1987:ISS** —

```
@TechReport{Burge:1987:ISS,
  author =       "W. Burge and S. Watt",
  title =        {{Infinite Structures in SCRATCHPAD II}},
  number =       "RC 12794 (\#57573)",
  institution =  "IBM Thomas J. Watson Research Center",
  address =      "Bos 218, Yorktown Heights, NY 10598, USA",
  pages =        "??",
  year =         "1987",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— **ignore** —

```
\bibitem[Burge 87a]{BWM87}
  author = "Burge, William H. and Watt, Stephen M. and Morrison, Scott C.",
  title = {{Streams and Power Series}},
  year = "1987",
  pages = "9-12",
in [Wit87], pp9-12
  keywords = "axiomref"
```

———-

— axiom.bib —

```
@inproceedings{Burg87b,
  author = "Burge, William H. and Watt, Stephen M.",
  title = {{Infinite structures in Scratchpad II}},
  booktitle = "EUROCAL '87. European Conference on Computer Algebra
               Proceedings",
  year = "1987",
  pages = "138-148",
  isbn = "3-540-51517-8",
  abstract =
    "An infinite structure is a data structure which cannot be fully
    constructed in any fixed amount of space.  Several varieties of
    infinite structures are currently supported in Scratchpad II: infinite
    sequences, radix expansions, power series and continued fractions. Two
    basic methods are employed to represent infinite structures:
    self-referential data structures and lazy evaluation. These may be
    employed either separately or in conjunction. This paper presents
    recently developed facilities in Scratchpad II for manipulating
    infinite structures. General techniques for manipulating infinite
    structures are covered, as well as the higher level manipulations on
    the various types of mathematical objects represented by infinite
    structures.",
  keywords = "axiomref",
  beebe = "Burge:1989:ISS"
}
```

———-

— Burge:1989:ISS —

```
@InProceedings{Burge:1989:ISS,
  author =       "W. H. Burge and S. M. Watt",
  title =        {{Infinite structures in Scratchpad II}},
  crossref =     "Davenport:1989:EEC",
  pages =        "138--148",
  month =        "",
  year =         "1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "An infinite structure is a data structure which cannot
                 be fully constructed in any fixed amount of space.
                 Several varieties of infinite structures are currently
                 supported in Scratchpad II: infinite sequences, radix
                 expansions, power series and continued fractions. Two
                 basic methods are employed to represent infinite
                 structures: self-referential data structures and lazy
                 evaluation. These may be employed either separately or
                 in conjunction. This paper presents recently developed
                 facilities in Scratchpad II for manipulating infinite
                 structures. General techniques for manipulating
                 infinite structures are covered, as well as the higher
                 level manipulations on the various types of
```

```
                mathematical objects represented by infinite
                structures.",
  acknowledgement = ack-nhfb,
  affiliation =  "IBM Thomas J. Watson Res. Center, Yorktown Heights,
                NY, USA",
  classification = "C6120 (File organisation); C7310 (Mathematics)",
  keywords =     "Continued fractions; Higher level manipulations;
                Infinite sequences; Infinite structure; Lazy
                evaluation; Mathematical objects; Power series; Radix
                expansions; Scratchpad II; Self-referential data
                structures",
  language =     "English",
  thesaurus =    "Algebra; Data structures; Mathematics computing;
                Series [mathematics]; Software packages; Symbol
                manipulation",
}
```

———-

1.40.3 C

— **axiom.bib** —

```
@inbook{Calm82,
  author = "Calmet, J. and Hulzen, J.A. van",
  title = {{Computer Algebra Applications}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "245-258",
  abstract =
    "A survey of applications based either on fundamental algorithms
    in computer algebra or on the use of a computer algebra system is
    presented. Since many survey articles are previously published, we
    did not attempts to be exhaustive. We discuss mainly recent work
    in biology, chemistry, physics, mathematics and computer science,
    thus again confirming that applications have both engineering and
    scientific aspects, i.e. apart from delivering results they assist
    in gaining insight as well.",
  paper = "Buch82.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Calmet 94]{Cal94} Calmet, J. (ed)
Rhine Workshop on Computer Algebra, Proceedings.
Universit{\"a}t Karsruhe, Karlsruhe, Germany 1994
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Calm87,
  author = "Calmet, Jacques",
  title = {{Intelligent Computer Algebra System: Myth, Fancy or
            Reality?}},
  journal = "Lecture Notes in Computer Science",
  volume = "296",
  year = "1987",
  pages = "2-11",
  paper = "Calm87.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@article{Calm87a,
  author = "Calmet, Jacques and Lugiez, Denis",
  title = {{A Knowledge-based System for Computer Algebra}},
  journal = "ACM SIGSAM Bulletin",
  volume = "21",
  number = "1",
  year = "1987",
  abstract =
    "This paper reports on a work in progress aiming at designing and
    implementing a system for representing and manipulating
    mathematical knowledge. Its kernel is a computer algebra system
    but it shows several of the features of the so-called
    knowledge-based systems. The main issues considered here are the
    software engineering aspects of the project, the definition of a
    new language to support the system and the use of AI techniques in
    a field where algebraic algorithms are the building stones of
    systems. This defines an environment which enables not only to
    have data-bases of knowledge but also to implement an expert use
    of this knowledge.",
  paper = "Calm87a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Calm89,
  author = "Calmet, J. and Comon, H. and Lugiez, D.",
  title = {{Type Inference Using Unification in Computer Algebra}},
```

```
  journal = "LNC",
  volume = "307",
  publisher = "Springer",
  year = "1989",
  paper = "Calm89.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Calm97,
  author = "Calmet, J. and Campbell, J.A.",
  title = {{A perspective on symbolic mathematical computing and
           artificial intelligence}},
  journal = "Ann. Math. Artif. Intell.",
  volume = "19",
  number = "3-4",
  pages = "261-277",
  year = "1997",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.5425}",
  abstract =
    "The nature and history of the research area common to artificial
    intelligence and symbolic mathematical computation are examined, with
    particular reference to the topics having the greatest current amount
    of activity or potential for further development: mathematical
    knowledge-based computing environments, autonomous agents and
    multi-agent systems, transformation of problem descriptions in logics
    into algebraic forms, exploitation of machine learning, qualitative
    reasoning, and constraint-based programming. Knowledge representation,
    for mathematical knowledge, is identified as a central focus for much
    of this work. Several promising topics for further research are stated.",
  paper = "Calm97.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Cami92,
  author = "Camion, Paul and Courteau, Bernard and Montpetit, Andre",
  title = {{A combinatorial problem in Hamming Graphs and its solution
           in Scratchpad}},
  comment = {Un probl\‘eme combinatoire dans les graphies de Hamming et sa
             solution en Scratchpad},
  year = "1992",
  month = "January",
  link = "\url{https://hal.inria.fr/inria-00074974/document}",
  type = "Research report",
  number = "1586",
```

```
  institution = "Institut National de Recherche en Informatique et en
                 Automatique, Le Chesnay, France",
  abstract =
    "We present a combinatorial problem which arises in the determination
    of the complete weight coset enumerators of error-correcting codes
    [1]. In solving this problem by exponential power series with
    coefficients in a ring of multivariate polynomials, we fall on a
    system of differential equations with coefficients in a field of
    rational functions. Thanks to the abstraction capabilities of
    Scratchpad this differential equation may be solved simply and
    naturally, which seems not to be the case for the other computer
    algebra systems now available.",
  paper = "Cami92.pdf",
  keywords = "axiomref",
  beebe = "Camion:1992:PCG"
}
```

———-

— Camion:1992:PCG —

```
@TechReport{Camion:1992:PCG,
  author =       "Paul Camion and Bernard Courteau and Andre Montpetit",
  title =        "Un probl{\'e}me combinatoire dans les graphes de
                 {Hamming} et sa solution en {Scratchpad}. ({English}:
                 {A} combinatorial problem in {Hamming} graphs and its
                 solution in {Scratchpad})",
  type =         "Rapports de recherche",
  number =       "1586",
  institution =  "Institut National de Recherche en Informatique et en
                 Automatique",
  address =      "Le Chesnay, France",
  pages =        "12",
  month =        jan,
  year =         "1992",
  bibdate =      "Sat Dec 30 08:42:16 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "We present a combinatorial problem which arises in the
                 determination of the complete weight coset enumerators
                 of error-correcting codes [1]. In solving this problem
                 by exponential power series with coefficients in a ring
                 of multivariate polynomials, we fall on a system of
                 differential equations with coefficients in a field of
                 rational functions. Thanks to the abstraction
                 capabilities of Scratchpad this differential equation
                 may be solved simply and naturally, which seems not to
                 be the case for the other computer algebra systems now
                 available.",
  acknowledgement = ack-nhfb,
}
```

———-

— **ignore** —

```
\bibitem[Capriotti 00]{CCR00}
  author = "Capriotti, Olga and Cohen, Arjeh M. and Riem, Manfred",
  title = {{Java Phrasebooks for Computer Algebra and Automated Deduction}},
  link = "\url{http://www.sigsam.org/bulletin/articles/132/paper8.pdf}",
  paper = "CCR00.pdf",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Capr99a,
  author = "Capriotti, O. and Carlisle, D.",
  title = {{OpenMath and MathML: Semantic Mark Up for Mathematics}},
  year = "1999",
  link = "\url{http://www.acm.org/crossroads/xrds6-2/openmath.html}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Capr99,
  author = "Capriotti, Olga and Cohen, Arjeh M. and Cuypers, Hans and
            Sterk, Hans",
  title = {{OpenMath Technology for Interactive Mathematical Documents}},
  year = "2002",
  pages = "51-66",
  publisher = "Springer-Verlag, Berlin, Germany",
  link = "\url{http://www.win.tue.nl/~hansc/lisbon.pdf}",
  misc = "in Multimedia Tools for Communicating Mathematics",
  paper = "Capr99.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Care06,
  author = "Carette, Jacques",
  title = {{Gaussian elimination: a case study in efficient genericity
           with MetaOCaml}},
  journal = "Sci. Comput. Program.",
  volume = "62",
  number = "1",
  pages = "3-24",
  year = "2006",
```

```
  link = "\url{http://www.cas.mcmaster.ca/~carette/publications/ge.pdf}",
  abstract =
    "The Gaussian Elimination algorithm is in fact an algorithm family
    common implementations contain at least six (mostly independent)
    ``design choices''. A generic implementation can easily be parametrized
    by all these design choices, but this usually leads to slow and
    bloated code. Using MetaOCamls staging facilities, we show how we can
    produce a natural and type-safe implementation of Gaussian Elimination
    which exposes its design choices at code-generation time, so that
    these choices can effectively be specialized away, and where the
    resulting code is quite efficient.",
  paper = "Care06.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Care11,
  author = "Carette, Jacques and Kiselyov, Oleg",
  title = {{Multi-stage programming with functors and monads: eliminating
           abstraction overhead from generic code}},
  journal = "Sci. Comput. Program",
  volume = "76",
  number = "5",
  pages = "349-375",
  year = "2011",
  link = "\url{http://www.cas.mcmaster.ca/~carette/metamonads/metamonads.pdf}",
  abstract =
    "We use multi-stage programming, monads and OCaml's advanced module
    system to demonstrate how to eliminate all abstraction overhead from
    generic programs, while avoiding any inspection of the resulting code.
    We demonstrate this clearly with Gaussian Elimination as a
    representative family of symbolic and numeric algorithms. We
    parameterize our code to a great extent -- over domain, input and
    permutation matrix representations, determinant and rank tracking,
    pivoting policies, result types, etc. -- at no run-time cost. Because
    the resulting code is generated just right and not changed afterward,
    MetoOCaml guarantees that the generated code is well-typed. We further
    demonstrate that various abstraction parameters (aspects) can be made
    orthogonal and compositional, even in the presence of name-generation
    for temporaries, and ``interleaving'' of aspects. We also show how to
    encode some domain-specific knowledge so that ``clearly wrong''
    compositions can be rejected at or before generation time, rather than
    during the compilation or running of the generated code.",
  paper = "Care11.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Carp04,
  author = "Carpent, Quentin and Conil, Christophe",
  title = {{Utilisation de logiciels libres pour la r\'ealisation de TP MT26}},
  year = "2004",
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/ac20.pdf}",
  comment = "french",
  abstract = "radicalSolve(x**3+x**2-7=0,x)",
  paper = "Carp04.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Caru10,
  author = "Caruso, Fabrizio",
  title = {{Factorization of Non-Commutative Polynomials}},
  link = "\url{https://arxiv.org/pdf/1002.3108.pdf}",
  year = "2010",
  abstract =
    "We describe an algorithm for the factorization of non-commutative
    polynomials over a field. The first sketch of this algorithm appeared
    in an unpublished manuscript (literally hand written notes) by James
    H.  Davenport more than 20 years ago. This version of the algorithm
    contains some improvements with respect to the original sketch. An
    improved version of the algorithm has been fully implemented in the
    Axiom computer algebra system.",
  paper = "Caru10.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Cavi85,
  author = "Caviness, Bob",
  title = {{Computer Algebra: Past and Future}},
  journal = "LNCS",
  volume = "203",
  pages = "1-18",
  year = "1985",
  paper = "Cavi85.pdf",
  keywords = "axiomref, printed"
}
```

———-

— axiom.bib —

```
@misc{Cavi03,
  author = "Caviness, Bob and Trager, Barry and Gianni, Patrizia",
  title = {{Dedicated to the Memory of Richard Dimick Jenks}},
  year = "2003",
  link = "\url{https://www.eecis.udel.edu/~caviness/jenks/issac-ded.pdf}",
  abstract =
    "On December 30, 2003, Dick Jenks died at the age of 66, after an
    extended and courageous battle with multiple system atrophy.

    He received his PhD in mathematics from the University of Illinois at
    Urbana-Champaign in 1966. The title of his dissertation was
    Quadratic Differential Systems for Mathematical Models and was
    written under the supervision of Donald Gilles. After completing his
    PhD, he was a post-doctoral fellow at Brookhaven National Laboratory
    on Long Island. In 1968 he joined IBM Research where he worked until
    his retirement in 2002.

    At IBM he was one of the principal architects of the Scratchpad
    system, one of the earliest computer algebra systems(1971). Dick
    always believed that natural user interfaces were essential and
    developed a user-friendly rule-based system for Scratchpad. Although
    this rule-based approach was easy to use, as algorithms for computer
    algebra became more complicated, he began to understand that an
    abstract data type approach would give sophisticated algorithm
    development considerably more leverage. In 1977 he began the Axiom
    development (originally called Scratchpad II) with the design of
    MODLISP, a merger of Lisp with types (modes). In 1980, with the help
    of many others, he completed an initial prototype design based on
    categories and domains that were intended to be natural for
    mathematically sophisticated users.

    During this period many researchers in computer algebra visited IBM
    Research in Yorktown Heights and contributed to the development of
    the Axiom system. All this activity made the computer algebra group at
    IBM one of the leading centers for research in this area and Dick was
    always there to organize the visits and provide a stimulating and
    pleasant working environment for everyone. He had a good perspective
    on the most important research directions and worked to attract
    world-renowned experts to visit and interact with his group. He was an
    ideal manager for whom to work, one who always put the project and the
    needs of the group members first. It was a joy to work in such a
    vibrant and stimulating environment.

    After many years of development, a decision was made to rename
    Scratchpad II to Axiom and to release it as a product. Dick and
    Robert Sutor were the primary authors of the book {\sl Axiom: The
    Scientific Computation System}.  In the foreword of the book, written
    by David and Gregory Chudnovsky, it is stated that ``The Scratchpad
    system took its time to blossom into the beautiful Axiom
    product. There is no rival to this powerful environment in its scope
    and, most importantly, in its structure and organization.'' Axiom was
    recently made available as free software. See
    http://savannah.nongnu.org/projects/axiom
```

```
    Dick was active in service to the computer algebra community as
    well. Here are some highlights. He served as Chair of ACM SIGSAM
    (1979-81) and Conference Co-chair (with J. A. van Hulzen) of EUROSAM
    84, a precursor of the ISSAC meetings. Dick also had a long period of
    service on the editorial board of the {\sl Journal of Symbolic
    Computation}. At ISSAC 95 in Montreal, Dick was elected to the
    initial ISSAC Steering Committee and was elected as the second Chair
    of the Committee in 1997. He, along with David Chudnovsky, organized
    the highly successful meetings on Computers and Mathematics that were
    held at Stanford in 1986 and MIT in 1989. As a legacy of those
    meetings, the Jenks Prize for outstanding contributions to software
    engineering in computer algebra has been established.

    Dick had many interests outside of his professional pursuits including
    reading, travel, physical fitness, and especially music. Dick was an
    accomplished pianist, organist, and vocalist. At one point he was the
    organist and choir master of the Church of the Holy Communion in
    Mahopac, NY. In the 1980s and 1990s, he sang in choral groups under
    the direction of Dr. Dennis Keene that performed at Lincoln Center in
    New York city.

    Personally, Dick was warm, generous, and outgoing with many
    friends. He will be missed for his technical accomplishments, his
    artist talents, and most of all for his positive, gentle, charming
    spirit.",
  paper = "Cavi03.pdf",
  keywords = "axiomref",
  beebe = "Caviness:2004:MRD"
}
```

———-

— Caviness:2004:MRD —

```
@Article{Caviness:2004:MRD,
  author =       "Bob Caviness and Barbara Gatje and James H. Griesmer
                 and Tony Hearn and Manual Bronstein and Erich
                 Kaltofen",
  title =        {{In Memoriam: {Richard Dimick Jenks}: {Axiom} Developer
                 and Computer Algebra Pioneer}},
  journal =      j-SIGSAM,
  volume =       "38",
  number =       "1",
  pages =        "30--30",
  month =        mar,
  year =         "2004",
  CODEN =        "SIGSBZ",
  ISSN =         "0163-5824 (print), 1557-9492 (electronic)",
  ISSN-L =       "0163-5824",
  bibdate =      "Sat Apr 17 11:49:58 MDT 2004",
  bibsource =    "http://portal.acm.org/;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
```

```
  link =          "\url{http://savannah.nongnu.org/projects/axiom/}",
  acknowledgement = ack-nhfb,
  fjournal =     "SIGSAM Bulletin",
  issue =        "147",
}
```

———-

— **axiom.bib** —

```
@misc{Cert16,
  author = "Certik, Ondrej",
  title = {{SymPy vs. Axiom}},
  link = "\url{https://github.com/sympy/sympy/wiki/SymPy-vs.-Axiom}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Chan04,
  author = "Chan, L. and Cheb-Terrab, E.S.",
  title = {{Non-Liouvillian solutions for second order linear ODEs}},
  booktitle = "Proc. ISSAC 04",
  pages = "80-86",
  year = "2004",
  isbn = "1-58113-827-X",
  link = "\url{http://www.cecm.sfu.ca/CAG/papers/edgardoIS04.pdf}",
  abstract =
    "There exist sound literature and algorithms for computing Liouvillian
    solutions for the important problem of linear ODEs with rational
    coefficients. Taking as sample the 363 second order equations of that
    type found in Kamke's book, for instance, 51\% of them admit Liouvillian
    solutions and so are solvable using Kovacic's algorithm. On the other
    hand, special function solutions not admitting Liouvillian form appear
    frequently in mathematical physics, but there are not so general
    algorithms for computing them. In this paper we present an algorithm
    for computing special function solutions which can be expressed using
    the $_2F_1$, $_1F_1$ or $_0F_1$ hypergeometric functions. They algorithm
    is easy to implement in the framework of a computer algebra system and
    systematically solves 91\% of the 363 Kamke's linear ODE examples
    mentioned.",
  paper = "Chan04.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Chic04,
  author = "Chicha, Yannis and Lloyd, Michael Oancea, Cosmin",
  title = {{Parametric Polymorphism for Computer Algebra Software Components}},
  booktitle = "6th Int. Symp. on Symbolic and Numeric Algorithms for
               Scientific Computing",
  series = "SYNASC 04",
  location = "Imisoara, Romania",
  year = "2004",
  link = "\url{http://www.csd.uwo.ca/~watt/pub/reprints/2004-synasc-ppca.pdf}",
  abstract =
    "This paper presents our experiments in providing mechanisms for
    parametric polymorphism for computer algebra software components.
    Specific interfaces between Aldor and C++ and between Aldor and Maple
    are described. We then present a general solution, Generic IDL (GIDL),
    an extension to CORBA IDL supporting generic types. We describe our
    language bindings for C++, Java 1.5 and Aldor as well as aspects of
    our implementation, consisting of a GIDL to IDL compiler and tools for
    generating interface code for the various language bindings.",
  paper = "Chic04.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Chouxx,
  author = "Chou, Shang-Ching and Gao, Xiao-Shan and Liu, Zhuo-Jun and
            Wang, Ding-Kang and Wang, Dongming",
  title = {{Geometric Theorem Provers and Algebraic Equations Solvers}},
  comment = "Chapter 20",
  link = "\url{http://www.mmrc.iss.ac.cn/~xgao/papers/soft1.pdf}",
  abstract =
    "The methods of mechanizing mathematics are realized by means of
    compputer software for solving scientic and engineering problems via
    symbolic and hybrid computation. This chapter provides a collection of
    geometric theorem provers and algebraic equations solvers that are
    pieces of mathematical software based mostly on Wu's method and were
    developed mainly by members of the extended Wu group. The early
    theorem provers, though ecient, were written in basic programming
    languages and on primitive computers. Now there exist more powerful
    and mature geometric theorem provers of which some have already been
    published as commercial software. On the other hand, building
    effective equations solvers is still at the experimental stage and
    remains for further research and development.",
  paper = "Chouxx.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Chu85,
  author = "Chudnovsky, D.V and Chudnovsky, G.V.",
  title = {{Elliptic Curve Calculations in Scratchpad II}},
  year = "1985",
  institution = "Mathematics Dept., IBM Research",
  type = "Scratchpad II Newsletter 1 (1)",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Chudnovsky 87]{Chu87}
  author = "Chudnovsky, D.V and Chudnovsky, G.V.",
  title = {{New Analytic Methods of Polynomial Root Finding}},
  year = "1987",
  pages = "2",
  keywords = "axiomref"
in [Wit87]
```

———-

— **ignore** —

```
\bibitem[Chudnovsky 89]{Chu89}
  author = "Chudnovsky, D.V. and Chudnovsky, G.V.",
  title = {{The computation of classical constants}},
  year = "1989",
  month = "November",
  pages = "8178-8182",
Proc. Natl. Acad. Sci. USA Vol 86 ,
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@proceedings{CJ86,
  editor = "Chudnovsky, David and Jenks, Richard",
  title = {{Computers in Mathematics}},
  year = "1986",
  month = "July",
  isbn = "0-8247-8341-7",
  note = "International Conference on Computers and Mathematics",
  publisher = "Marcel Dekker, Inc",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@InProceedings{Coli95,
  author = "Colin, Antoine",
  title = {{Formal computation of Galois groups with relative resolvants}},
  booktitle = "Proc. AAECC-11",
  series = "AAECC-11",
  year = "1995",
  publisher = "Springer-Verlag",
  location = "Paris, France",
  pages = "169-182",
  abstract =
    "Let $k$ be a field and $f \in k[x]$ be a polynomial of degree $n$. The
    permutation action of $G$ on the roots $\{\alpha_i\}_{i=1}^n$ of $f$
    can be determined by an algorithm suggested by R. Stauduhar
    [Math. Comput. 27, 981-996 (1973; Zbl 0282.12004)] that approximates $G$
    via successive steps in a chain of subgroups
    $S_n=H_0 > H_1 > \ldots > H_k=G$. In each step $H_{i-1} > H_i$
    it checks as a test for $G \le H_i$ whether a relative invariant $k_i
    \in k[x_1,\ldots,x_n]$ yields a value under the specialization
    $\varphi : g(x_1,\ldots,x_n) \mapsto g(\alpha_1,\ldots,\alpha_n)$.  In
    implementations this evaluation has been done using $p$-adic
    [H. Darmon and D. Ford, Commun. Algebra 17, No. 12, 2941-2943 (1989;
    Zbl 0693.12010)] or numerical (R. Stauduhar [ibid.]; Y. Eichenlaub and
    M. Olivier [preprint]) approximation of the roots.

    The paper under review presents a new approach which avoids all
    approximations: If $G \le H_i$ and $H_i$ is maximal in $H_{i-1}$ the
    invariant $h_i$ is a primitive element of the invariant field
    $k_i=k(x_1,\ldots,x_n)^{H_i}$ as an extension of
    $k_{i-1}=K(x_1,\ldots,x_n)^{H_{i-1}}$.
    The author develops an algorithm to express the specialized values
    $\varphi(g)$ of elements $g \in k_i$ in terms of $k_{i-1}$ and the
    specialization $\varphi(h_i)$.

    This algorithm then is applied to the relative resolvent polynomial
    \[s_i = \prod_a{(y-a(x_1,\ldots,x_n))}\]
    where $a$ runs through the images of $h_i$ under $H_{i-1}$.
    It has $y$-coefficients which are in $k_{i-1}$.
    The algorithm then permits to express the coefficients of the
    specialization $r_i(y)=\varphi(s_i) \in k[y]$ recursively in the
    (already known) specializations $\varphi(h_i)$ for $j \le i-1$,
    using the coefficients of $f$ (as $S_n$-invariants in the roots)
    as a seed. A root of $r_i(y)$ in the base field then proves that $G$
    is contained in (a conjugate of) $H_i$, and this value of the root can
    be used as specialized $\varphi(h_{i+1})$ in the next step of the
    algorithm. Special care is given to the case when denominators of
    elements in $k(x_1,\ldots,x_n)$ evaluate to zero after specialization.

    The paper closes with a short discussion of applicability. An
    implementation using AXIOM and GAP is in process but has not yet been
    completed.",
  keywords = "axiomref"
```

}

———-

— **axiom.bib** —

```
@article{Coli97,
  author = "Colin, Antoine",
  title = {{Solving a system of algebraic equations with symmetries}},
  journal = "J. Pure Appl. Algebra",
  volume = "117-118",
  pages = "195-215",
  year = "1997",
  abstract =
    "Let $(F)$ be a system of $p$ polynomial equations
    $F_i({\bf X}) \in k[{\bf X}]$, where $k$ is a commutative field and
    ${\bf X} := (X_1,\cdots,X_n)$ are indeterminates. Let $G$ be a subgroup
    of $GL_n(k)$. A polynomial $P \in k[{\bf X}]$ (resp. rational function
    $P \in k({\bf X})$ ) is an invariant of $G$ if and only if for all
    $A \in G$ we have $A\cdot P = P$. We denote $k[{\bf X}]^G$ by (resp.
    $k({\bf X})^G$) the algebra of polynomial (resp. rational function)
    invariants of $G$. If $L$ is another subgroup of $GL_n(k)$ such that
    $G \subset L$, $P$ is called a primary invariant of $G$ relative to $L$ if
    and only if $Stab_L(P) = G$ (where $Stab_L(P)$ is the stabilizer of
    $P$ in $L$).

    The paper describes the algebra of the invariants of a finite group
    and how to express these invariants in terms of a small number of
    them, from both the Cohen-Macaulay algebra and the field theory points
    of view. A method is proposed to solve $(F)$ by expressing it in terms of
    primary invariants $\Pi_1,\cdots,\Pi_n$
    (e.g. the elementary symmetric polynomials) and one
    ``primitive'' secondary invariant.

    The main thrust of the paper is contained in the following theorem.
    Let $(F)$ be a set of invariants of $G$. Let $L$ be a subgroup of
    $GL_n(k)$ such that $G \subset L$ and $k({\bf X})^L$ is a purely
    transcendental extension of $k_i$, let $\Pi_1,\cdots,\Pi_n$ be
    polynomials such that $k({\bf X})^L = k(\Pi_1,\cdots,\Pi_n)$,
    and let $\Theta \in k[{\bf X}]^G$ be a primitive polynomial invariant
    of $G$ relative to $L$.
    When possible, it is convenient to choose $\Theta$ to be one of the
    polynomials in $(F)$.  An algorithm is given that allows each polynomial
    $F_i$ to be expressed as $F_i({\bf X}) = H_i(\Pi_1,\cdots,\Pi_n,\Theta)$,
    an algebraic fraction in $\Pi_1,\cdots,\Pi_n$ and a polynomial in
    $\Theta$. Now let $L$ be the minimal polynomial of $\Theta$ over
    $k[{\bf X}]^L$; we have
    \[L({\bf X},T)=\prod_{\Theta^{'} \in L\cdot \Theta}(T-\Theta^{'})
    \in k[{\bf X}]^L[T]\]
    (where $L$ is called a generic Lagrange resolvent).
    As $k(\Pi_1,\cdots,\Pi_n)=k({\bf X})^L$, we can write
    $L({\bf X},T)=H_0(\Pi_1,\cdots,\Pi_n,T)$ where $H_0$ is some
    rational function. The question
```

```
    $H_0(\Pi_1,\cdots,\Pi_n,\Theta)=0$ is always satisfied because
    $\Theta$ is a root of $L$. Then, we solve the system of ($p=1$)
    algebraic equations $H_i(\Pi_1,\cdots,\Pi_n,\Theta)=0$,
    $0 \le i \le p$ for $\Pi_1,\cdots,\Pi_n,\Theta$ as indeterminates.

    Theorem 1: Let $D \in k[\Pi_1,\cdots,\Pi_n]$ be the LCM of the
    denominators of all the fractions $H_i$,$0 \le i \le p$ and let
    $H_i^{'}=DH_i$. For every solution
    $x:=(x_1,\cdots,x_n)$ of the system $(F)$:$F_i({\bf X})=0$,
    $1 \le i \le p$, there exists a solution ($\pi_1,\cdots,\pi_n,\Theta$)
    of the system
    $(H^{'}):H_i^{'}(\Pi_1,\cdots,\Pi_n,\Theta)=0$, $0 \le i \le p$ such
    that $x$ is a solution of the system
    $(P_\pi):\Pi_i({\bf X})=\pi_i$, $1 \le i \le n$ , and of the equation
    $\Theta({\bf X})=0$. Conversely, for any solution
    $(\pi_1,\cdots,\pi_n,\theta)$ of the system $(H^{'})$ such that
    $D(\pi_1,\cdots,\pi_n) \ne 0$, if $x$ is a solution of the system
    $(P_\pi)$ relative to $(\pi_1,\cdots,\pi_n)$, then there exists
    some $A \in L$ such that $\Theta(A\cdot x)=\theta$, and then for all
    $B \in G$, $BA\cdot x$, is a solution of the system $(F)$.

    A slighly more general version of this theorem is also given. The
    paper then presents an algorithm that applies the theory and has been
    implemented in AXIOM. It is followed by several examples.",
  keywords = "axiomref"
}
```

— axiom.bib —

```
@inbook{Coll82,
  author = "Collins, G.E. and Mignotte, M. and Winkler, F.",
  title = {{Arithmetic in Basic Algebraic Domains}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  isbn = "978-3-211-81684-4",
  pages = "189-220",
  year = "1982",
  abstract =
    "This chapter is devoted to the arithmetic operations, essentially
    addition, multiplication, exponentiation, division, gcd calculations
    and evaluation, on the basic algebraic domains. The algorithms for
    these basic domains are those most frequently used in any computer
    algebra system. Therefore the best known algorithms, from a
    computational point of view, are presented. The basic domains
    considered here are the rational integers, the rational numbers,
    integers modulo $m$, Gaussian integers, polynomials, rational
    functions, power series, finite fields and $p$-adic numbers. BOunds on
    the maximum, minimum and average computing time ($t^{+},t^{-},t^{*}$) for
    the various algorithms are given.",
  paper = "Buch82.pdf"
}
```

———-

— **axiom.bib** —

```
@inbook{Coll82b,
  author = "Collins, G.E.",
  title = {{Quantifier Elimination for Real Closed Fields: A Guide to
            the Literature}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "79-82",
  abstract =
    "This article provides a brief summary of the most important
    publications relating to quantifier elimination for the elementary
    theory of real closed fields. Especially mentioned is the
    cylindrical algebraic decomposition method and its relation to the
    facilities of computer algebra facilities.",
  paper = "Buch82.pdf"
}
```

———-

— **axiom.bib** —

```
@inbook{Coll82c,
  author = "Collins, G.E. and Loos, R.",
  title = {{Real Zeros of Polynomials}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "83-94",
  abstract =
    "Let $A$ be a polynomial over $\mathbb{Z}$, $\mathbb{Q}$,
    $\mathbb{Q}(\alpha)$ where $\alpha$ is a real algebraic
    number. The problem is to compute a sequence of disjoint intervals
    with rational endpoints, each containing exactly one real zero of
    $A$ and together containing all real zeros of $A$. We describe an
    algorithm due to Kronecker based on the minimum root separation,
    Sturm's algorithm, an algorithm based on Rolle's theorem due to
    Collins and Loos and the modified Uspensky algorithm due to
    Collins and Aritas. For the last algorithm a recursive version
    with correctness proof is given which appears in print for the
    first time.",
  paper = "Buch82.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Cohe03,
  author = "Cohen, Arjeh and Cuypers, H. and Barreiro, Hans and
            Reinaldo, Ernesto and Sterk, Hans",
  title = {{Interactive Mathematical Documents on the Web}},
  year = "2003",
  pages = "289-306",
  editor = "Joswig, M. and Takayma, N.",
  publisher = "Springer-Verlag, Berlin, Germany",
  misc = "in Algebra, Geometry and Software Systems",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Cohen 91]{CC91} Cohen, G.; Charpin, P.; (ed)
EUROCODE '90 International Symposium on
Coding Theory and Applications Proceedings. Springer-Verlag, Berlin, Germany
/ Heidelberg, Germany / London, UK / etc., 1991 ISBN 0-387-54303-1
(New York), 3-540-54303-1 (Berlin), LCCN QA268.E95 1990
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@book{Cohe03a,
  author = "Cohen, Joel S.",
  title = {{Computer algebra and symbolic computation. Mathematical Methods}},
  year = "2003",
  link = "\url{http://eclass.uth.gr/eclass/modules/document/file.php/MHX102/Cohen\_Computer\_Algebra\_and\_Symbo
  publisher = "A. K. Peters",
  isbn = "1-56881-159-4",
  paper = "Cohe03a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Cohe03b,
  author = "Cohen, Joel S.",
  title = {{Computer algebra and symbolic computation. Elementary Algorithms}},
  year = "2003",
  publisher = "A. K. Peters",
  isbn = "1-56881-159-4",
  paper = "Cohe03b.pdf",
```

```
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Conrxxa,
  author = "Conrad, Marc and French, Tim and Maple, Carsten and Pott, Sandra",
  title = {{Approaching Inheritance from a Natural Mathematical Perspective
           and from a Java Driven Viewpoint: a Comparative Review}},
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/McTfCmSp-axiom.pdf}",
  abstract = "
    It is well-known that few object-oriented programming languages allow
    objects to change their nature at run-time. There have been a number
    of reasons presented for this, but it appears that there is a real
    need for matters to change. In this paper we discuss the need for
    object-oriented programming languages to reflect the dynamic nature of
    problems, particularly those arising in a mathematical context. It is
    from this context that we present a framework that realistically
    represents the dynamic and evolving characteristic of problems and
    algorithms.",
  paper = "Conrxxa.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Conrxxb,
  author = "Conrad, Marc and French, Tim and Maple, Carsten and Pott, Sandra",
  title = {{Mathematical Use Cases lead naturally to non-standard Inheritance
        Relationships: How to make them accessible in a mainstream language?}},
  abstract = "
    Conceptually there is a strong correspondence between Mathematical
    Reasoning and Object-Oriented techniques. We investigate how the ideas
    of Method Renaming, Dynamic Inheritance and Interclassing can be used
    to strengthen this relationship. A discussion is initiated concerning
    the feasibility of each of these features.",
  paper = "Conrxxb.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Corl00,
  author = "Corless, Robert M. and Jeffrey, David J. and Watt, Stephen M. and
           Davenport, James H.",
```

```
  title = {{``According to Abramowitz and Stegun'' or
            arccoth needn't be Uncouth}},
  journal = "SIGSAM Bulletin - Special Issue on OpenMath",
  volume = "34",
  number = "2",
  pages = "58-65",
  year = "2000",
  algebra =
   "\newline\refto{category OM OpenMath}
    \newline\refto{domain COMPLEX Complex}
    \newline\refto{domain DFLOAT DoubleFloat}
    \newline\refto{domain FLOAT Float}
    \newline\refto{domain FRAC Fraction}
    \newline\refto{domain INT Integer}
    \newline\refto{domain LIST List}
    \newline\refto{domain SINT SingleInteger}
    \newline\refto{domain STRING String}
    \newline\refto{domain SYMBOL Symbol}
    \newline\refto{package OMEXPR ExpressionToOpenMath}
    \newline\refto{package OMSERVER OpenMathServerPackage}",
  abstract =
    "This paper addresses the definitions in OpenMath of the elementary
    functions. The original OpenMath definitions, like most other sources,
    simply cite [2] as the definition. We show that this is not adequate,
    and propose precise definitions, and explore the relationships between
    these definitions.In particular, we introduce the concept of a couth
    pair of definitions, e.g. of arcsin and arcsinh, and show that the
    pair arccot and {\sl arccoth} can be couth.",
  paper = "Corl00.pdf"
}
```

———-

— axiom.bib —

```
@book{Coxx07,
  author = "Cox, David and Little, John and O'Shea, Donald",
  title = {{Ideals, varieties and algorithms. An introduction to computational
           algebraic geometry and commutative algebra}},
  publisher = "Springer",
  isbn = "978-0-387-35650-1",
  year = "2007",
  link = "\url{http://www.dm.unipi.it/~caboara/Misc/Cox,\%20Little,\%20O'Shea\%20-\%20Ideals,\%20varieties\%20an
  algebra = "\newline\refto{package GB GroebnerPackage}
             \newline\refto{package PSEUDLIN PseudoLinearNormalForm}
             \newline\refto{package PGROEB PolyGroebner}
             \newline\refto{domain DMP DistributedMultivariatePolynomial}
    \newline\refto{domain GDMP GeneralDistributedMultivariatePolynomial}
    \newline\refto{domain HDMP HomogeneousDistributedMultivariatePolynomial}",
  abstract =
    "Around 1980 two new directions in science and technique came
    together. One was Buchbergers algorithms in order to handle Groebner
    bases in an effective way for solving polynomial equations. The second
```

```
    one was the development of the personal computers. This was the
    starting point of a computational perspective in commutative algebra
    and algebraic geometry. In 1991 the three authors invented the first
    edition of their book as an introduction for undergraduates to some
    interesting ideas in commutative algebra and algebraic geometry with a
    strong perspective to practical and computational aspects. A second
    revised edition appeared in 1996. That means from the very beginning
    the book provides a bridge for the new, computational aspects in the
    field of commutative algebra and algebraic geometry.

    To be more precise, the book gives an introduction to Buchbergers
    algorithm with applications to syzygies, Hilbert polynomials, primary
    decompositions. There is an introduction to classical algebraic
    geometry with applications to the ideal membership problem, solving
    polynomial equations, and elimination theory. Some more spectacular
    applications are about robotics, automatic geometric theorem proving,
    and invariants of finite groups. It seems to the reviewer to carry
    coals to Newcastle for estimating the importance and usefulness of the
    book. It should be of some interest to ask how many undergraduates
    have been introduced to algorithmic aspects of commutative algebra and
    algebraic geometry following the line of the book. The reviewer will
    be sure that this will continue in the future too.

    What are the changes to the previous editions? There is a significant
    shorter proof of the Extension Theorem, see 3.6 in Chapter 3,
    suggested by A.H.M. Levelt. A major update has been done in Appendix C
    ``Computer Algebra Systems''. This concerns in the main the section
    about MAPLE. Some minor updated information concern the use of AXIOM,
    CoCoA, Macaulay2, Magma, Mathematica, and SINGULAR. This reflects
    about the recent developments in Computer Algebra Systems. It
    encourages an interested reader to more practical exercises. The
    authors have made changes on over 200 pages to enhance clarity and
    correctness. Many individuals have reported typographical errors and
    gave the authors feedback on the earlier editions. The book is
    well-written. The reviewer guesses that it will become more and more
    difficult to earn 1 dollar (sponsored by the authors) for every new
    typographical error as it was the case also with the first and second
    edition. The reviewer is sure that it will be a excellent guide to
    introduce further undergraduates in the algorithmic aspect of
    commutative algebra and algebraic geometry.",
  paper = "Coxx07.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Crou95,
  author = "Crouch, Peter E. and Lamnabhi-Lagarrigue, Francoise and
            Pinchon, Didier",
  title = {{Some realizations of algorithms for nonlinear input-output
            systems}},
```

```
  journal = "Int. J. Control",
  volume = "62",
  number = "4",
  pages = "941-960",
  year = "1995",
  abstract =
    "The first two authors previously developed an algorithm for
    constructing a parametrization of the observation space of a nonlinear
    control system directly from the differential equation representation
    of the input-output behaviour. This paper extends the previous
    algorithm by including settings where a set of implicit input-output
    differential equations is given as well as more general state-space
    representations in which the controls enter nonlinearly. Various
    state-space realizations, including bilinear, polynomial and nilpotent
    approximating realizations are discussed. The final section of the
    paper sketches the implementation of the algorithm using the symbolic
    manipulation package AXIOM to find the realizations mentioned above in
    feasible cases.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Cuyp10,
  author = "Cuypers, Hans and Hendriks, Maxim and Knopper, Jan Willem",
  title = {{Interactive Geometry inside MathDox}},
  year = "2010",
  link = "\url{http://www.win.tue.nl/~hansc/MathDox_and_InterGeo_paper.pdf}",
  paper = "Cuyp10",
  keywords = "axiomref"
}
```

———-

1.40.4 D

— **axiom.bib** —

```
@inproceedings{Dalm97,
  author = {Dalmas, St\'ephane and Ga\"etano, Marc and Watt, Stephen},
  title = {{An OpenMath 1.0 Implementation}},
  booktitle = "Proc. 1997 Int. Symp. on Symbolic and Algebraic Computation",
  series = "ISSAC'97",
  year = "1997",
  isbn = "0-89791-875-4",
  location = "Kihei, Maui, Hawaii, USA",
  pages = "241-248",
  numpages = "8",
  link = "\url{http://doi.acm.org/10.1145/258726.258794}",
```

```
  doi = "10.1145/258726.258794",
  acmid = "258794",
  publisher = "ACM, New York, NY USA",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Dalm92,
  author = "Dalmas, Stephane",
  title = {{A polymorphic functional language applied to symbolic
           computation}},
  year = "1992",
  booktitle = "Proc. ISSAC 1992",
  series = "ISSAC 1992",
  pages = "369-375",
  isbn = "0-89791-489-9 (soft cover) 0-89791-490-2 (hard cover)",
  abstract =
    "The programming language in which to describe mathematical objects
    and algorithms is a fundamental issue in the design of a symbolic
    computation system.  XFun is a strongly typed functional programming
    language. Although it was not designed as a specialized language, its
    sophisticated type system can be successfully applied to describe
    mathematical objects and structures. After illustrating its main
    features, the author sketches how it could be applied to symbolic
    computation. A comparison with Scratchpad II is attempted. XFun seems
    to exhibit more flexibility simplicity and uniformity.",
  paper = "Dalm92.pdf",
  keywords = "axiomref",
  beebe = "Dalmas:1992:PFL"
}
```

———-

— **Dalmas:1992:PFL** —

```
@InProceedings{Dalmas:1992:PFL,
  author =       "S. Dalmas",
  title =        {{A polymorphic functional language applied to symbolic
                 computation}},
  crossref =     "Wang:1992:ISS",
  pages =        "369--375",
  month =        "",
  year =         "1992",
  bibdate =      "Tue Sep 17 06:35:39 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The programming language in which to describe
                 mathematical objects and algorithms is a fundamental
                 issue in the design of a symbolic computation system.
                 XFun is a strongly typed functional programming
```

```
                language. Although it was not designed as a specialized
                language, its sophisticated type system can be
                successfully applied to describe mathematical objects
                and structures. After illustrating its main features,
                the author sketches how it could be applied to symbolic
                computation. A comparison with Scratchpad II is
                attempted. XFun seems to evaluate more flexibility
                simplicity and uniformity.",
  acknowledgement = ack-nhfb,
  affiliation =  "Inst. Nat. de Recherche d'Inf. et d'Autom., Valbonne,
                France",
  classification = "C6140D (High level languages); C7310 (Mathematics)",
  keywords =     "Mathematical objects; Polymorphic functional language;
                Scratchpad II; Symbolic computation; XFun",
  language =     "English",
  thesaurus =    "Functional programming; High level languages; Symbol
                manipulation",
}
```

———-

— **axiom.bib** —

```
@misc{Daly08,
  author = "Daly, Timothy",
  title = {{Axiom Computer Algebra System Information Sources}},
  video = "https://www.youtube.com/watch?v=CV8y3UrpadY",
  year = "2008",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Daly88,
  author = "Daly, Timothy",
  title = {{Axiom in an Educational Setting, Axiom course slide deck}},
  year = "1988",
  month = "January",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Daly02,
  author = "Daly, Timothy",
  title = {{Axiom as open source}},
  journal = "SIGSAM Bulletin",
```

```
  volume = "36",
  number = "1",
  pages = "28-28",
  month = "March",
  year = "2002",
  keywords = "axiomref",
  beebe = "Daly:2002:AOS"
}
```

———-

— **Daly:2002:AOS** —

```
@Article{Daly:2002:AOS,
  author =       "T. Daly",
  title =        {{Axiom as Open Source}},
  journal =      j-SIGSAM,
  volume =       "36",
  number =       "1",
  pages =        "28--??",
  month =        mar,
  year =         "2002",
  CODEN =        "SIGSBZ",
  ISSN =         "0163-5824 (print), 1557-9492 (electronic)",
  ISSN-L =       "0163-5824",
  bibdate =      "Mon Apr 29 07:16:09 MDT 2002",
  bibsource =    "http://portal.acm.org/;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  fjournal =     "SIGSAM Bulletin",
  issue =        "139",
}
```

———-

— **axiom.bib** —

```
}
@misc{Daly03,
  author = "Daly, Timothy",
  title = {{Axiom Website: http://axiom-developer.org}},
  link = "\url{http://axiom-developer.org}",
  keywords = "axiomref",
  year = "2003"
}
```

———-

— **axiom.bib** —

```
@misc{Daly05,
  author = "Daly, Timothy",
```

```
  title = {{LispNYC Presentation at Trinity}},
  year = "2005",
  month = "May",
  day = "10",
  abstract =
    "Timothy Daly presents Axiom

    Timothy Daly, published author, academic researcher, open source
    programmer and lead developer of Axiom will be presenting about his role
    as the driving force behind Axiom.  With over 70 developers and 200
    researchers worldwide it can best be described as:

    Axiom is a general purpose Computer Algebra system. It is useful
    for research and development of mathematical algorithms providing
    a very high level way to express abstract mathematical concepts.
    The Axiom Library defines over 1,000 strongly-typed mathematical
    domains and categories.

    Axiom consists of an interpreter and compiler, a browser, a graphical
    interface, and a new online wiki that allows users to create web pages
    that inline computations.

    Axiom is built upon Common Lisp.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Dalyxx,
  author = "Daly, Timothy",
  title = {{Tim Daly on Lisp in Industry}},
  link = "\url{https://news.ycombinator.com/item?id=1580904}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Daly06,
  author = "Daly, Timothy",
  title = {{Axiom Volume 1: Tutorial}},
  publisher = "Lulu, Inc.",
  address = "860 Aviation Parkway, Suite 300, Morrisville, NC 27560 USA",
  year = "2006",
  isbn = "1-4116-6597-X",
  link = "\url{http://www.lulu.com/content/190827}",
  keywords = "axiomref",
  beebe = "Daly:2006:AVA"
}
```

———-

— **Daly:2006:AVA** —

```
@Book{Daly:2006:AVA,
  author =       "Timothy Daly",
  title =        {{Axiom Volume 1: {Axiom} Tutorial}},
  publisher =    "Lulu, Inc.",
  address =      "860 Aviation Parkway, Suite 300, Morrisville, NC
                 27560, USA",
  pages =        "iv + 285",
  year =         "2006",
  ISBN =         "1-4116-6597-X",
  ISBN-13 =      "978-1-4116-6597-2",
  LCCN =         "????",
  bibdate =      "Thu Mar 23 05:24:19 2006",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  price =        "US\$15.00",
  link =          "\url{http://www.lulu.com/content/190827}",
  acknowledgement = ack-nhfb,
}
```

———-

— **ignore** —

```
\bibitem[Daly 09]{Dal09} Daly, Timothy
  title = {{The Axiom Literate Documentation}},
  link = "\url{http://axiom-developer.org/axiom-website/documentation.html}",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Daly 13]{Dal13} Daly, Timothy
  title = {{Literate Programming in the Large}},
April 8-9, 2013 Portland Oregon
  url1 = "http://conf.writethedocs.org",
  url2 = "http://daly.axiom-developer.org",
  link = "\url{http://www.youtube.com/watch?v=AvOPQDVTP4A}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Daly13a,
  author = "Daly, Timothy and Barnes, Nick",
  title = {{Ten reasons you must publish your code}},
```

```
  year = "2013",
  link = "\url{http://climatecode.org/blog/2013/07/ten-reasons-you-must-publish-your-code/}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Dave90a,
  author = "Davenport, James H.",
  title = {{Current Problems in Computer Algebra Systems Design}},
  journal = "LNCS",
  volume = "429",
  year = "1990",
  pages = "1-9",
  abstract =
    "Computer Algebra systems have been with us for over twenty years, but
    there is still no consensus on what an ‘‘ideal’’ system would look
    like.  There are all sorts of tradeoffs between portability,
    functionality, and efficiency. This paper discusses some of those
    issues.",
  paper = "Dave90a.pdf",
  keywords = "axiomref, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@techreport{Dave92e,
  author = "Davenport, James H.",
  title = {{The AXIOM System}},
  type = "technical report",
  institution = "Numerical Algorithms Group, Oxford, U.K.",
  number = "TR5/92",
  year = "1992",
  paper = "Dave92a.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Dave99,
  author = "Davenport, James",
  title = {{A Small OpenMath Type System}},
  year = "1999",
  link = "\url{https://www.openmath.org/standard/sts.pdf}",
  paper = "Dave99.pdf",
```

```
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Dave05,
  author = "Davenport, James H.",
  title = {{Integration -- What do we want from the theory?}},
  booktitle = "Computer Algebra",
  publisher = "Springer",
  series = "Lecture Notes in Computer Science 162",
  pages = "2-11",
  year = "2005",
  abstract =
    "The theory of integration has moved a long way in the last fourteen
    years, thought not far enough to satisfy the demands placed on it by
    its customers. This paper outlines what problems have yet to be solved,
    and tries to explain why they are not trivial."
}
```

———-

— **ignore** —

```
\bibitem[Davenport 79a]{Dav79a} Davenport, J.H.
  title = {{What can SCRATCHPAD/370 do?}},
VM/370 SPAD.SCRIPTS August 24, 1979 SPAD.SCRIPT
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@techreport{Dave80,
  author = "Davenport, James H. and Jenks, Richard D.",
  title = {{MODLISP: A Preliminary Design}},
  institution = "IBM Research",
  type = "Research Report",
  year = "1980",
  number = "RC 8073",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Dave80a,
  author = "Davenport, James H. and Jenks, Richard D.",
```

```
  title = {{MODLISP}},
  institution = "IBM Research",
  type = "Research Report",
  year = "1980",
  number = "RC 8537 (\#37198)",
  comment = "http://www.computerhistory.org/collections/catalog/102719109",
  paper = "Dave80a.pdf",
  keywords = "axiomref, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Dave81a,
  author = "Davenport, James H. and Jenks, Richard D.",
  title = {{MODLISP}},
  year = "1981",
  journal = "ACM SIGSAM Bulletin",
  volume = "15",
  issue = "1",
  pages = "11-20",
  publisher = "ACM",
  abstract =
    "This paper discusses the design and implementation of MODLISP, a
    LISP-like language enhanced with the idea of MODes. This extension
    permits, but does not require, the user to declare the types of
    various variables, and to compile functions with the arguments
    declared to be of a particular type. It is possible to declare several
    functions of the same name, with arguments of different type
    (e.g. PLUS could be declared for Integer arguments, or Rational, or
    Real, or even Polynomial arguments) and the system will apply the
    correct function for the types of the arguments.",
  paper = "Dave81a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@manual{Dave84,
  author = "Davenport, James H. and Gianni, Patrizia and Jenks, Richard D. and
            Miller, Victor and Morrison, Scott C. and Rothstein, Michael and
            Sundaresan, Christine and Sutor, Robert S. and Trager, Barry M.",
  title = {{Scratchpad}},
  organization = "Mathematical Sciences Department",
  address = "IBM Thomas Watson Research Center, Yorktown Heights, NY",
  year = "1984",
  keywords = "axiomref",
  beebe = "Davenport:1984:S"
}
```

———-

— **Davenport:1984:S** —

```
@Manual{Davenport:1984:S,
  author =       "J. Davenport and P. Gianni and R. Jenks and V. Miller
                 and S. Morrison and M. Rothstein and C. Sundaresan and
                 R. Sutor and B. Trager",
  title =        {{Scratchpad}},
  organization = "Mathematical Sciences Department",
  address =      "IBM Thomas Watson Research Center",
  pages =        "??",
  year =         "1984",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.1.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— **axiom.bib** —

```
@misc{Dave84a,
  author = "Davenport, James H.",
  title = {{A New Algebra System}},
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/Davenport-1984-a\_new\_algebra\_system.pdf}",
  abstract = "Seminal internal paper discussing Axiom design decisions.",
  paper = "Dave84a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Dave84b,
  author = "Davenport, James H. and Gianni, Patrizia and Jenks, Ricard D. and
            Miller, Victor and Morrison, Scott and Rothstein, Michael and
            Sundaresan, Christine J. and Sutor, Robert S. and Trager, Barry",
  title = {{SCRATCHPAD System Programming Language Manual}},
  year = "1984",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Dave85,
  author = "Davenport, James H.",
  title = {{The LISP/VM Foundation of Scratchpad II}},
```

```
  journal = "The Scratchpad II Newsletter",
  volume = "1",
  number = "1",
  year = "1985",
  month = "September",
  institution = "IBM Research",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@book{Dave88,
  author = "Davenport, James H. and Siret, Y. and Tournier, E.",
  title = {{Computer Algebra: Systems and Algorithms for Algebraic
           Computation}},
  publisher = "Academic Press",
  year = "1988",
  isbn ="0-12-204230-1",
  link = "\url{http://staff.bath.ac.uk/masjhd/masternew.pdf}",
  paper = "Dave88.pdf",
  keywords = "axiomref, shelf",
  beebe = "Davenport:1988:CA"
}
```

———-

— Davenport:1988:CA —

```
@Book{Davenport:1988:CA,
  author =       "J. H. Davenport and Y. Siret and E. Tournier",
  title =        {{Computer Algebra: Systems and Algorithms for Algebraic
                 Computation}},
  publisher =    pub-AP,
  address =      pub-AP:adr,
  pages =        "xix + 267",
  year =         "1988",
  ISBN =         "0-12-204230-1",
  ISBN-13 =      "978-0-12-204230-0",
  LCCN =         "QA155.7.E4 D38 1988",
  bibdate =      "Fri Dec 29 18:14:51 1995",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  notes =        "{\footnotesize Dies ist die englische Ausgabe des
                 urspr{\"u}nglich bei Masson 1987 erschienen Buches {\em
                 Calcul Formel}. Es ist die erste Monographie {\"u}ber
                 Computeralgebra. Es wird in etwa die Theorie behandelt,
                 die heute in den gr{\"o}{\ss}eren Systemen wie MACSYMA,
                 MAPLE, REDUCE oder SCRATCHPAD II realisiert ist. Das
                 erste Kapitel ist der Diskussion verschiedener \CA\
                 Systeme mit Beispielen gewidmet. Die wichtige Frage der
                 Repr{\"a}sentation der mathematischen Objekte auf einem
```

```
                Computer ist das Thema des zweiten Kapitels. Der
                Algorithmus von Buchberger, zylindrische Dekomposition,
                Berechnung von gr{\"o}{\ss}ten gemeinsamen Teilern,
                p-adische Methoden und Faktorisierung,
                Differentialgleichungen und Stammfunktionen sind die
                wichtigsten behandelten Gegenst{\"a}nde des Buches, das
                mit einer ausf{\"u}hrlichen Bibliographie und einer
                Beschreibung von REDUCE im Anhang endet. \hfill J.
                Grabmeier}",
}
```

———-

— **axiom.bib** —

```
@techreport{Dave89,
  author = "Davenport, James H.",
  title = {{Looking at a set of equations}},
  institution = "University of Bath, School of Mathematical Sciences",
  year = "1989",
  type = "technical report",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.261.767}",
  abstract =
    "This working paper describes our experiences with using the
    Groebner-basis method [Buchberger, 1985] to solve some related systems
    of polynomial equations. While we have not yet been able to solve the
    system that was our primary motivation, we feel that these experiences
    may prove useful to others investigating Buchberger's algorithm in
    this context, especially when, as is the case for the system under
    investigation, the equations are highly structured. We conclude with
    some examples of the polynomials that we factored in the course of
    this investigation.",
  paper = "Dave89.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Dave93,
  author = "Davenport, James H.",
  title = {{The PoSSo Project}},
  paper = "Dave93.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Dave00,
```

```
  author = "Davenport, James H.",
  title = {{Abstract data types in computer algebra}},
  booktitle = "Mathematical foundations of computer science",
  series = "MFCS 2000",
  year = "2000",
  location = "Bratislava, Slovakia",
  pages = "21-35",
  abstract =
    "The theory of abstract data types was developed in the late 1970s and
    the 1980s by several people, including the ``ADJ'' group, whose work
    influenced the design of Axiom. One practical manifestation of this
    theory was the OBJ-3 system. An area of computing that cries out for
    this approach is computer algebra, where the objects of discourse are
    mathematical, generally satisfying various algebraic rules. There have
    been various theoretical studies of this in the literature. The aim of
    this paper is to report on the practical applications of this theory
    within computer algebra, and also to outline some of the theoretical
    issues raised by this practical application. We also give a
    substantial bibliography.",
  paper = "Dave00.pdf",
  keywords = "axiomref, printed"
}
```

———-

— axiom.bib —

```
@article{Dave02,
  author = "Davenport, James H.",
  title = {{Equality in computer algebra and beyond}},
  journal = "J. Symbolic Computing",
  volume = "34",
  number = "4",
  pages = "259-270",
  year = "2002",
  link =
    "\url{http://www.calculemus.net/meetings/siena01/Papers/Davenport.pdf}",
  abstract =
    "Equality is such a fundamental concept in mathematics that, in
    fact, we seldom explore it in detail, and tend to regard it as
    trivial. When it is shown to be non-trivial, we are often
    surprised. As is often the case, the computerization of
    mathematical computation in computer algebra systems on the one
    hand, and mathematical reasoning in theorem provers on the other
    hand, forces us to explore the issue of equality in greater
    detail.In practice, there are also several ambiguities in the
    definition of equality. For example, we refer to $\mathbb{Q}(x)$
    as ``rational functions'', even though $\frac{x^2-1}{x-1}$ and
    $x+1$ are not equal as functions from $\mathbb{R}$ to
    $\mathbb{R}$, since the former is not defined at $x=1$, even
    though they are equal as elements of $\mathbb{Q}(x)$. The aim of
    this paper is to point out some of the problems, both with
    mathematical equality and with data structure equality, and to
```

```
    explain how necessary it is to keep a clear distintion between the two.",
  paper = "Dave02.pdf",
  keywords = "axiomref, CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Dave07,
  author = "Davenport, James H. and Fitch, John",
  title = {{Computer Algebra and the three 'E's: Efficiency, Elegance, and
           Expressiveness}},
  link = "\url{http://staff.bath.ac.uk/masjhd/Drafts/PLMMS2007}",
  abstract =
    "What author of a programming language would not claim that the 3 'E's
    were the goals? Nevertheless, we claim that computer algebra does lead
    to particular emphases, and constraints, in these areas.

    We restrict ``efficiency'' to mean machine efficiency, since the other
    'E's cover programmer efficiency. For the sake of clarity, we describe
    as ``expressiveness'', what can be expressed in the language, and
    ``elegance'' as how it can be expressed.",
  paper = "Dave07.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Davenport 14]{Dav14} Davenport, James H.
  title = {{Computer Algebra textbook}},
  link = "\url{http://staff.bath.ac.uk/masjhd/JHD-CA.pdf}",
  paper = "Dav14.pdf",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Davenport 89]{Dav89} Davenport, J.H. (ed)
EUROCAL '87 European Conference on Computer Algebra Proceedings
Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London,
UK / etc., 1989 ISBN 3-540-51517-8 LCCN QA155.7.E4E86 1987
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@InProceedings{Dave90,
  author = "Davenport, James H. and Trager, Barry M.",
  title = {{Scratchpad's view of algebra I: Basic commutative algebra}},
  booktitle = "Design and Implementation of Symbolic Computation Systems",
  year = "1990",
  pages = "40-54",
  series = "DISCO '90",
  location = "Capri, Italy",
  publisher = "Springer-Verlag",
  isbn = "0-387-52531-9",
  link = "\url{http://opus.bath.ac.uk/32336/1/Davenport\_DISCO\_1990.pdf}",
  comment = "AXIOM Technical Report, ATR/1, NAG Ltd., Oxford, 1992",
  keywords = "axiomref",
  abstract =
    "While computer algebra systems have dealt with polynomials and
    rational functions with integer coefficients for many years, dealing
    with more general constructs from commutative algebra is a more recent
    problem. In this paper we explain how one system solves this problem,
    what types and operators it is necessary to introduce and, in short,
    how one can construct a computational theory of commutative
    algebra. Of necessity, such a theory is rather different from the
    conventional, non-constructive, theory. It is also somewhat different
    from the theories of Seidenberg [1974] and his school, who are not
    particularly concerned with practical questions of efficiency.",
  paper = "Dave90.pdf",
  keywords = "axiomref",
  beebe = "Davenport:1990:SVA"
}
```

———-

— Davenport:1990:SVA —

```
@InProceedings{Davenport:1990:SVA,
  author =       "J. H. Davenport and B. M. Trager",
  title =        {{Scratchpad's View of Algebra {I}: Basic Commutative
                 Algebra}},
  crossref =     "Miola:1990:DIS",
  pages =        "40--54",
  month =        "",
  year =         "1990",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "auch in: {AXIOM} Technical Report, ATR/1, NAG Ltd.,
                 Oxford, 1992.",
  abstract =     "The paper describes the constructive theory of
                 commutative algebra which underlies that part of
                 Scratchpad which deals with commutative algebra. The
                 authors begin by explaining the background that led the
                 Scratchpad group to construct such a general theory.
                 They contrast the general theory in Scratchpad with
                 Reduce-3's theory of domains, which is in many ways
```

```
                more limited, but is the closest approach to an
                implemented general theory to be found outside
                Scratchpad. This leads them to describe the general
                Scratchpad view of data types and categories, and the
                possibilities it offers. They then digress a little to
                ask what criteria should be adopted in choosing what
                types to define. Having discussed the philosophical
                issues, they then discuss commutative algebra proper,
                breaking this up into the sections ‘up to Ring’,
                ‘Integral Domain’, ‘Gcd Domain’ and ‘Euclidean
                Domain’.",
  acknowledgement = ack-nhfb,
  affiliation =  "Sch. of Math. Sci., Bath Univ., UK",
  classification = "C1110 (Algebra); C7310 (Mathematics)",
  keywords =     "Categories; Commutative algebra; Constructive theory;
                Data types; Euclidean Domain; Gcd Domain; Greatest
                common divisors; Integral Domain; Philosophical issues;
                Ring; Scratchpad",
  language =     "English",
  thesaurus =    "Algebra; Software packages; Symbol manipulation",
}
```

———-

— axiom.bib —

```
@inproceedings{Dave91,
  author = "Davenport, J. H. and Gianni, P. and Trager, B. M.",
  title = {{Scratchpad's View of Algebra II:
           A Categorical View of Factorization}},
  booktitle = "Proc. 1991 Int. Symp. on Symbolic and Algebraic Computation",
  series = "ISSAC '91",
  year = "1991",
  isbn = "0-89791-437-6",
  location = "Bonn, West Germany",
  pages = "32--38",
  numpages = "7",
  link = "\url{http://doi.acm.org/10.1145/120694.120699}",
  doi = "10.1145/120694.120699",
  acmid = "120699",
  publisher = "ACM",
  address = "New York, NY, USA",
  abstract = "
    This paper explains how Scratchpad solves the problem of presenting a
    categorical view of factorization in unique factorization domains,
    i.e.  a view which can be propagated by functors such as
    SparseUnivariatePolynomial or Fraction. This is not easy, as the
    constructive version of the classical concept of
    UniqueFactorizationDomain cannot be so propagated. The solution
    adopted is based largely on Seidenberg's conditions (F) and (P), but
    there are several additional points that have to be borne in mind to
    produce reasonably efficient algorithms in the required generality.
```

```
    The consequence of the algorithms and interfaces presented in this
    paper is that Scratchpad can factorize in any extension of the
    integers or finite fields by any combination of polynomial, fraction
    and algebraic extensions: a capability far more general than any other
    computer algebra system possesses. The solution is not perfect: for
    example we cannot use these general constructions to factorize
    polyinmoals in $\overline{Z[\sqrt{-5}]}[x]$ since the domain
    $Z[\sqrt{-5}]$ is not a unique factorization domain, even though
    $\overline{Z[\sqrt{-5}]}$ is, since it is a field. Of course, we can
    factor polynomials in $\overline{Z}[\sqrt{-5}][x]$",
  paper = "Dave91.pdf",
  keywords = "axiomref",
  beebe = "Davenport:1991:SVA"
}
```

———-

— Davenport:1991:SVA —

```
@InProceedings{Davenport:1991:SVA,
  author =       "J. H. Davenport and P. Gianni and B. M. Trager",
  title =        {{Scratchpad's view of algebra. II. A categorical
                 view of factorization}},
  crossref =     "Watt:1991:PIS",
  pages =        "32--38",
  month =        "",
  year =         "1991",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "auch in: {AXIOM} Technical Report, ATR/2, NAG Ltd.,
                 Oxford, 1992.",
  abstract =     "For pt.I see Proc. DISCO 1990 (p.40-54). The paper
                 explains how Scratchpad solves the problem of
                 presenting a categorical view of factorization in
                 unique factorization domains, i.e. a view which can be
                 propagated by functors such as
                 SparseUnivariatePolynomial or Fraction. This is not
                 easy, as the constructive version of the classical
                 concept of UniqueFactorizationdomain cannot be so
                 propagated. The solution adopted is based largely on
                 the Seidenberg conditions (F) and (P), but there are
                 several additional points that have to be borne in mind
                 to produce reasonably efficient algorithms in the
                 required generality. The consequence of the algorithms
                 and interfaces presented is that Scratchpad can
                 factorize in any extension of the integers or finite
                 fields by any combination of polynomial, fraction and
                 algebraic extensions: a capability far more general
                 than any other computer algebra system possesses.",
  acknowledgement = ack-nhfb,
  affiliation =  "Sch. of Math., Bath Univ., Claverton Down, UK",
  classification = "C4130 (Interpolation and function approximation);
```

```
                C7310 (Mathematics)",
  keywords =    "Algebraic extensions; Categorical view; Computer
                algebra system; Factorization; Finite fields; Fraction;
                Integers; Polynomial; Scratchpad; Seidenberg
                conditions",
  language =    "English",
  thesaurus =   "Mathematics computing; Polynomials; Symbol
                manipulation",
}
```

———-

— axiom.bib —

```
@techreport{Dave92c,
  author = "Davenport, James H. and Trager, Barry M.",
  title = {{Scratchpad's view of algebra I: Basic commutative algebra}},
  number = "TR3/92 (ATR/1) (NP2490)",
  institution = "Numerical Algorithm Group (NAG) Ltd.",
  year = "1992",
  abstract =
    "While computer algebra systems have dealt with polynomials and
    rational functions with integer coefficients for many years, dealing
    with more general constructs from commutative algebra is a more recent
    problem. In this paper we explain how one system solves this problem,
    what types and operators it is necessary to introduce and, in short,
    how one can construct a computational theory of commutative
    algebra. Of necessity, such a theory is rather different from the
    conventional, non-constructive, theory. It is also somewhat different
    from the theories of Seidenberg [1974] and his school, who are not
    particularly concerned with practical questions of efficiency.",
  paper = "Dave90c.pdf",
  keywords = "axiomref",
  beebe = "Davenport:1992:SVAa"
}
```

———-

— Davenport:1992:SVAa —

```
@TechReport{Davenport:1992:SVAa,
  author =       "J. H. Davenport and B. M. Trager",
  title =        {{Scratchpad's View of Algebra {I}: Basic Commutative
                 Algebra}},
  number =       "TR3/92 (ATR/1) (NP2490)",
  institution =  inst-NAG,
  address =      inst-NAG:adr,
  pages =        "??",
  month =        dec,
  year =         "1992",
  bibdate =      "Fri Dec 29 16:31:49 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
```

```
  acknowledgement = ack-nhfb,
}
```

———-

— axiom.bib —

```
@techreport{Dave92d,
  author = "Davenport, James H. and Gianni, Patrizia and Trager, Barry M.",
  title = {{Scratchpad's view of algebra II:
           A categorical view of factorization}},
  type = "Technical Report",
  number = "TR4/92 (ATR/2) (NP2491)",
  institution = "Numerical Algorithms Group, Inc.",
  address = "Downer's Grove, IL, USA and Oxford, UK",
  year = "1992",
  link = "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
  abstract = "
    This paper explains how Scratchpad solves the problem of presenting a
    categorical view of factorization in unique factorization domains,
    i.e.  a view which can be propagated by functors such as
    SparseUnivariatePolynomial or Fraction. This is not easy, as the
    constructive version of the classical concept of
    UniqueFactorizationDomain cannot be so propagated. The solution
    adopted is based largely on Seidenberg's conditions (F) and (P), but
    there are several additional points that have to be borne in mind to
    produce reasonably efficient algorithms in the required generality.

    The consequence of the algorithms and interfaces presented in this
    paper is that Scratchpad can factorize in any extension of the
    integers or finite fields by any combination of polynomial, fraction
    and algebraic extensions: a capability far more general than any other
    computer algebra system possesses. The solution is not perfect: for
    example we cannot use these general constructions to factorize
    polyinmoals in $\overline{Z[\sqrt{-5}]}[x]$ since the domain
    $Z[\sqrt{-5}]$ is not a unique factorization domain, even though
    $\overline{Z[\sqrt{-5}]}$ is, since it is a field. Of course, we can
    factor polynomials in $\overline{Z}[\sqrt{-5}][x]$",
  paper = "Dave91.pdf",
  keywords = "axiomref",
  beebe = "Davenport:1992:SVAb"
}
```

———-

— Davenport:1992:SVAb —

```
@TechReport{Davenport:1992:SVAb,
  author =       "J. H. Davenport and P. Gianni and B. M. Trager",
  title =        {{Scratchpad's View of Algebra II: A Categorical
                 View of Factorization}},
  number =       "TR4/92 (ATR/2) (NP2491)",
  institution =  inst-NAG,
```

```
  address =      inst-NAG:adr,
  pages =        "??",
  month =        dec,
  year =         "1992",
  bibdate =      "Fri Dec 29 16:31:49 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
  acknowledgement = ack-nhfb,
}
```

———-

— axiom.bib —

```
@techreport{Dave92a,
  author = "Davenport, James H.",
  title = {{The AXIOM system}},
  type = "technical report",
  number = "TR5/92 (ATR/3) (NP2492)",
  institution = "Numerical Algorithms Group, Inc.",
  year = "1992",
  abstract =
    "AXIOM is a computer algebra system superficially like many others,
    but fundamentally different in its internal construction, and
    therefore in the possibilities it offers to its users. In these
    lecture notes, we will
    \begin{itemize}
    \item outline the high-level design of the AXIOM kernel and the AXIOM type
    system,
    \item explain some of the algebraic facilities implemented in AXIOM,
    which may be more general than the reader is used to,
    \item show how the type system and the information system interact,
    \item give some references to the literature on particular aspects of
    AXIOM and,
    \item suggest the way forward.
    \end{itemize}",
  paper = "Dave92a.pdf",
  keywords = "axiomref, DONE",
  beebe = "Davenport:1992:AS"
}
```

———-

— Davenport:1992:AS —

```
@TechReport{Davenport:1992:AS,
  author =       "J. H. Davenport",
  title =        {{The AXIOM System}},
  type =         "AXIOM Technical Report",
  number =       "TR5/92 (ATR/3) (NP2492)",
  institution =  inst-NAG,
  address =      inst-NAG:adr,
  pages =        "??",
```

```
  month =        dec,
  year =         "1992",
  bibdate =      "Fri Dec 29 16:31:49 1995",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@techreport{Dave92b,
  author = "Davenport, James H.",
  title = {{How does one program in the AXIOM system?}},
  institution = "Numerical Algorithms Group, Inc.",
  year = "1992",
  type = "technical report",
  number = "TR6/92 (ATR/4)(NP2493)",
  link = "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
  abstract =
    "Axiom is a computer algebra system superficially like many others, but
    fundamentally different in its internal construction, and therefore in
    the possibilities it offers to its users and programmers. In these
    lecture notes, we will explain, by example, the methodology that the
    author uses for programming substantial bits of mathematics in Axiom.",
  paper = "Dave92b.pdf",
  keywords = "axiomref",
  beebe = "Davenport:1992:HDO"
}
```

———-

— **Davenport:1992:HDO** —

```
@TechReport{Davenport:1992:HDO,
  author =       "J. H. Davenport",
  title =        {{How Does One Program in the AXIOM System?}},
  type =         "AXIOM Technical Report",
  number =       "TR6/92 (ATR/4) (NP2493)",
  institution =  inst-NAG,
  address =      inst-NAG:adr,
  pages =        "??",
  month =        dec,
  year =         "1992",
  bibdate =      "Fri Dec 29 16:31:49 1995",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
  acknowledgement = ack-nhfb,
}
```

———-

— **ignore** —

```
\bibitem[Davenport 92c]{DT92} Davenport, J. H.; Trager, B. M.
  title = {{Scratchpad's view of algebra I: Basic commutative algebra}},
DISCO 90 Capri, Italy April 1990 ISBN 0-387-52531-9 pp40-54
Technical Report TR3/92 (ATR/1)(NP2490), Numerical
Algorithms Group, Inc., Downer's Grove, IL, USA and Oxford, UK,
December 1992.
  link = "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Dave92,
  author = "Davenport, James H.",
  title = {{Primality Testing Revisited}},
  link = "\url{http://staff.bath.ac.uk/masjhd/ISSACs/ISSAC1992.pdf}",
  booktitle = "Proc. ISSAC 1992",
  series = "ISSAC 92",
  publisher = "ACM",
  pages = "123-129",
  year = "1992",
  report = "Technical Report TR2/93 Numerical Algorithms Group, Inc",
  algebra = "\newline\refto{package PRIMES IntegerPrimesPackage}",
  abstract =
    "Rabin's algorithm is commonly used in computer algebra systems and
    elsewhere for primality testing. This paper presents an experience
    with this in the Axiom computer algebra system. As a result of this
    experience, we suggest certain strengthenings of the algorithm.",
  paper = "Dave92.pdf",
  keywords = "axiomref, printed, DONE",
  beebe = "Davenport:1993:PTR"
}
```

———-

— **Davenport:1993:PTR** —

```
@TechReport{Davenport:1993:PTR,
  author =       "J. H. Davenport",
  title =        {{Primality Testing Revisited}},
  number =       "TR2/93 (ATR/6) (NP2556)",
  institution =  inst-NAG,
  address =      inst-NAG:adr,
  pages =        "??",
  month =        aug,
  year =         "1993",
  bibdate =      "Fri Dec 29 16:31:49 1995",
```

```
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
  abstract =     "Rabin's algorithm is commonly used in computer algebra
                 systems and elsewhere for primality testing. This paper
                 presents an experience with this in the Axiom computer
                 algebra system. As a result of this experience, we
                 suggest certain strengthenings of the algorithm.",
  acknowledgement = ack-nhfb,
}
```

———-

— **axiom.bib** —

```
@techreport{Faur00,
  author = "Faure, Christele and Davenport, James H. and Naciri, Hanane",
  title = {{Multi-Valued Computer Algebra}},
  year = "2000",
  type = "technical report",
  institution = "INRIA CAFE",
  number = "4001",
  abstract =
    "One of the main strengths of computer algebra is being able to solve
    a family of problems with one computation. In order to express not
    only one problem but a family of problems, one introduces some symbols
    which are in fact the parameters common to all the problems of the
    family. The user must be able to understand in which way these
    parameters affect the result when he looks at the answer. Otherwise it
    may lead to completely wrong calculations, which when used for
    numerical applications bring nonsensical answers. This is the case in
    most current Computer Algebra Systems we know because the form of the
    answer is never explicitly conditioned by the values of the
    parameters. The user is not even informed that the given answer may be
    wrong in some cases then computer algebra systems can not be entirely
    trustworthy. We have introduced multi-valued expressions called
    conditional expressions, in which each potential value is associated
    with a condition on some parameters. This is used, in particular, to
    capture the situation in integration, where the form of the answer can
    depend on whether certain quantities are positive, negative or
    zero. We show that it is also necessary when solving modular linear
    equations or deducing congruence conditions from complex expressions.",
  paper = "Faur00.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Dave94,
  author = {Davenport, James and Faure, Christ\'ele},
  title = {{The Unknown in Computer Algebra}},
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/TheUnknownInComputerAlgebra.pdf}",
```

```
  year = "1994",
  abstract = "
    Computer algebra systems have to deal with the confusion between
    ``programming variables'' and ``mathematical symbols''. We claim that
    they should also deal with ``unknowns'', i.e. elements whose values
    are unknown, but whose type is known. For examples $x^p \ne x$ if $x$
    is a symbol, but $x^p = x$ if $x \in GF(p)$. We show how we have
    extended Axiom to deal with this concept.",
  paper = "Dave94.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Davenport 00]{Dav00} Davenport, James
``13th OpenMath Meeting''
James H. Davenport
  title = {{A New Algebra System}},
May 1984
  link = "\url{http://xml.coverpages.org/openmath13.html}",
  paper = "Dav00.pdf",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Dave11,
  author = "Davenport, James H.",
  title = {{CICM 2011: Conferences on Intelligent Computer Mathematics 2011}},
  journal = "Springer Lecture Notes in Artificial Intelligence 6824",
  pages = "1-67",
  link = "\url{http://people.bath.ac.uk/masjhd/Meetings/CICM2011.pdf}",
  comment = "http://www.springerlink.com/conten/978-3-642-22672-4",
  year = "2011",
  paper = "Dave11.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Dave12a,
  author = "Davenport, James H.",
  title = {{Computer Algebra or Computer Mathematics?}},
  year = "2012",
  link =
    "\url{http://people.bath.ac.uk/masjhd/Slides/CalculemusSchool2002.pdf}",
```

```
  abstract =
    "Scope: ``polynomial-type'' systems: Axiom, Macsyma/Maxima, Maple,
     Mathematica, and Reduce.",
  paper = "Dave12a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dave12b,
  author = "Davenport, James H.",
  title = {{Small Algorithms for Small Systems}},
  journal = "ACM Comm. in Computer Algebra",
  volume = "46",
  number = "1",
  year = "2012",
  paper = "Dave12b.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Davenport 12]{Dav12} Davenport, J.H.
  title = {{Computer Algebra}},
  link = "\url{http://staff.bath.ac.uk/masjhd/JHD-CA.pdf}",
  keywords = "axiomref"}
```

———-

— **axiom.bib** —

```
@misc{Dave15,
  author = "Davenport, James H.",
  title = {{SIAM AAG 15 and ICIAM 2015}},
  link = "\url{http://people.bath.ac.uk/masjhd/Meetings/AAG-ICIAM15.pdf}",
  paper = "Dave15.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Dave18,
  author = "Davenport, James H",
  title = {{Computer Algebra: Systems and Algorithms for Algebraic
            Computation (updated)}},
  publisher = "Davenport",
  year = "2018",
```

```
  link = "\url{http://staff.bath.ac.uk/masjhd/JHD-CA.pdf}",
  paper = "Dave18.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Deckxx,
  author = "Decker, Wolfram",
  title = {{Some Introductory Remarks on Computer Algebra}},
  link = "\url{https://www.math.uni-bielefeld.de/~rehmann/ECM/cdrom/3ecm/pdfs/pant3/decker.pdf}",
  abstract =
    "Computer algebra is a relatively young but rapidly growing field. In
    this introductory note to the mini-symposium on computer algebra
    organized as part of the third European Congress of Mathematics I will
    not even attempt to adress all major streams of research and the many
    applications of computer algebra. I will concentrate on a few aspects,
    mostly from a mathematical point of view, and I will discuss a few
    typical applications in mathematics. I will present a couple of
    examples which underline the fact that computer algebra systems
    provide easy access to powerful computing tools. And, I will quote
    from and refer to a couple of survey papers, textbooks and web-pages
    which I recommend for further reading.",
  paper = "Deckxx.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Dell99,
  author = "Delliere, Stephane",
  title = {Trangularisation de syst\‘emes constructibles Application \‘a
           l’\’evaluation dynamique},
  school = {L’Universit\’e de Limoges},
  year = "1999",
  link = "\url{http://www.unilim.fr/laco/theses/1999/T1999_03.pdf}",
  paper = "Dell99.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Dell00,
  author = "Delliere, Stephane and Wang, Dongming",
  title = {{simple systems and dynamic constructible closure}},
```

```
  institution = "Universite de Limoges",
  year = "2000",
  type = "technical report",
  number = "2000-16",
  link = "\url{http://www.unilim.fr/laco/rapports/2000/R2000\_16.pdf}",
  abstract =
    "Dynamic evaluation is a general method for computing with parameters
    [6, 9]. In 1994, T. Gomez-Diaz implemented the dynamic constructible
    closure in the scientific computation system Axiom [17]: by simulating
    dynamic evaluation, it offers the possibility to compute with
    parameters in a very large way [13]. The outputs of a calculs with
    T. Gomez-Diaz programs are represented by a finite collection of
    constructible triangular systems defined in [12, definition
    p.106]. Though there are numerous applications of these programs
    (notably polynomial system solving with parameters [11], automatic
    geometric theorem proving [14, 15], computation of Jordan forms with
    parameters [16]), nobody gives theorical interest to this kind of
    triangular systems. The main reason of this phenomenon is that they
    are definied in [12] within the dynamic evaluation context. On the
    opposite, most notions of triangular systems (J.F. Ritt-W.T. Wu
    characteristic sets [24, 28], M. Kalkbrener regular chains [18],
    D. Lazard triangular sets [20], M. Moreno Maza regular sets [22],
    D.M. Wang simple systems [26, 27]) are defined in terms of commutative
    algebra. This problem is at the origin of the work done in [7] where
    we give a relevant algebraic model of T. Gomez-Diaz systems within
    commutative algebra terminology. This allows us to relate them to many
    concepts of triangular systems [7]. Thus, we give interest to the
    connections with D. Lazard triangular sets in [8]. In a way, this
    paper is the continuation of this previous work. This time, we study
    relationships between T. Gomez-Diaz systems and D.M. Wang simple
    systems. The paper is structured as follows. We have collected in
    section 2 some needed notations. In section 3, we give all the
    terminology related to our algebraic model of T. Gomez-Diaz
    systems. Thus, we define the notion of weak constructible triangular
    systems and introduce the properties of normalization and
    squarefreeness. Section 4 is more detailed.  First of all, we study a
    weaker form of normalization called $L$-normalization. Then we give
    many properties of constructible triangular systems verifying this new
    notion. We obtain an algebraic and geometric framework which permits,
    in section 5, to explore the connections between T. Gomez-Diaz systems
    and D.M. Wang simple systems. In particular, this last section will
    demonstrate well the importance of our $L$-normalization
    property. Indeed, we show that simple systems and squarefree
    $L$-normalized constructible triangular systems are equivalent.",
  paper = "Dell00.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@techreport{Dell00a,
```

```
  author = "Delliere, Stephane",
  title = {{A first course to $D_7$ with examples}},
  institution = "Universite de Limoges",
  year = "2000",
  type = "technical report",
  number = "2000-17",
  link = "\url{http://www.unilim.fr/laco/rapports/2000/R2000_17.pdf}",
  paper = "Dell00a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Dell01,
  author = "Delliere, Stephane",
  title = {{On the links between triangular sets and dynamic constructable
           closure}},
  journal = "J. Pure Appl. Algebra",
  volume = "163",
  number = "1",
  pages = "49-68",
  year = "2001",
  abstract =
    "Two kinds of triangular systems are studied: normalized triangular
    polynomial systems (a weaker form of Lazards triangular sets
    [D. Lazard, Discrete Appl. Math. 33, No. 1-3, 147-160 (1991; Zbl
    0753.13013)] and constructible triangular systems (involved in the
    dynamic constructible closure programs of T. Gomez-Daz [Quelques
    applications de l'evaluation dynamique, Ph.D. Thesis, Universite de
    Limoges (1994)]. This paper shows that these notions are strongly
    related. In particular, combining the two points of view
    (constructible and polynomial) on the subject of square-free
    conditions, it allows us to effect dramatic improvements in the
    dynamic constructible closure programs.",
  paper = "Dell01.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Deut73,
  author = "Deutsch, L. Peter",
  title = {{An Interactive Program Verifer}},
  school = "University of California, Berkeley",
  year = "1973",
  paper = "Deut73.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Dewa92,
  author = "Dewar, Michael C.",
  title = {{Using Computer Algebra to Select Numerical Algorithms}},
  booktitle = "Proc. ISSAC 1992",
  series = "ISSAC 1992",
  year = "1992",
  location = "Berkeley, CA",
  pages = "1-8",
  algebra =
   "\newline\refto{domain D01AJFA d01ajfAnnaType}
    \newline\refto{domain D01AKFA d01akfAnnaType}
    \newline\refto{domain D01ALFA d01alfAnnaType}
    \newline\refto{domain D01AMFA d01amfAnnaType}
    \newline\refto{domain D01ANFA d01anfAnnaType}
    \newline\refto{domain D01APFA d01apfAnnaType}
    \newline\refto{domain D01AQFA d01aqfAnnaType}
    \newline\refto{domain D01ASFA d01asfAnnaType}
    \newline\refto{domain D01FCFA d01fcfAnnaType}
    \newline\refto{domain D01GBFA d01gbfAnnaType}
    \newline\refto{domain D01TRNS d01TransformFunctionType}
    \newline\refto{domain D02BBFA d02bbfAnnaType}
    \newline\refto{domain D02BHFA d02bhfAnnaType}
    \newline\refto{domain D02CJFA d02cjfAnnaType}
    \newline\refto{domain D02EJFA d02ejfAnnaType}
    \newline\refto{domain D03EEFA d03eefAnnaType}
    \newline\refto{domain D03FAFA d03fafAnnaType}
    \newline\refto{domain E04DGFA e04dgfAnnaType}
    \newline\refto{domain E04FDFA e04fdfAnnaType}
    \newline\refto{domain E04GCFA e04gcfAnnaType}
    \newline\refto{domain E04JAFA e04jafAnnaType}
    \newline\refto{domain E04MBFA e04mbfAnnaType}
    \newline\refto{domain E04NAFA e04nafAnnaType}
    \newline\refto{domain E04UCFA e04ucfAnnaType}
    \newline\refto{domain NIPROB NumericalIntegrationProblem}
    \newline\refto{domain ODEPROB NumericalODEProblem}
    \newline\refto{domain OPTPROB NumericalOptimizationProblem}
    \newline\refto{domain PDEPROB NumericalPDEProblem}",
  abstract =
    "Many real-life problems require a compbination of both symbolic and
    numerical methods for their solution. This has led to the development
    of intgrated, interactive symbolic / numeric packages which use a
    computer algebra system for the former and a standard subroutine
    library for the latter. These systems may also be viewed as simplified
    front-ends to the numerical library. To use these packages, however, a
    user must be able to select which of the many available routines is
    the most appropriate for his or her problem, which contrsts with the
    ``black-box'' style interfaces available in computer algebra
    systems. This paper describes how a computer algebra system can be
    used to make this decision, thus providing a much-simplified and more
    orthogonal interface.",
```

```
  paper = "Dewa92.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Dewar 94]{Dew94} Dewar, M. C.
  title = {{Manipulating Fortran Code in AXIOM and the AXIOM-NAG Link}},
Proceedings of the Workshop on Symbolic and Numeric Computing, ed by Apiola, H.
and Laine, M. and Valkeila, E. pp1-12 University of Helsinki, Finland (1994)
  keywords = "axiomref",
```

———-

— **axiom.bib** —

```
@misc{Dewa95,
  author = "Dewar, Mike C.",
  title = {{AXIOM and A\#: Current Status and Future Plans}},
  year = "1995",
  link = "\url{ftp://ftp.inf.ethz.ch/org/cathode/workshops/jan95/abstracts/dewar.ps}",
  paper = "Dewa95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Dewa,
 author = "Dewar, Mike",
 title = {{OpenMath: An Overview}},
 link = "\url{http://www.sigsam.org/bulletin/articles/132/paper1.pdf}",
 paper = "Dewa.pdf",
 keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Diaz06,
  author = "Diaz, Glauco Alfredo Lopez",
  title = {{Symbolic Methods for Factoring Linear Differential Operators}},
  school = "Johannes Kepler Universitat, Linz",
  year = "2006",
  month = "February",
  abstract =
    "A survey of symbolic methods for factoring linear differential
    operators is given. Starting from basic notions  ring of operators,
```

```
    differential Galois theory  methods for finding rational and
    exponential solutions that can provide first order right-hand factors
    are considered. Subsequently several known algorithms for
    factorization are presented. These include Singers eigenring
    factorization algorithm, factorization via Newton polygons, van
    Hoeijs methods for local factorization, and an adapted version of
    Pade approximation.

    In addition a procedure based on pure algebraic methods for factoring
    second order linear partial differential operators is
    developed. Splitting an operator of this kind reduces to solving a
    system of linear algebraic equations. Those solutions which satisfy a
    certain different ial condition, immediately produce linear factors of
    the operator. The method applies also to operators of third order,
    thereby resulting in a more complicated system of equations. In
    contrast to the second order case, differential equations must also be
    solved, which, in particular cases, are simplified with the aid of
    characteristic sets.

    Finally, complete decomposition into linear factors of ordinary
    differential operators of arbitrary order is discussed. A splitting
    formula is developed, provided that a linear basis of solutions is
    available.  This theoretical representation is valuable in
    understanding the nature of the classical Beke algorithm and its
    variants like the algorithm LODEF by Schwarz and the Beke-Bronstein
    algorithm.",
  paper = "Diaz06.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{DiBl95,
  author = "DiBlasio, Paolo and Temperini, Marco",
  title = {{Subtyping Inheritance and Its Application in Languages for
           Symbolic Computation Systems}},
  journal = "J. Symbolic Computation",
  volume = "19",
  pages = "39-63",
  year = "1995",
  abstract =
    "Application of object-oriented programming techniques to design and
    implementation of symbolic computation is investigated. We show the
    significance of certain correctness problems, occurring in programming
    environments based on specialization inheritance, due to use of method
    redefinition and polymorphism. We propose a solution to these
    problems, by defining a mechanism of subtyping inheritance and the
    prototype of an object-oriented programming language for a symbolic
    computation system. We devise the subtyping inheritance {\sl ESI
    (Enhanced String Inheritance)} by lifting to programming language
    constructs a given model of subtyping, which is established by a
```

```
    monotonic (covariant) subtyping rule. Type safeness of language
    instructions is proved.

    The adoption of {\sl ESI} allows to model method and class
    specialization in a natural way. The {\sl ESI} mechanism verifies the
    type correctness of language statements by means of type checking
    rules and preserves their correctness at run-time by a suitable method
    lookup algorithm.",
  paper = "DiBl95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{DiBl97,
  author = "DiBlasio, Paolo and Temperini, Marco",
  title = {{On subtyping in languages for symbolic computation systems}},
  booktitle = "Advances in the design of symbolic computation systems",
  series = "Monographs in Symbolic Computation",
  year = "1997",
  publisher = "Springer",
  pages = "164-178",
  abstract =
    "We want to define a strongly typed OOP language suitable as the
    software development tool of a symbolic computation system, which
    provides class structure to manage ADTs and supports multiple
    inheritance to model specialization hierarchies. In this paper, we
    provide the theoretical background for such a task.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Dicr88,
  author = "Dicrescenzo, C. and Duval, D.",
  title = {{Algebraic extensions and algebraic closure in Scratchpad II}},
  booktitle = "Proc. ISSAC 1988",
  series = "ISSAC 1998",
  year = "1998",
  pages = "440-446",
  isbn = "3-540-51084-2",
  abstract =
    "Many problems in computer algebra, as well as in high-school
    exercises, are such that their statement only involves integers but
    their solution involves complex numbers. For example, the complex
    numbers $\sqrt{2}$ and $-\sqrt{2}$ appear in the solutions of
    elementary problems in various domains.
    \begin{itemize}
```

```
    \item in {\bf integration}:
    \[\int{\frac{dx}{x^2-2}} = \frac{Log(x-\sqrt{2})}{2\sqrt{2}}
    +\frac{Log(x-(-\sqrt{2}))}{2(-\sqrt{2})}\]
    \item in {\bf linear algebra}: the eigenvalues of the matrix
    \[\left(\begin{array}{cc}
    1 & 1\\
    1 & -1
    \end{array}\right) = \sqrt{2} {\rm\ and\ }-\sqrt{2}\]
    \item in {\bf geometry}: the line $y=x$ intersects the circle
    $y^2+x^2=1$ at the points
    \[(\sqrt{2},\sqrt{2}) {\rm\ and\ }(-\sqrt{2},-\sqrt{2})\]
    \end{itemize}
    Of course, more ``complicated'' complex numbers appear in more
    complicated examples.

    But two facts have to be emphasized:
    \begin{itemize}
    \item in general, if a problem is stated over the integers (or over
    the field $\mathbb{Q}$ of rational numbers), the complex numbers that
    appear are {\sl algebraic} complex numbers, which means that they are
    roots of some polynomial with rational coefficients, like $\sqrt{2}$
    and $-\sqrt{2}$ are roots of $T^2-2$.
    \item Similar problems appear with base fields different from
    $mathbb{Q}$. For example finite fields, or fields of rational
    functions over $\mathbb{Q}$ or over a finite field. The general
    situation is that a given problem is stated over some ``small field''
    $K$, and its solution is expressed in an {\sl algebraic closure}
    $\overline{K}$ of $K$, which means that this solution involves numbers
    which are roots of polynomials with coefficients in $K$.
    \end{itemize}

    The aim of this paper is to describe an implementation of an algebraic
    closure domain constructor in the language Scratchpad II, simply
    called Scratchpad below. In the first part we analyze the problem, and
    in the second part we describe a solution based on the D5 system.",
  paper = "Dicr88.pdf",
  keywords = "axiomref",
  beebe = "Dicrescenzo:1989:AEA"
}
```

———-

— Dicrescenzo:1989:AEA —

```
@InProceedings{Dicrescenzo:1989:AEA,
  author =       "C. Dicrescenzo and D. Duval",
  title =        {{Algebraic extensions and algebraic closure in
                 Scratchpad II}}",
  crossref =     "Gianni:1989:SAC",
  pages =        "440--446",
  year =         "1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
```

```
                http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =    "Many problems in computer algebra, as well as in
                high-school exercises, are such that their statement
                only involves integers but their solution involves
                complex numbers. For example, the complex numbers $
                \sqrt 2 $ and $ - \sqrt 2 $ appear in the solutions of
                elementary problems in various domains. The authors
                describe an implementation of an algebraic closure
                domain constructor in the language Scratchpad II. In
                the first part they analyze the problem, and in the
                second part they describe a solution based on the D5
                system.",
  acknowledgement = ack-nhfb,
  affiliation =  "TIM3, INPG, Grenoble, France",
  classification = "C7310 (Mathematics)",
  keywords =    "Algebraic closure domain constructor; D5 system;
                Language Scratchpad II",
  language =    "English",
  thesaurus =   "Mathematics computing; Symbol manipulation",
}
```

———-

— **axiom.bib** —

```
@misc{Dicr95,
  author = "Dicrescenzo, C. and Jung, Francoise",
  title = {{COMPASS package}},
  year = "1995",
  link = "\url{ftp://ftp.inf.ethz.ch/org/cathode/workshops/jan95/abstracts/bronstein.ps}",
  paper = "Dicr95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Dicr05,
  author = "Dicrescenzo, C. and Duval, D.",
  title = {{Algebraic extensions and algebraic closure in Scratchpad II}},
  booktitle = "Symbolic and Algebraic Computation",
  series = "Lecture Notes in Computer Science 358",
  year = "2005",
  publisher = "Springer",
  pages = "440-446",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Ding94,
  author = "Dingle, Adam and Fateman, Richard",
  title = {{Branch Cuts in Computer Algebra}},
  year = "1994",
  booktitle = "Proc. ISSAC 1994",
  series = "ISSAC 94",
  link = "\url{http://www.cs.berkeley.edu/~fateman/papers/ding.ps}",
  abstract =
    "Many standard functions, such as the logarithms and square root
    functions, cannot be defined continuously on the complex
    plane. Mistaken assumptions about the properties of these functions
    lead computer algebra systems into various conundrums. We discuss how
    they can manipulate such functions in a useful fashion.",
  paper = "Ding94.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[DLMF]{DLMF}.
  title = {{Digital Library of Mathematical Functions}},
  link = "\url{http://dlmf.nist.gov/software/#T1}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@InProceedings{Dool98,
  author = "Dooley, Samuel S.",
  title = {{Coordinating mathematical content and presentation markup in
           interactive mathematical documents}},
  booktitle = "Proc. ISSAC 1998",
  series = "ISSAC 98",
  year = "1998",
  publisher = "ACM Press",
  location = "Rostock, Germany",
  pages = "13-15",
  abstract =
    "This paper presents a method for representing mathematical content
    and presentation markup in interactive mathematical documents that
    treats each view of the information on a separate and equal
    footing. By providing extensible, overridable, default mappings from
    content to presentation in a way that supports efficient mappings back
    from the presentation to the underlying content, a user interface for
    an interactive textbook has been implemented where the user interacts
    with high-quality presentation markup that supports user operations
    defined in terms of the mathematical content. In addition, the user
    interface can be insulated from content-specific information, while
```

```
    still being enabled to transfer that information to other programs for
    computation. This method has been employed to embed interactive
    mathematical content into the IBM techexplorer Interactive Textbook
    for Linear Algebra. The issues involved in the implementation of the
    interactive textbook also shed some light on the problems faced by the
    MathML working group in representing both presentation and content for
    mathematics for interactive web documents.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Dool02,
  author = "Dooley, Samuel S.",
  title = {{Editing mathematical content and presentation markup in
           interactive mathematical documents}},
  booktitle = "Proc. ISSAC 2002",
  series = "ISSAC 02",
  year = "2002",
  publisher = "ACM Press",
  pages = "55-62",
  abstract =
    "The IBM MathML Expression Editor is a two-dimensional
    mathematical editor for expressions encoded using MathML content
    and presentation elements. It allows a user to interact with the
    visual presentation of an expression, while simultaneously
    creating the underlying structure of the expression. This paper
    describes the internal expression framework used by the editor to
    represent the content and presentation structures, the layout
    mechanisms used to transform content into presentation, the
    structural navigation conventions used to select subexpressions,
    the editing templates used to support visual input of MathML
    content expressions, and the customization framework that allows
    for a fully extensible set of content operators, including
    complete support for MathML 2.0 content elements as well as
    user-defined function symbols and operators."
}
```

———-

— **ignore** —

```
\bibitem[Dooley 99]{Doo99} Dooley, Sam editor.
ISSAC 99: July 29-31, 1999, Simon Fraser University,
Vancouver, BC, Canada: proceedings of the 1999 International Symposium on
Symbolic and Algebraic Computation. ACM Press, New York, NY 10036, USA, 1999.
ISBN 1-58113-073-2 LCCN QA76.95.I57 1999
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Reis06,
  author = "Dos Reis, Gabriel and Stroustrup, Bjarne",
  title = {{Specifying C++ Concepts}},
  journal = "POPL",
  publisher = "ACM",
  year = "2006",
  link = "\url{http://www.stroustrup.com/popl06.pdf}",
  abstract =
    "C++ templates are key to the design of current successful mainstream
    libraries and systems. They are the basis of programming techniques in
    diverse areas ranging from conventional general-purpose programming to
    software for safety-critical embedded systems. Current work on improving
    templates focuses on the notion of {\sl concepts} (a type system for
    templates), which promises significantly improved error diagnostics and
    increased expressive power such as concept-based overloading and function
    template partial specialization. This paper presents C++ templates with
    an emphasis on problems related to separate compilation. We consider the
    problem of how to express concepts in a precise way that is simple enough
    to be usable by ordinary programmers. In doing so, we expose a few
    weaknesses of the current specification of the C++ standard library and
    suggest a far more precise and complete specification. We also present
    a systematic way of translating our proposed concept definitions, based
    on use-patterns rather than function signatures, into constraint sets
    that can serve as convenient basis for concept checking in a compiler.",
  paper = "Reis06.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Reis12,
  author = "Dos Reis, Gabriel",
  title = {{A System for Axiomatic Programming}},
  journal = "Proc. Conf. on Intelligent Computer Mathematics",
  publisher = "Springer",
  year = "2012",
  link = "\url{http://www.axiomatics.org/~gdr/liz/cicm-2012.pdf}",
  abstract = "
    We present the design and implementation of a system for axiomatic
    programming, and its application to mathematical software
    construction. Key novelties include a direct support for user-defined
    axioms establishing local equality between types, and overload
    resolution based on equational theories and user-defined local
    axioms. We illustrate uses of axioms, and their organization into
    concepts, in structured generic programming as practiced in
    computational mathematical systems.",
  paper = "Reis12.pdf",
  keywords = "axiomref, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Doye97,
  author = "Doye, Nicolas James",
  title = {{Order Sorted Computer Algebra and Coercions}},
  school = "University of Bath",
  year = "1997",
  abstract =
    "Computer algebra systems are large collections of routines for solving
    mathematical problems algorithmically, efficiently and above all,
    symbolically. The more advanced and rigorous computer algebra systems
    (for example, Axiom) use the concept of strong types based on
    order-sorted algebra and category theory to ensure that operations are
    only applied to expressions when they ``make sense''.

    In cases where Axiom uses notions which are not covered by current
    mathematics we shall present new mathematics which will allow us to
    prove that all such cases are reducible to cases covered by the
    current theory. On the other hand, we shall also point out all the
    cases where Axiom deviates undesirably from the mathematical ideal.
    Furthermore we shall propose solutions to these deviations.

    Strongly typed systems (especially of mathematics) become unusable
    unless the system can change the type in a way a user expects. We wish
    any change expected by a user to be automated, ``natural'', and
    unique. ``Coercions'' are normally viewed as ``natural type changing
    maps''. This thesis shall rigorously define the word ``coercion'' in
    the context of computer algebra systems.

    We shall list some assumptions so that we may prove new results so
    that all coercions are unique. This concept is called ``coherence''.

    We shall give an algorithm for automatically creating all coercions in
    type system which adheres to a set of assumptions. We shall prove that
    this is an algorithm and that it always returns a coercion when one
    exists. Finally, we present a demonstration implementation of this
    automated coerion algorithm in Axiom.",
  paper = "Doye97.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Doye99,
  author = "Doye, Nicolas James",
  title = {{Automated coercion for Axiom}},
  booktitle = "Proc. ISSAC 1999",
```

```
  pages = "229-235",
  year = "1999",
  isbn = "1-58113-073-2",
  link = "\url{http://www.acm.org/citation.cfm?id=309944}",
  paper = "Doye99.pdf",
  keywords = "axiomref, printed",
  beebe = "Doye:1999:ACA"
}
```

———-

— Doye:1999:ACA —

```
@InProceedings{Doye:1999:ACA,
  author =       "Nicolas J. Doye",
  title =        {{Automated coercion for Axiom}}",
  crossref =     "Dooley:1999:IJS",
  pages =        "229--235",
  year =         "1999",
  bibdate =      "Sat Mar 11 16:39:42 MST 2000",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://www.acm.org/pubs/contents/proceedings/issac/309831/}",
  acknowledgement = ack-nhfb,
}
```

———-

— axiom.bib —

```
@InProceedings{Domi01,
  author = {Dom\'inguez, C\'esar; Rubio, Julio},
  title = {{Modeling Inheritance as Coercion in a Symbolic Computation
           System}},
  booktitle = "Proc. ISSAC 2001",
  series = "ISSAC 2001",
  year = "2001",
  abstract = "
    In this paper the analysis of the data structures used in a symbolic
    computation system, called Kenzo, is undertaken. We deal with the
    specification of the inheritance relationship since Kenzo is an
    object-oriented system, written in CLOS, the Common Lisp Object
    System. We focus on a particular case, namely the relationship between
    simplicial sets and chain complexes, showing how the order-sorted
    algebraic specifications formalisms can be adapted, through the
    ``inheritance as coercion'' metaphor, in order to model this Kenzo
    fragment.",
  paper = "Domi01.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Drag10,
  author = "Dragan, Laurentiu and Watt, Stephen",
  title = {{Type Specialization in Aldor}},
  booktitle = "Computer algebra in scientific computing",
  series = "CASC 2010",
  year = "2010",
  location = "Tsakhadzor, Armenia",
  pages = "73-84",
  link = "\url{http://www.csd.uwo.ca/~watt/pub/reprints/2010-casc-specdom.pdf}",
  abstract =
    "Computer algebra in scientific computation squarely faces the dilemma
    of natural mathematical expression versus efficiency. While
    higher-order programming constructs and parametric polymorphism
    provide a natural and expressive language for mathematical
    abstractions, they can come at a considerable cost. We investigate how
    deeply nested type constructions may be optimized to achieve
    performance similar to that of hand-tuned code written in lower-level
    languages.",
  paper = "Drag10.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Dunstan 97]{Dun97} Dunstan, Martin and Martin, Ursula and
     Linton, Steve
  title = {{Embedded Verification Techniques for Computer Algebra Systems}},
Grant citation GR/L48256 Nov 1, 1997-Feb 28, 2001
  link = "\url{http://www.cs.st-andrews.ac.uk/research/output/detail?output=ML97.php}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Duns99b,
  author = "Dunstan, Martin",
  title = {{An Introduction to Aldor and its Type System}},
  year = "1999",
  link = "\url{http://www.aldor.org/docs/reports/cfc99/aldor-cfc99.pdf}",
  comment = "slides",
  paper = "Duns99b.pdf"
}
```

———-

— axiom.bib —

```
@misc{Dupe95,
  author = "Dupee, Brian J. and Davenport, James H.",
  title = {{Using Computer Algebra to Choose and Apply Numerical Routines}},
  year = "1995",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.5645}",
  algebra =
   "\newline\refto{domain D01AJFA d01ajfAnnaType}
    \newline\refto{domain D01AKFA d01akfAnnaType}
    \newline\refto{domain D01ALFA d01alfAnnaType}
    \newline\refto{domain D01AMFA d01amfAnnaType}
    \newline\refto{domain D01ANFA d01anfAnnaType}
    \newline\refto{domain D01APFA d01apfAnnaType}
    \newline\refto{domain D01AQFA d01aqfAnnaType}
    \newline\refto{domain D01ASFA d01asfAnnaType}
    \newline\refto{domain D01FCFA d01fcfAnnaType}
    \newline\refto{domain D01GBFA d01gbfAnnaType}
    \newline\refto{domain D01TRNS d01TransformFunctionType}
    \newline\refto{domain D02BBFA d02bbfAnnaType}
    \newline\refto{domain D02BHFA d02bhfAnnaType}
    \newline\refto{domain D02CJFA d02cjfAnnaType}
    \newline\refto{domain D02EJFA d02ejfAnnaType}
    \newline\refto{domain D03EEFA d03eefAnnaType}
    \newline\refto{domain D03FAFA d03fafAnnaType}
    \newline\refto{domain E04DGFA e04dgfAnnaType}
    \newline\refto{domain E04FDFA e04fdfAnnaType}
    \newline\refto{domain E04GCFA e04gcfAnnaType}
    \newline\refto{domain E04JAFA e04jafAnnaType}
    \newline\refto{domain E04MBFA e04mbfAnnaType}
    \newline\refto{domain E04NAFA e04nafAnnaType}
    \newline\refto{domain E04UCFA e04ucfAnnaType}
    \newline\refto{domain NIPROB NumericalIntegrationProblem}
    \newline\refto{domain ODEPROB NumericalODEProblem}
    \newline\refto{domain OPTPROB NumericalOptimizationProblem}
    \newline\refto{domain PDEPROB NumericalPDEProblem}",
  abstract =
    "In applied mathematics, electronic and chemical engineering, the
    modelling process can produce a number of mathematical problems which
    require numerical solutions for which symbolic methods are either not
    possible or not obvious. With the plethora of numerical library
    routines for the solution of these problems often the numerical
    analyst has to answer the question {\sl Which routine to choose?} and
    {\sl How do I use it?}. Some analysis needs to be carried out before
    the appropriate routine can be identifed, i.e. {\sl How stiff is this
    ODE?} and {\sl Is this function continuous?}. It may well be the case
    that more than one routine is applicable to the problem. So the
    question may become {\sl Which is likely to be the best?}. Such a
    choice may be critical for both accuracy and efficiency.

    An expert system is thus required to make this choice based on the
    results of its own analysis of the problem, call the routine and act
    on the outcome. This may be to put the answer in a relevant form or
    react to an apparent failure of the chosen routine and thus choose and
    call an alternative. It should also have sufficient explanation
```

```
    mechanisms to inform on the choice of routine and the reasons for that
    choice. Much of this work can be achieved using computer algebra and
    symbolic algebra packages.

    This paper describes an expert system currently in prototype in terms
    of both its object-based structure and its computational agents. Some
    of these agents are described in detail, paying particular attention
    to the practical aspects of their algorithms and the use of computer
    algebra.

    The {\bf axiom2} Symbolic Algebra System is used as a user interface
    as well as the link to the NAG Foundation Library for the numerical
    routines and the inference mechanisms for the expert system.",
  paper = "Dupe95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Dupe99,
  author = "Dupee, Brian J. and Davenport, James H.",
  title = {{An Automatic Symbolic-Numeric Taylor Series ODE Solver}},
  booktitle = "Computer Algebra in Scientific Computing, CASC'99",
  isbn = "978-3-540-66047-7",
  pages = "37-50",
  year = "1999",
  link = "\url{http://people.eecs.berkeley.edu/~fateman/papers/casc99-34.pdf}",
  comment = "Contains FORTRAN Code of Taylor Series",
  algebra = "\newline\refto{package EXPRODE ExpressionSpaceODESolver}",
  abstract =
    "One of the basic techniques in every mathematician's toolkit is the
    Taylor series representation of functions. It is of such fundamental
    importance and it is so well understood that its use is often a first
    choice in numerical analysis. This faith has not, unfortunately, been
    transferred to the design of computer algorithms.

    Approximation by use of Taylor series methods is inherently partly a
    symbolic process and partly numeric> This aspect has often, with
    reason, been regared as a major hindrance in algorithm design. Whilst
    attempts have been made in the past to build a consistent set of
    programs for the symbolic and numeric paradigms, these have been
    necessarily multi-stage processes.

    Using current technology it has at last become possible to integrate
    these two concepts and build an automatic adaptive symbolic-numeric
    algorithm within a uniform framework which can hide the internal
    workings behind a modern interface.",
  paper = "Dupe99.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@inproceedings{Dupe05,
  author = "Dupee, Brian J. and Davenport, James H.",
  title = {{An Intelligent Interface to Numerical Routines}},
  booktitle = "Design and Implementation of Symbolic Computation Systems",
  series = "Lecture Notes in Computer Science 1128",
  pages = "252-262",
  publisher = "Springer",
  year = "2005",
  abstract =
    "Links from Computer Algebra Systems to Numerical Libraries have been
    increasingly made available. However, the remain, like the numerical
    routines which comprise those libraries, difficult to use by a novice
    and there is little help in choosing the appropriate routine for any
    given problem, should there be a choice.

    Compuer Algebra Systems use generic names for each problem area. For
    examples, 'integrate' (or 'int') is used for integration of a
    function, whatever method the code may use. Numeric interfaces still
    use different names for each method together with a variety of extra
    parameters, some of which may be optional. Ideally, we should extend
    the generic name structure to cover numerical routines. This would
    then, necessarily, require algorithms for making an assessment of the
    efficacy of different methods where such a choice exists.

    This paper considers the link to the NAG Fortran Library from version
    2.0 of Axiom and shows how we can build on this to extend and simplify
    the interface using an expert system for choosing and using the
    numerical routines.",
  paper = "Dupe05.pdf",
  ksyaodxz = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Duva92,
  author = "Duval, Anne and Loday-Richaud, Michele",
  title = {{Kovacic's Algorithm and Its Application to Some Families
           of Special Functions}},
  journal = "Applicable Algebra in Engineering, Communication, and Computing",
  series = "AAECC 3",
  pages = "211-246",
  year = "1992",
  publisher = "Springer-Verlag",
  abstract =
    "We apply the Kovacic algorithm to some families of special functions,
    mainly the hypergeometric one and that of Heun, in order to discuss
```

```
    the existence of closed-form solutions. We begin by giving a slightly
    modified version of the kovacic algorithm and a sketch proof.",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@inproceedings{Duva92a,
  author = "Duval, Dominique and Jung, F.",
  title = {{Examples of problem solving using computer algebra}},
  booktitle = "Programming environments for high-level scientific problem
               solving",
  series = "IFIP Transactions",
  editor = "Gaffney, Patrick W. and Houstis, Elias N.",
  publisher = "North-Holland",
  pages = "133-143",
  year = "1992",
  keywords = "axiomref",
  beebe = "Duval:1992:EPS"
}
```

———-

— Duval:1992:EPS —

```
@Article{Duval:1992:EPS,
  author =       "D. Duval and F. Jung",
  title =        {{Examples of problem solving using computer algebra}},
  journal =      j-IFIP-TRANS-A,
  volume =       "A-2",
  pages =        "133--141, 143",
  month =        "",
  year =         "1992",
  CODEN =        "ITATEC",
  ISSN =         "0926-5473",
  bibdate =      "Tue Sep 17 06:41:20 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Computer algebra, in contrast with numerical analysis,
                 aims at returning exact solutions to given problems.
                 One consequence is that the shape of the solutions may,
                 at first, look somewhat surprising. The authors present
                 two examples of problem solving using computer algebra,
                 with emphasis on the shape of the solutions. The first
                 example is the resolution of linear differential
                 equations with polynomial coefficients, and the second
                 one is the resolution of polynomial equations in one
                 variable. In the first example the solution may look
                 useless since it makes use of divergent series, and in
                 the second example the solution may look rather
                 awkward. But in both examples it is shown that these
                 solutions are in the right shape for a lot of
```

```
                applications, including numerical ones. It is also
                shown that some features of the computer algebra system
                Scratchpad, especially strong typing and genericity,
                are useful for the implementation of a method for a
                second problem, i.e. for the implementation of the
                'dynamic' algebraic closure of a field.",
  acknowledgement = ack-nhfb,
  affiliation =  "Lab. de Theorie des Nombres et Algorithmique, Limoges
                Univ., France",
  classification = "C7310 (Mathematics)",
  fjournal =     "IFIP Transactions. A. Computer Science and
                Technology",
  keywords =     "Algebraic closure; Computer algebra; Divergent series;
                Exact solutions; Genericity; Linear differential
                equations; Polynomial coefficients; Polynomial
                equations; Problem solving; Scratchpad; Strong typing",
  language =     "English",
  thesaurus =    "Linear differential equations; Polynomials; Symbol
                manipulation",
}
```

———-

— **axiom.bib** —

```
@misc{Duva94e,
  author = "Duval, Dominique",
  title = {{Symbolic or algebraic computation?}},
  booktitle = "Publication du LACO",
  year = "1995",
  location = "Madrid Spain",
  comment = "NAG conference",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Duva94d,
  author = "Duval, Dominique and Senechaud, Pascale",
  title = {{Sketches and parametrization}},
  journal = "Theor. Comput. Sci.",
  volume = "123",
  number = "1",
  pages = "117-130",
  year = "1994",
  abstract =
    "The paper deals with problems about conception and design of
    high-level computer algebra systems. Here we use a categorical
    approach given by the notion of sketches. Sketches allow to describe
    computation mechanisms in a syntactic way, well adapted to
```

```
    implementation.

    A computer algebra system must allow the manipulation of algebraic
    structures, in particular, the construction of new structures from
    known ones. In the paper we give a definition, at the sketch level, of
    parametrization of a structure by another one.",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Duva95,
  author = "Duval, Dominique",
  title = {{Dynamic evaluation and algebraic closure in Axiom}},
  comment = "Evaluation dynamique et cl\^oture alg\'ebrique en Axiom",
  journal = "Journal of Pure and Applied Algebra",
  volume = "99",
  year = "1995",
  pages = "267--295",
  abstract =
    "Dynamic evaluation allows to compute with algebraic numbers without
    factorizing polynomials.  It also allows to manipulate parameters in a
    flexible and user-friendly way.  The aim of this paper is the
    following: Explain what is dynamic evaluation, with its basic notions
    of dynamic set and splitting.  Present its application to computations
    involving algebraic numbers, which amounts to defining the dynamic
    algebraic closure of a field.  Describe the Axiom program which
    implements this, and give a user guide for it (only this last point
    assumes some knowledge of Axiom) Dynamic evaluation is described here
    without any reference to sketch theory, however our presentation, less
    rigourous, may be considered as more accessible.",
  paper = "Duva95.pdf",
  keywords = "axiomref",
  beebe = "Duval:1995:DEA"
}
```

———-

— Duval:1995:DEA —

```
@Article{Duval:1995:DEA,
  author =       "Dominique Duval",
  title =        "{{\'E}}valuation dynamique et cl{\^o}ture
                 alg{\'e}brique en {Axiom}. (French) {Dynamic evaluation
                 and algebraic closure in Axiom}",
  journal =      "J. Pure Appl. Algebra",
  volume =       "99",
  number =       "3",
  pages =        "267--295",
  year =         "1995",
  DOI =          "http://dx.doi.org/10.1016/0022-4049(94)00053-L",
```

```
  ISSN =         "0022-4049",
  MRclass =      "11Y40 (Algebraic number theory computations) 68W30
                 (Symbolic computation and algebraic computation) 68Q65
                 (Abstract data types; algebraic specification) 18C10
                 (Algebraic theories, etc.)",
  bibdate =      "Tue Mar 30 18:47:12 2010",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  ZMnumber =     "0851.1107",
  abstract =     "Dynamic evaluation is a method of computing that
                 permits the computation to be refined into different
                 cases that are considered separately. A precise
                 description and mathematical foundation was given in
                 terms of sketch theory by {\it D. Duval} and {\it J.-C.
                 Reynaud} [Math. Structures Comput. Sci. 4, 239-271
                 (1994; Zbl 0822.68063)]. In the present paper, the
                 mechanism of dynamic evaluation is explained without
                 reference to sketch theory in order to make it
                 accessible to a wider audience. Next, it is shown how
                 dynamic evaluation can be employed to compute with
                 algebraic numbers without having to do explicit
                 factorization of polynomials. The essential step here
                 is to define the dynamic algebraic closure of a field.
                 Finally, a program for the Axiom system implementing
                 dynamic algebraic closure is presented.",
  acknowledgement = ack-nhfb,
  keywords =     "algebraic numbers; Axiom; dynamic algebraic closure of
                 a field; dynamic evaluation; sketches",
  language =     "French",
  reviewer =     "A. Bijlsma (Eindhoven)",
}
```

———-

1.40.5 E

— axiom.bib —

```
@mastersthesis{ElAl01,
  author = "El-Alfy, Hazem Mohamed",
  title = {{Computer Algebra and its Applications}},
  school = "Alexandria University, Department of Engineering, Mathematics,
            and Physics",
  year = "2001",
  link = "\url{http://www.umiacs.umd.edu/~helalfy/pub/mscthesis01.pdf}",
  file = "ElAl01.pdf",
  abstract =
    "In the recent decades, it has been more and more realized that
    computers are of enormous importance for numerical
    computations. However, these powerful general-purpose machines can
    also be used for transforming, combining and computing symbolic
    algebraic expressions. In other words, computers can not only deal
    with numbers, but also with abstract symbols representing mathematical
```

```
    formulas. This fact has been realized much later and is only now
    gaining acceptance among mathematicians and engineers. [Franz Winkler,
    1996].

    Computer Algebra is that field of computer science and mathematics,
    where computation is performed on symbols representing mathematical
    objects rather than their numeric values.

    This thesis attempts to present a definition of computer algebra by
    means of a survey of its main topics, together with its major
    application areas. The survey includes necessary algebraic basics and
    fundamental algorithms, essential in most computer algebra problems,
    together with some problems that rely heavily on these algorithms. The
    set of applications, presented from a range of fields of engineering
    and science, although very short, indicates the applied nature of
    computer algebra systems.

    A recent research area, central in most computer algebra software
    packages and in geometric modeling, is the implicitization
    problem. Curves and surfaces are naturally reperesented either
    parametrically or implicitly. Both forms are important and have their
    uses, but many design systems start from parametric
    representations. Implicitization is the process of converting curevs
    and surfaces from parametric form into implicit form.

    We have surveyed the problem of implicitization and investigated its
    currently available methods. Algorithms for such methods have been
    devised, implemented and tested for practical examples. In addition, a
    new method has been devised for curves for which a direct method is
    not available. The new method has been called {\sl near implicitization}
    since it relies on an approximation of the input problem. Several
    variants of the method try to compromise between accuracy and
    complexity of the designed algorithms.

    The problem of implicitization is an active topic where research is
    still taking place.  Examples of further research points are included
    in the conclusion",
  paper = "ElAl01.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Ency16,
  author = "Unknown",
  title = {{Encyclopedia of Mathematics}},
  link = "\url{https://www.encyclopediaofmath.org/index.php/Computer\_algebra\_package}",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Erocal 10]{ES10} Er\"ocal, Burcin; Stein, William
  title = {{The Sage Project}},
  link = "\url{http://wstein.org/papers/icms/icms_2010.pdf}",
  abstract = "
    Sage is a free, open source, self-contained distribution of
    mathematical software, including a large library that provides a
    unified interface to the components of this distribution. This library
    also builds on the components of Sage to implement novel algorithms
    covering a broad range of mathematical functionality from algebraic
    combinatorics to number theory and arithmetic geometry.",
  paper = "ES10.pdf",
  keywords = "axiomref"
```

———-

1.40.6 F

— **axiom.bib** —

```
@article{Fakl97,
  author = "Fakler, Winfried",
  title = {{On second order homogeneous linear differential equations with
           Liouvillian solutions}},
  journal = "Theor. Comput. Sci.",
  volume = "187",
  number = "1-2",
  pages = "27-48",
  year = "1997",
  abstract =
    "We determine all minimal polynomials for second order homogeneous
    linear differential equations with algebraic solutions decomposed into
    invariants and we show how easily one can recover the known conditions
    on differential Galois groups [J. Kovacic, J. Symb. Comput. 2, 3-43
    (1986; Zbl 0603.68035), M. F. Singer and F. Ulmer,
    J. Symb. Comput. 16, 9-36, 37-73 (1993; Zbl 0802.12004, Zbl
    0802.12005), F.Ulmer and J. A. Weil, J. Symb. Comput. 22, 179-200
    (1996; Zbl 0871.12008)] using invariant theory. Applying these
    conditions and the differential invariants of a differential equation
    we deduce an alternative method to the algorithms given in (loc. cit.)
    for computing Liouvillian solutions. For irreducible second order
    equations our method determines solutions by formulas in all but three
    cases.",
  paper = "Fakl97.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Farm03,
  author = "Farmer, William M. and von Mohrenschildt, Martin",
  title = {{An overview of A Formal Framework For Managing Mathematics}},
  journal = "Ann. Math. Artif. Intell.",
  volume = "38",
  number = "1-3",
  pages = "165-191",
  year = "2003",
  link = "\url{https://www.emis.de/proceedings/MKM2001/farmer.ps}",
  abstract =
    "Mathematics is a process of creating, exploring, and connecting
    mathematical models. This paper presents an overview of a formal
    framework for managing the mathematics process as well as the
    mathematical knowledge produced by the process. The central idea of
    the framework is the notion of a biform theory which is simultaneously
    an axiomatic theory and an algorithmic theory. Representing a
    collection of mathematical models, a biform theory provides a formal
    context for both deduction and computation. The framework includes
    facilities for deriving theorems via a mixture of deduction and
    computation, constructing sound deduction and computation rules, and
    developing networks of biform theories linked by interpretations. The
    framework is not tied to a specific underlying logic; indeed, it is
    intended to be used with several background logics
    simultaneously. Many of the ideas and mechanisms used in the framework
    are inspired by the IMPS Interactive Mathematical Proof System and the
    Axiom computer algebra system.",
  paper = "Farm03.pdf",
  keywords = "axiomref, CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Fate79,
  author = "Fateman, Richard J.",
  title = {{MACSYMA's General Simplifier: Philosophy and Operation}},
  booktitle = "Proc. Macsyma Users' Conference 1979",
  year = "1979",
  link = "\url{http://people.eecs.berkeley.edu/~fateman/papers/simplifier.txt}",
  abstract =
    "Ideally the transformations performed by MACSYMA's simplification
    program on algebraic expressions correspond to those simplifications
    desired by each user and each program.  Since it is impossible for a
    program to intuit all users' requirements simultaneously, explicit
    control of the simplifier is necessary to override default
    transformations.  A model of the simplification process is helpful in
    controlling this large and complex program.

    Having examined several algebraic simplification programs, it appears
    that to date no program has been written which combines a conceptually
    simple and useful view of simplification with a program nearly as
```

```
    powerful as MACSYMA's.  {note, 1979. not clear this would be different
    in 2001.  RJF} Rule-directed transformation schemes struggle to
    approach the power of the varied control structures in more usual
    program schemes [Fenichel, 68].  {note, Mathematica pushes rules
    further. RJF}

    It is our belief that a thorough grasp of the decision and data
    structures of the MACSYMA simplifier program itself is the most direct
    way of understanding its potential for algebraic expression
    transformation.  This is an unfortunate admission to have to make, but
    it appears to reflect the state of the art in dealing with
    formalizations of complex programs.  Simplification is a perplexing
    task.  Because of this, we feel it behooves the ``guardians of the
    simplifier'' to try to meet the concerned MACSYMA users part-way by
    documenting the program as it has evolved.  We hope this paper
    continues to grow to reflect a reasonably accurate, complete, and
    current description.

    Of course Lisp program details are available to the curious, but even
    for those without a working knowledge of the Lisp language (in which
    the simplifier is written) we expect this paper to be of some help in
    answering questions which arise perennially as to why MACSYMA deals
    with some particular class of expressions in some unanticipated
    fashion, or is inefficient in performing some set of transformations.
    Most often difficulties such as these are accounted for by implicit
    design decisions which are not evident from mere descriptions of what
    is done in the anticipated and usual cases.  We also hope that
    improvements or revisions of the simplifier will benefit from the more
    centralized treatment of issues given here.  We also provide
    additional commentary which reflects our current outlook on how
    simplification programs should be written, and what capabilities they
    should have.",
  paper = "Fate79.txt",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Fate90,
  author = "Fateman, Richard J.",
  title = {{Advances and trends in the design and construction of algebraic
           manipulation systems}},
  booktitle = "Proc. ISSAC 1990",
  publisher = "ACM",
  pages = "60-67",
  isbn = "0-89791-401-5",
  year = "1990",
  link = "\url{http://people.eecs.berkeley.edu/~fateman/papers/advances.pdf}",
  abstract =
    "We compare and contrast several techniques for the implementation of
    components of an algebraic manipulation system. On one hand is the
```

```
    mathematical-algebraic approach which chaaracterizes (for example)
    IBM's Axiom. On the other hand is the more {\sl ad hoc} approach which
    characterizes many other popular systems (for example, Macsyma,
    Reduce, Maple, and Mathematica). While the algebraic approach has
    generally positive results, careful examination suggests that there
    are significant remaining problems, expecially in the representation
    and manipulation of analytical, as opposed to algebraic,
    mathematics. We describe some of these problems and some general
    approaches for solutions.",
  paper = "Fate90.pdf",
  keywords = "axiomref",
  beebe = "Fateman:1990:ATD"
}
```

———-

— Fateman:1990:ATD —

```
@InProceedings{Fateman:1990:ATD,
  author =       "R. J. Fateman",
  title =        {{Advances and trends in the design and construction of
                 algebraic manipulation systems}},
  crossref =     "Watanabe:1990:IPI",
  pages =        "60--67",
  year =         "1990",
  DOI =          "http://dx.doi.org/10.1145.96895",
  bibdate =      "Thu Jul 26 09:04:25 2001",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Compares and contrasts several techniques for the
                 implementation of components of an algebraic
                 manipulation system. On one hand is the mathematical
                 algebraic approach which characterizes (for example)
                 IBM's Scratchpad II. On the other hand is the more ad
                 hoc approach which characterizes many other popular
                 systems (for example, Macsyma, Reduce, Maple, and
                 Mathematica). While the algebraic approach has
                 generally positive results, careful examination
                 suggests that there are significant remaining problems,
                 especially in the representation and manipulation of
                 analytical, as opposed to algebraic mathematics. The
                 author describes some of these problems, and some
                 general approaches for solutions.",
  acknowledgement = ack-nhfb,
  affiliation =  "California Univ., Berkeley, CA, USA",
  classification = "C4240 (Programming and algorithm theory); C7310
                 (Mathematics)",
  keywords =     "Algebraic manipulation systems; Algebraic mathematics;
                 Macsyma; Maple; Mathematica; Mathematical algebraic;
                 Reduce; Scratchpad II",
  language =     "English",
  thesaurus =    "Algebra; Symbol manipulation",
}
```

———-

— **axiom.bib** —

```
@misc{Fate94,
  author = "Fateman, Richard J.",
  title = {{On the Design and Construction of Algebraic Manipulation Systems}},
  link = "\url{http://www.cs.berkeley.edu/~fateman/papers/asmerev94.ps}",
  abstract =
    "We compare and contrast several techniques for the implementation of
    components of an algebraic manipulation system. On one hand is the
    mathematical-algebraic approach which characterizes (for example)
    IBM's Axiom. On the other hand is the more {\sl ad-hoc} approach which
    characterizes many other popular systems (for example, Macsyma,
    Reduce, Maple, and Mathematica). While the algebraic approach has
    generally positive results, careful examination suggests that there
    are significant remaining problems, especially in the representation
    and manipulation of analytical, as opposed to algebraic,
    mathematics. We describe some of these problems and some general
    approaches for solutions.",
  paper = "Fate94.pdf",
  keywords = "axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Fate96,
  author = "Fateman, Richard J.",
  title = {{A Review of Symbolic Solvers}},
  booktitle = "Proc 1996 ISSAC",
  series = "ISSAC 96",
  year = "1996",
  pages = "86-94",
  link = "\url{http://http.cs.berkeley.edu/~fateman/papers/eval.ps}",
  abstract =
    "``Evaluation'' of expressions and programs in a computer algebra
    system is central to every system, but inevitably fails to provide
    complete satisfaction. Here we explain the conflicting requirements,
    describe some solutions from current systems, and propose alternatives
    that might be preferable sometimes. We give examples primarily from
    Axiom, Macsyma, Maple, Mathematica, with passing metion of a few other
    systems.",
  paper = "Fate96.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Fate97,
  author = "Fateman, Richard J.",
  title = {{Network Servers for Symbolic Mathematics}},
  booktitle = "Proc. ISSAC 1997",
  pages = "249-256",
  year = "1997",
  isbn = "0-89791-875-4",
  link = "\url{http://http.cs.berkeley.edu/~fateman/papers/cas-serve.ps}",
  abstract =
    "We describe advantages to using network socket facilities for
    communication and distributed computing from the perspective of
    symbolic mathematics systems. For some applications, an easily
    constructed Lisp server model provides a flexible portal between
    computer algebra and other programs, and one need not use new
    languages or new systems or write new stand-alone web-specific cgi-bin
    applications. Such socket programs can use, if necessary, HTML as a
    common transport encoding, but more efficient means are possible. We
    show that, rather than distributing all information to each computer
    algebra user's system, it makes sense to consider networking for
    accessing tables of information maintained at one or a few
    sites. Finally, we mention some consequences of the economic value of
    computation.",
  paper = "Fate97.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Fate99,
  author = "Fateman, Richard J.",
  title = {{Symbolic Mathematics System Evaluators}},
  booktitle = "Proc. ISSAC 1996",
  pages = "86-94",
  year = "1999",
  link = "\url{http://people.eecs.berkeley.edu/~fateman/papers/evalnew.pdf}",
  abstract =
    "``Evaluation'' of expressions and programs in a computer algebra
    system is central to every system, but inevitably fails to provide
    complete satisfaction. Here we explain the conflicting requirements,
    describe some solutions from current systems, and propose alternatives
    that might be preferable sometimes. We give examples primarily from
    Axiom, Macsyma, Maple, Mathematica, with passing metion of a few other
    systems.",
  paper = "Fate99.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fate99a,
  author = "Fateman, Richard J. and Caspi, Eylon",
  title = {{Parsing TeX into Mathematics}},
  year = "1999",
  link = "\url{http://lib.org.by/\_djvu/\_Papers/Computer\_algebra/CAS%20systems/}",
  abstract =
    "Communication, storage, transmission, and searching of complex
    material has become increasingly important. Mathematical computing in
    a distributed environment is also becoming more plausible as libraries
    and computing facilities are connected with each other and with user
    facilites. TeX is a well-known mathematical typesetting language, and
    from the display perspective it might seem that it could be used for
    communication between computer systems as well as an intermediate form
    for the results of OCR (optical character recognition) of mathematical
    expressions. There are flaws in this reasoning, since exchanging
    mathematical informaiton requires a system to parse and semantically
    ``understand'' the TeX, even if it is ``ambiguous'' notationally. A
    program we developed can handle 43\% of 10,740 TeX formulas in a
    well-known table of integrals. We expect that a higher success rte can
    be achieved easily.",
  paper = "Fate99a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Fate00,
  author = "Fateman, Richard J.",
  title = {{Problem solving environments and symbolic computing}},
  booktitle = "Enabling technologies for computational science",
  publisher = "Kluwer Academic Publishers",
  year = "2000",
  pages = "91-102",
  link = "\url{http://people.eecs.berkeley.edu/~fateman/papers/pse-kluwer.pdf}",
  abstract =
    "What role should be played by symbolic mathematical computation
    facilities in scientific and engineering ``problem solving
    environments''? Drawing upon standard facilities such as numerical and
    graphical libraries, symbolic computation should be useful for: The
    creation and manipulation of mathematical models; The production of
    custom optimized numerical software; The solution of delicate classes
    of mathematical problems that require handling beyond that available
    in traditional machine-supported floating-point computation. Symbolic
    representation and manipulation can potentially play a central
    organizing role in PSEs since their more general object representation
    allows a program to deal with a wider range of computational
    issues. In particular numerical, graphical, and other processing can
    be viewed as special cases of symbolic manipulation with interactive
    symbolic computing providing both an organizing backbone and the
    communication ``glue'' among otherwise dissimilar components",
  paper = "Fate00.pdf",
```

```
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fate91,
  author = "Fateman, Richard J.",
  title = {{A Review of Mathematica}},
  year = "1991",
  link = "\url{https://people.eecs.berkeley.edu/~fateman/papers/mma.review.pdf}",
  abstract =
    "The Mathematica computer system is reviewed from the perspective of
    its contributions to symbolic and algebraic computation, as well as
    its stated goals. Design and implementation issues are discussed.",
  paper = "Fate91.pdf",
  keywords = "axiomref, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Fate01,
  author = "Fateman, Richard J.",
  title = {{A Review of Macsyma}},
  journal = "IEEE Trans. Knowl. Eng.",
  volume = "1",
  number = "1",
  year = "2001",
  link = "\url{http://people.eecs.berkeley.edu/~fateman/papers/mac82b.pdf}",
  abstract =
    "We review the successes and failures of the Macsyma algebraic
    manipulation system from the point of view of one of the original
    contributors. We provide a retrospective examination of some of the
    controversial ideas that worked, and some that did not. We consider
    input/output, language semantics, data types, pattern matching,
    knowledge-adjunction, mathematical semantics, the user community,
    and software engineering. We also comment on the porting of this
    system to a variety of computing systems, and possible future
    directions for algebraic manipulation system-building.",
  paper = "Fate01.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fate02,
```

```
  author = "Fateman, Richard J.",
  title = {{Comparing the speed of programs for sparse polynomial
           multiplication}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/fastmult.pdf}",
  paper = "Fate02.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fate05,
  author = "Fateman, Richard J.",
  title = {{An incremental approach to building a mathematical
           expert out of software}},
  conference = "Axiom Computer Algebra Conference",
  location = "City College of New York, CAISS project",
  year = "2005",
  month = "April",
  day = "19",
  link = "\url{http://www.cs.berkeley.edu/~fateman/papers/axiom.pdf}",
  paper = "Fat05.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fate05a,
  author = "Fateman, Richard J.",
  title = {{Haddock's eyes and computer algebra systems: Some essays}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/haddock.pdf}",
  year = "2005",
  abstract =
    "(From {\sl Through the Looking Glass} by Lewis Caroll)

    The White Knight proposes to comfort Alice by singing her a song:

    ``Is it very long?'' Alice asked, for she had heard a good deal of
    poetry that day.

    ``It's long,'' said the Knight, ``but it's very, very beautiful.
    Everybody that hears me sing it--either it brings tears into the
    eyes, or else --

    ``Or else what?'' said Alice, for the Knight had made a sudden pause.

    ``Or else it doesn't, you know. The name of the song is called
    `Haddock's Eyes'.''
```

```
    ‘‘Oh, that’ the name of the song, is it?’’ Alice said, trying to feel
    interested.

    ‘‘No, you don’t understand,’’ the Knight said, looking a little vexed.

    ‘‘That’s what the name is called. The name really is ’The Aged Aged Man’’’

    ‘‘Then I ought to have said ’That’s what the song is called?’’ Alice
    corrected herself.

    ‘‘No, you oughtn’t: that’s quite another thing! The song is called
    ‘Ways and Means’: but that’s only what it’s called, you know!’’

    ‘‘Well, what is the song, then?’’ said Alice, who was by this time
    completely bewildered.

    ‘‘I was coming to that,’’ the Knight said. ‘‘The song really is
    ‘A-sitting on a Gate’: and the tune’s my own invention.’’",
  paper = "Fate05a.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Fateman 06]{Fat06} Fateman, R. J.
  title = {{Building Algebra Systems by Overloading Lisp}},
  link = "\url{http://www.cs.berkeley.edu/~fateman/generic/overload-small.pdf}",
  abstract = "
    Some of the earliest computer algebra systems (CAS) looked like
    overloaded languages of the same era. FORMAC, PL/I FORMAC, Formula
    Algol, and others each took advantage of a pre-existing language base
    and expanded the notion of a numeric value to include mathematical
    expressions. Much more recently, perhaps encouraged by the growth in
    popularity of C++, we have seen a renewal of the use of overloading to
    implement a CAS.

    This paper makes three points. 1. It is easy to do overloading in
    Common Lisp, and show how to do it in detail. 2. Overloading per se
    provides an easy solution to some simple programming problems. We show
    how it can be used for a ‘‘demonstration’’ CAS. Other simple and
    plausible overloadings interact nicely with this basic system. 3. Not
    all goes so smoothly: we can view overloading as a case study and
    perhaps an object lesson since it fails to solve a number of
    fairly-well articulated and difficult design issues in CAS for which
    other approaches are preferable.",
  paper = "Fat06.pdf",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Fate06a,
  author = "Fateman, Richard J.",
  title = {{Comments on Extending Macsyma with New Data Types}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/addformat.pdf}",
  abstract =
    "Any design for a computer algebra system (CAS) naturally includes a
    set of data layouts for symbolic or mathematical algebraic expressions
    intended for use by built-in or user-written programs. The CAS cannot
    build in all plausible data designs, but supports those of most
    interest to the programmers. In such a situation it is almost
    inevitable that some new data encoding idea will come to mind and with
    it an interest in adding additional data forms. The motivation may be
    for compact representation, or efficient (fast) manipulation, or for
    other reasons such as interchange with other programs. Most CAS
    therefore include at least one way to extend the base set of
    operations. We comment on the kinds of extensions possible, using
    Macsyma as an example CAS. The particular interest in Macsyma and its
    open-source soruceforge variant ``Maxima'' is that a substantial group
    of very-losely coupled independent researchers are approaching this
    problem and may benefit from some guidance. Some the observations
    apply to other CAS, even though they are not open-source.",
  paper = "Fate06a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Fate06b,
  author = "Fateman, Richard J.",
  title = {{Building Algebra Systems by Overloading Lisp: Automatic
           Differentiation}},
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/overload-AD.pdf}",
  year = "2006",
  abstract =
    "In an earlier paper we began a discussion of the use of overloaded
    languages for support of computer algebra systems. Here we extend that
    notion to provide a more detailed approach to Automatic
    Differentiation or Algorithm Differention (AD).

    This paper makes three points. 1. It is extremely easy to do express
    AD by overloading in Common Lisp. 2. While the resulting program is
    not the most efficient approach in terms of run-time, it is quite
    small and very general. It also interacts nicely with some other kinds
    of generic arithmetic. 3. A more efficient AD compile-time program
    generation approach is described as well.",
  paper = "Fate06b.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Faug94,
 author = "Faug\'ere, J.C. and Gianni, P. and Lazard, D. and Mora, T.",
 title = {{Efficient Computation of Zero-dimensional Grobner Bases by
          Change of Ordering}},
 journal = "J. Symbolic Computation",
 issue_date = "February 1994",
 volume = "11",
 number = "2",
 month = "February",
 year = "1984",
 link = "\url{http://www-polsys.lip6.fr/~jcf/Papers/FGLM.pdf}",
 publisher = "Academic Press Limited",
 algebra = "\newline\refto{package LEXTRIPK LexTriangularPackage}",
 abstract = "
    We present an efficient algorithm for the transformation of a
    Grobner basis of a zero-dimensional ideal with respect to any given
    ordering into a Grobner basis with respect to any other
    ordering. This algorithm is polynomial in the degree of the idea. In
    particular, the lexicographical Grobner basis can be obtained by
    applying this algorithm after a total degree Grobner basis
    computation: it is usually much faster to compute the basis this way
    than with a direct application of Buchberger's algorithm.",
 paper = "Faug94.pdf",
 keywords = "axiomref"
}
```

————-

— **axiom.bib** —

```
@misc{Faurxx,
  author = {Davenport, James and Faure, Christ\'ele},
  title = {{Parameters in Computer Algebra}},
  abstract =
    "One of the main strengths of computer algebra is being able to solve
    a family of problems with one computation. In order to express not
    only one problem but a family of problems, one introduces some symbols
    which are in fact the parameters common to all the problems of the family.

    The user must be able to understand in which way these parameters
    affect the result when he looks at the answer. This is not the case in
    most current Computer Algebra Systems we know because the form of the
    answer is never explicitly conditioned by the values of the
    parameters. We have introduced multi-valued expressions called
    {\sl conditional expressions}, in which each potential value is associated
    with a condition on some parameters. This is used, in particular, to
    capture the situation in integration, where the form of the answer can
    depend on whether certain quantities are positive, negative, or zero.",
  keywords = "axiomref, provisos"
}
```

———-

— **ignore** —

```
\bibitem[Faure 00b]{FDN00b} Faure, Christ\'ele; Davenport, James;
Naciri, Hanane
  title = {{Multi-values Computer Algebra}},
ISSN 0249-6399 Institut National De Recherche en Informatique et en
Automatique Sept. 2000 No. 4001
  link = "\url{http://hal.inria.fr/inria-00072643/PDF/RR-4401.pdf}",
  abstract = "
    One of the main strengths of computer algebra is being able to solve a
    family of problems with one computation. In order to express not only
    one problem but a family of problems, one introduces some symbols
    which are in fact the parameters common to all the problems of the
    family.

    The user must be able to understand in which way these parameters
    affect the result when he looks at the answer. Otherwise it may lead
    to completely wrong calculations, which when used for numerical
    applications bring nonsensical answers. This is the case in most
    current Computer Algebra Systems we know because the form of the
    answer is never explicitly conditioned by the values of the
    parameters. The user is not even informed that the given answer may be
    wrong in some cases then computer algebra systems can not be entirely
    trustworthy. We have introduced multi-valued expressions called {\sl
    conditional} expressions, in which each potential value is associated
    with a condition on some parameters. This is used, in particular, to
    capture the situation in integration, where the form of the answer can
    depend on whether certain quantities are positive, negative or
    zero. We show that it is also necessary when solving modular linear
    equations or deducing congruence conditions from complex expressions.",
  paper = "FDN00b.pdf",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Fitch 84]{Fit84} Fitch, J. P. (ed)
EUROSAM '84: International Symposium on Symbolic and
Algebraic Computation, Cambridge, England, July 9-11, 1984, volume 174 of
Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany /
Heildelberg, Germany / London, UK / etc., 1984 ISBN 0-387-13350-X
LCCN QA155.7.E4 I57 1984
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@InProceedings{Fitc93,
```

```
  author = "Fitch (ed), John P.",
  title = {{Design and Implementation of Symbolic Computation Systems}},
  year = "1992",
  booktitle = "Int. Symp. DISCO '92 Proceedings",
  series = "DISCO 92",
  publisher = "Springer-Verlag, Berlin",
  isbn = "0-387-57272-4",
  paper = "Fitc93.tex",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Flei94,
  author = "Fleischer, J. and Grabmeier, J. and Hehl, F.W. and
            Kuchlin, W. (eds)",
  title = {{Proc. Conf. Computer Algebra in Science and Engineering}},
  booktitle = "Computer Algebra in Science and Engineering",
  year = "1994",
  location = "Bielefeld, Germany",
  publisher = "World Scientific, River Edge, NJ",
  abstract =
    "Systems and tools of computer algebra (Like AXIOM, Derive, FORM,
    Mathematica, Maple, Mupad, REDUCE, Macsyma) let us manipulate
    extremely complex algebraic formulae symbolically on a
    computer. Contrary to numerics these computations are exact and there
    is no loss of accuracy. After decades of research and development,
    these tools are now becoming as indispensable in Science and
    Engineering as traditional number crunching already is.

    The ZiF'94 workshop is amongst the first devoted specifically to
    applications of computer algebra (CA) in Science and Engineering. The
    book documents the state of the art in this area and serves as an
    important reference for future work."
}
```

———-

— **ignore** —

```
\bibitem[Fogus 11]{Fog11} Fogus, Michael
  title = {{UnConj}},
  link = "\url{http://clojure.com/blog/2011/11/22/unconj.html}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@techreport{Fort85,
  author = "Fortenbacher, A. and Jenks, Richard and Lucks, Michael and
            Sutor, Robert and Trager, Barry and Watt, Stephen",
  title = {{An Overview of the Scratchpad II Language and System}},
  institution = "IBM",
  year = "1985",
  type = "Research Report",
  publisher = "IBM Research Computer Algebra Group",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Fort90,
  author = "Fortenbacher, Albrecht",
  title = {{Efficient type inference and coercion in computer algebra}},
  booktitle = "Design and Implementation of Symbolic Computation Systems",
  series = "Lecture Notes in Computer Science 429",
  pages = "56-60",
  isbn = "0-387-52531-9",
  year = "1990",
  publisher = "Springer",
  abstract =
    "Computer algebra systesm of the new generation, like SCRATCHPAD, are
    characterized by a very rich type concept, which models the
    relationship between mathematical domains of computation. To use these
    systems interactively, however, the user should be freed of type
    information. A type inference mechanism determines the appropriate
    function to call. All known models which allow to define a semantics
    for type inference cannot express the rich ‘‘mathematical’’ type
    structure, so presently type inference is done heuristically. The
    following paper defines a semantics for a subproblem therof, namely
    coercion, which is based on rewrite rules. From this definition, an
    efficient coercion algorithm for SCRATCHPAD is constructed using graph
    techniques.",
  paper = "Fort90.pdf",
  keywords = "axiomref, printed",
  beebe = "Fortenbacher:1990:ETI"
}
```

———-

— **Fortenbacher:1990:ETI** —

```
@InProceedings{Fortenbacher:1990:ETI,
  author =       "A. Fortenbacher",
  title =        {{Efficient type inference and coercion in computer
                 algebra}},
  crossref =     "Miola:1990:DIS",
  pages =        "56--60",
  month =        "",
```

```
  year =         "1990",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Computer algebra systems of the new generation, like
                 Scratchpad, are characterized by a very rich type
                 concept, which models the relationship between
                 mathematical domains of computation. To use these
                 systems interactively, however, the user should be
                 freed of type information. A type inference mechanism
                 determines the appropriate function to call. All known
                 models which define a semantics for type inference
                 cannot express the rich 'mathematical' type structure,
                 so presently type inference is done heuristically. The
                 following paper defines a semantics for a subproblem,
                 namely coercion, which is based on rewrite rules. From
                 this definition, an efficient coercion algorithm for
                 Scratchpad is constructed using graph techniques.",
  acknowledgement = ack-nhfb,
  affiliation =  "Sci. Center Heidelberg, IBM Deutschland GmbH,
                 Germany",
  classification = "C1110 (Algebra); C4210 (Formal logic); C6120 (File
                 organisation); C7310 (Mathematics)",
  keywords =     "Coercion algorithm; Computer algebra; Graph
                 techniques; Rewrite rules; Scratchpad; Type inference
                 mechanism",
  language =     "English",
  thesaurus =    "Algebra; Data structures; Inference mechanisms;
                 Mathematics computing; Rewriting systems; Symbol
                 manipulation",
}
```

———-

— axiom.bib —

```
@article{Fort05,
  author = "Fortuna, E. and Gianni, P. and Luminati, D. and Parenti, P.",
  title = {{The adjacency graph of a real algebraic surface}},
  journal = "Appl. Algebra Eng. Commun. Comput.",
  volume = "16",
  number = "5",
  pages = "271-292",
  year = "2005",
  link = "\url{http://eprints.biblio.unitn.it/788/1/UTM671.pdf}",
  abstract =
    "The paper deals with the question of recognizing the mutual positions
    of the connected components of a non-singular real projective surface
    $S$ in the real projective 3-space. We present an algorithm that
    answers this question through the computation of the adjacency graph
    of the surface; it also allows to decide whether each connected
    component is contractible or not. The algorithm, combined with a
    previous one returning as an output the topology of the surface,
    computes a set of data invariant up to ambient-homeomorphism which,
```

```
    though not sufficient to determine the pair $(\mathbb{R}\mathbb{P}^3,S)$,
    give information about the nature of the surface as an embedded object.",
  paper = "Fort05.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Fouc90,
  author = "Fouche, Francois",
  title = {{Une implantation de l'algorithme de Kovacic en Scratchpad}},
  type = "technical report",
  number = "ULP-IRMA-447-P-254",
  year = "1990",
  institution = {Institut de Recherche Math{\'{e}}matique Avanc{\'{e}}e''},
  location = "Strasbourg, France",
  keywords = "axiomref",
  beebe = "Fouche:1990:ILK"
}
```

———-

— **Fouche:1990:ILK** —

```
@TechReport{Fouche:1990:ILK,
  author =       "Francois Fouche",
  title =        {{Une Implantation de l'algorithme de Kovacic en
                 Scratchpad}}",
  institution =  "Institut de Recherche Math{\'{e}}matique
                 Avanc{\'{e}}e",
  address =      "Strasbourg, France",
  pages =        "31",
  year =         "1990",
  bibdate =      "Sat Dec 30 08:25:26 MST 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
}
```

———-

— **ignore** —

```
\bibitem[FSF 14]{FSF14} FSF
  title = {{Free Software Directory}},
  link = "\url{http://directory.fsf.org/wiki/Axiom}",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Frisco ]{Fris} Frisco
  title = {{Objectives and Results}},
  link = "\url{http://www.nag.co.uk/projects/frisco/frisco/node3.htm}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Frit94,
  author = "Fritzson, D. and Fritzson, P. and Viklund, L. and Herber, J.",
  title = {{Object-oriented mathematical modelling - applied to machine
           elements}},
  journal = "Comput. Struct.",
  volume = "51",
  number = "3",
  pages = "241-253",
  year = "1994",
  abstract =
    "Machine element analysis has a goal of describing function and other
    aspects of machine elements in a theoretical form. This paper shows
    how ideas from object-oriented modelling can be applied to machine
    elment analysis. The models thus obtained are both easier to
    understand, better structured, and allow a higher degree of re-use
    than conventional models. An object-oriented model description is
    natural and suitable for machine element analysis. As a realistic
    example an equational model of rolling bearings is presented. The
    structure of the model is general, and applies to many types of
    rolling bearings. The model and one solution require approximately
    200+200 equations. The model is extensible, e.g. simple submodels of
    detailed properties can be made more complex without altering the
    overall structure. The example model has been implemented in a
    language of our own design. ObjectMath (Object-oriented Mathematical
    language for scientific computing). Using ObjectMath, it is possible
    to model classes of equation objects, to support multiple and single
    inheritance of equations, to support composition of equations, and to
    solve systems of equations. Algebraic transformations can conveniently
    be done since ObjectMath models are translated into the Mathematica
    computer algebra language. When necessary, equations can be
    transformed int C++ code for efficient numerical solution. The re-use
    of equations through inheritance reduced the size of the model by a
    factor of two, compared to a direct representation of the model in the
    Mathematica computer algebra language.",
  paper = "Frit94.pdf",
  keywords = "axiomref"
}
```

———-

1.40.7 G

— axiom.bib —

```
@article{Gall90,
  author = "Galligo, A. and Grimm, J. and Pottier, L.",
  title = {{The Design of SISYPHE: A System for doing Symbolic and
           Algebraic Computations}},
  journal = "LNCS",
  volume = "429",
  pages = "30-39",
  year = "1990",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@techreport{Gall92,
  author = "Gallopoulos, Stratis and Houstis, Elias and Rice, John",
  title = {{Future Research Directions in Problem Solving Environments for
           Computational Science}},
  institution = "Purdue University",
  year = "1992",
  type = "technical report",
  number = "CSD-TR-92-032",
  link = "\url{http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1953\&context=cstech}",
  abstract =
    "During the early 19605 some were visualizing that computers could
    provide a powerful problem solving environment (PSE) which would
    interact with scientists on their own terms.  By the mid 1960s there
    were many attempts underway to create these PSEs, but the early
    1970s almost all of these attempts had been abandoned, because the
    technological infrastructure could not yet support PSEs in
    computational science.  The dream of the 1960s can be the reality of
    the 1990s; high performance computers combined with better
    understanding of computing and computational science have put PSEs
    well within our reach.",
  paper = "Gall92.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@book{Ganz00,
  author = "Ganzha, Victor G. and Vorozhtsov, Evgenii V. and Wester, Michael",
  title = {{An Assessment of the Efficiency of Computer Algebra Systems in
           the Solution of Scientific Computing Problems}},
  booktitle = "Computer Algebra in Scientific Computing",
```

```
  year = "2000",
  isbn = "978-3-540-41040-9",
  publisher = "Springer",
  pages = "145-166",
  abstract =
    "Computer algebra systems (CASs) have become an important tool for the
    solution of scientific computing problems. With the increasing number
    of general purpose CASs, there is now a need for an assessment of the
    efficiency of these systems. We discuss some peculiarities associated
    with the analysis of CPU time efficiency in CASs, and then present
    results from three specific systems (Maple Vr5, Mathematics 4.0 and
    MuPAD 1.4) on a sample of intermediate size problems. These results
    show that Maple Vr5 is generally the speediest on our
    examples. Finally, we formulate some requirements for developing a
    comprehensive suite for analyzing the efficiency of CASs.",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@inproceedings{Gebu86,
  author = "Gebauer, R{\"u}diger and M{\"o}ller, H. Michael",
  title = {{Buchberger's algorithm and staggered linear bases}},
  booktitle = "Proc. 1986 Symposium on Symbolic and Algebraic Computation",
  series = "SYMSAC '86",
  year = "1986",
  pages = "218-221",
  publisher = "ACM Press",
  isbn = "0-89791-199-7",
  doi = "http://dx.doi.org/10.1145/32439.32482",
  keywords = "axiomref",
  beebe = "Gebauer:1986:BAS"
}
```

———-

— Gebauer:1986:BAS —

```
@InProceedings{Gebauer:1986:BAS,
  author =       "R{\"u}diger Gebauer and H. Michael M{\"o}ller",
  editor =       "Bruce W. Char",
  booktitle =    "Proceedings of the 1986 Symposium on Symbolic and
                 Algebraic Computation: Symsac '86, July 21--23, 1986,
                 Waterloo, Ontario",
  title =        {{Buchberger's algorithm and staggered linear bases}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "218--221",
  year =         "1986",
  DOI =          "http://dx.doi.org/10.1145.32482",
  ISBN =         "0-89791-199-7",
```

```
  ISBN-13 =      "978-0-89791-199-3",
  LCCN =         "QA155.7.E4 A281 1986",
  bibdate =      "Thu Jul 26 09:06:12 2001",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "ACM order number 505860.",
  acknowledgement = ack-nhfb,
  bookpages =    "254",
}
```

———-

— axiom.bib —

```
@article{Geba88,
  author = "Gebauer, Rudiger and Moller, H. Michael",
  title = {{On an installation of Buchberger's algorithm}},
  journal = "Journal of Symbolic Computation",
  volume = "6",
  number = "2-3",
  pages = "275-286",
  year = "1988",
  abstract =
    "Buchberger's algorithm calculates Groebner bases of polynomial
    ideals. Its efficiency depends strongly on practical criteria for
    detecting superfluous reductions. Buchberger recommends two
    criteria. The more important one is interpreted in this paper as a
    criterion for detecting redundant elements in a basis of a module of
    syzygies. We present a method for obtaining a reduced, nearly minimal
    basis of that module. The simple procedure for detecting (redundant
    syzygies and )superfluous reductions is incorporated now in our
    installation of Buchberger's algorithm in SCRATCHPAD II and REDUCE
    3.3. The paper concludes with statistics stressing the good
    computational properties of these installations.",
  paper = "GM88.pdf",
  keywords = "axiomref",
  beebe = "Gebauer:1988:IBA"
}
```

———-

— Gebauer:1988:IBA —

```
@Article{Gebauer:1988:IBA,
  author =       "R. Gebauer and H. M. M{\"o}ller",
  title =        {{On an installation of Buchberger's algorithm}},
  journal =      j-J-SYMBOLIC-COMP,
  volume =       "6",
  number =       "2-3",
  pages =        "275--286",
  month =        oct # "-" # dec,
  year =         "1988",
  CODEN =        "JSYCEH",
  ISSN =         "0747-7171 (print), 1095-855X (electronic)",
```

```
  ISSN-L =       "0747-7171",
  bibdate =      "Tue Sep 17 08:24:38 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Buchberger's algorithm calculates Gr{\"o}bner bases of
                 polynomial ideals. Its efficiency depends strongly on
                 practical criteria for detecting superfluous
                 reductions. Buchberger recommends two criteria. The
                 more important one is interpreted in this paper as a
                 criterion for detecting redundant elements in a basis
                 of a module of syzygies The authors present a method
                 for obtaining a reduced, nearly minimal basis of that
                 module. The simple procedure for detecting (redundant
                 syzygies and) superfluous reductions is incorporated
                 now in the installation of Buchberger's algorithm in
                 SCRATCHPAD II and REDUCE 3.3. The paper concludes with
                 statistics stressing the good computational properties
                 of these installations.",
  acknowledgement = ack-nhfb,
  affiliation =  "Springer-Verlag, New York, NY, USA",
  classification = "C4130 (Interpolation and function approximation);
                 C6130 (Data handling techniques); C7310 (Mathematics)",
  fjournal =     "Journal of Symbolic Computation",
  link =  "\url{http://www.sciencedirect.com/science/journal/07477171}",
  keywords =     "Buchberger algorithm installation; Gr{\"o}bner bases;
                 Polynomial ideals; Superfluous reductions; Redundant
                 elements; Module of syzygies; SCRATCHPAD II; REDUCE
                 3.3; Computational properties",
  language =     "English",
  pubcountry =   "UK",
  thesaurus =    "Polynomials; Symbol manipulation",
}
```

———-

— axiom.bib —

```
@book{Gedd92,
  author = "Geddes, Keith and Czapor, O. and Stephen R. and Labahn, George",
  title = {{Algorithms For Computer Algebra}},
  year = "1992",
  publisher = "Kluwer Academic Publishers",
  isbn = "0-7923-9259-0",
  month = "September",
  abstract =
    "Computer Algebra (CA) is the name given to the discipline of
    algebraic, rather than numerical, computation. There are a number of
    computer programs  Computer Algebra Systems (CASs)  available for
    doing this. The most widely used general-purpose systems that are
    currently available commercially are Axiom, Derive, Macsyma, Maple,
    Mathematica and REDUCE. The discipline of computer algebra began in
    the early 1960s and the first version of REDUCE appeared in 1968.

    A large class of mathematical problems can be solved by using a CAS
```

purely interactively, guided only by the user documentation. However, sophisticated use requires an understanding of the considerable amount of theory behind computer algebra, which in itself is an interesting area of constructive mathematics. For example, most systems provide some kind of programming language that allows the user to expand or modify the capabilities of the system.

This book is probably the most general introduction to the theory of computer algebra that is written as a textbook that develops the subject through a smooth progression of topics. It describes not only the algorithms but also the mathematics that underlies them. The book provides an excellent starting point for the reader new to the subject, and would make an excellent text for a postgraduate or advanced undergraduate course. It is probably desirable for the reader to have some background in abstract algebra, algorithms and programming at about second-year undergraduate level.

The book introduces the necessary mathematical background as it is required for the algorithms. The authors have avoided the temptation to pursue mathematics for its own sake, and it is all sharply focused on the task of performing algebraic computation. The algorithms are presented in a pseudo-language that resembles a cross between Maple and C. They provide a good basis for actual implementations although quite a lot of work would still be required in most cases. There are no code examples in any actual programming language except in the introduction.

The authors are all associated with the group that began the development of Maple. Hence, the book reflects the approach taken by Maple, but the majority of the discussion is completely independent of any actual system. The authors experience in implementing a practical CAS comes across clearly.

The book focuses on the core of computer algebra. The first chapter introduces the general concept and provides a very nice historical survey. The next three chapters discuss the fundamental topics data structures, representations and the basic arithmetic of integers, rational numbers, multivariate polynomials and rational functions on which the rest of the book is built.

A major technique in CA involves projection onto one or more homomorphic images, for which the ground ring is usually chosen to be a finite field. The image solution is lifted back to the original problem domain by means of the Chinese Remainder Theorem in the case of multiple homomorphic images, or the Hensel (-adic or ideal-adic) construction in the case of a single image. The next two chapters are devoted to these techniques in a fairly general setting. The two subsequent chapters specialise them to GCD computation and factorisation for multivariate polynomials; the first of these chapters also discusses the important but difficult topic of subresultants.

The next two chapters describe the use of fraction-free Gaussian elimination, resultants and Grbner Bases for manipulation and exact

```
    solution of linear and nonlinear polynomial equations. The two final
    chapters describe ‘‘classical’’ algorithms and the more recent Risch
    algorithm for symbolic indefinite integration, and provide an
    introduction to differential algebra.

    The book does not consider more specialised problem areas such as
    symbolic summation, definite integration, differential equations,
    group theory or number theory. Nor does it consider more applied
    problem areas such as vectors, tensors, differential forms, special
    functions, geometry or statistics, even though Maple and other CASs
    provide facilities in all or many of these areas. It does not consider
    questions of CA programming language design, nor any of the important
    but non-algebraic facilities provided by current CASs such as their
    user interfaces, numerical and graphical facilities.

    This is a long book (nearly 600 pages); it is generally very well
    presented and the three authors have merged their contributions
    seamlessly. I noticed very few typographical errors, and none of any
    consequence. I have only two complaints about the book. The typeface
    is too small, particularly for the relatively large line spacing used,
    and it is much too expensive, particularly for a book that would
    otherwise be an excellent student text. I recommend it highly to
    anyone who can afford it.",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Gianni 87]{Gia87} Gianni, Patrizia
  title = {{Primary Decomposition of Ideals}},
in [Wit87], pp12-13
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Gianni 89a]{Gia89} Gianni, P. (Patrizia) (ed)
Symbolic and Algebraic Computation.
International Symposium ISSAC ’88, Rome, Italy, July 4-8, 1988. Proceedings,
volume 358 of Lecture Notes in Computer Science. Springer-Verlag, Berlin,
Germany / Heildelberg, Germany / London, UK / etc., 1989. ISBN 3-540-51084-2
LCCN QA76.95.I57 1988 Conference held jointly with AAECC-6
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Gian89,
  author = "Gianni, Patrizia and Mora, T.",
  title = {{Algebraic solution of systems of polynomial equations
           using Groebner bases.}},
  booktitle = "Applied Algebra, Algebraic Algorithms and Error-Correcting
               Codes",
  series = "AAECC-5",
  pages = "247-257",
  year = "1989",
  isbn = "3-540-51082-6",
  abstract =
    "One of the most important applications of Buchberger's algorithm for
    Groebner basis computation is the solution of systems of polynomial
    equations (having finitely many roots), i.e. the computation of zeros
    of 0-dimensional polynomial ideals. It is based on a relation between
    Groebner bases w.r.t. a lexicographical ordering and elimination
    ideals, which was discovered by Trinks.

    Packages for isolation of real roots of systems of polynomial
    equations using Groebner basis computation are currently available in
    different computer algebra systems, including SAC-2, Reduce,
    Scratchpad II, Maple.

    In principle, Buchberger-Trinks algorithm should allow to compute
    solutions of such systems in the algebraic closure of the coefficient
    field $k$ (usually the rational numbers), in the sense that it is
    possible to represent explicitly a finite extension of $k$ containing
    all solutions and to express the roots in this field.

    However, this requires several factorisations of polynomials over a
    tower of algebraic extensions of $k$, which is usually very costly, so
    that the resulting algorithm is not very feasible and, as far as we
    know, no implementation is available.

    The results of [GT2] on primary decomposition of ideals include a
    thorough study on the structure of Groebner bases for 0-dimensional
    ideals; in particular, the paper shows, that after a ``generic''
    linear change of coordinates, the roots of a system of polynomial
    equations can be expressed in a simple extension of $k$. Therefore, in
    this case, no factorisation of polynomials over towers of algebraic
    extensions is needed.

    However performing a change of coordinates has the undesirable effects
    of introducing dense polynomials and of increasing the size of
    coefficients.

    The problem then arises of producing strategies to compute Groebner
    bases for (0-dimensional) ideals, which at least are able to control
    the influence of these side-effects: two such strategies are presented
    in this paper, together with the application to the present problem of
    an algorithm by Gianni that computes the radical of a 0-dimensional
    ideal after a ``generic'' change of coordinates.

    A different approach, based on her ``splitting algorithm'', to compute
```

```
    solutions of systems of polynomial equations without the need of
    polynomial factorisations has been proposed by D. Duval; also her
    algorithm should be simplified by a ``generic'' change of coordinates.

    The algorithms discussed in this paper are implemented in SCRATCHPAD II.

    In the first section we recall some well-known properties of Groebner
    bases and properties on the structure of Groebner bases of
    zero-dimensional ideals from [GT2]; in the second section we recall
    the Groebner basis algorithm for solving systems of algebraic
    equations.

    The original results are contained in Sections 3 to 5; in Section 3 we
    take advantage of the obvious fact that density can be controlled by
    performing ``small'' changes of coordinates: we show that such
    approach is possible during a Groebner basis computation, in such a
    way that computations done before a change of coordinates are valid
    also after it; in Section 4 we propose a ``linear algebra'' approach
    to obtain the Groebner basis w.r.t the lexicographical ordering from
    the one w.r.t the total-degree ordering; in Section 5, we present a
    zero-dimensional radical algorithm and show how to apply it to the
    present problem.",
  paper = "Gian89.pdf",
  keywords = "axiomref",
  beebe = "Gianni:1989:ASS"
}
```

———-

— Gianni:1989:ASS —

```
@InProceedings{Gianni:1989:ASS,
  author =       "P. Gianni and T. Mora",
  title =        {{Algebraic solution of systems of polynomial equations
                 using Gr\"o}bner bases}},
  crossref =     "Huguet:1989:AAA",
  pages =        "247--257",
  month =        "",
  year =         "1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "One of the most important applications of Buchberger's
                 algorithm for Gr{\"o}bner basis computation is the
                 solution of systems of polynomial equations (having
                 finitely many roots), i.e. the computation of zeros of
                 0-dimensional polynomial ideals. It is based on a
                 relation between Gr{\"o}bner bases w.r.t. a
                 lexicographical ordering and elimination ideals. The
                 algorithms discussed in this paper are implemented in
                 SCRATCHPAD II. In the first section the authors recall
                 some well-known properties of Gr{\"o}bner bases and
                 properties on the structure of Gr{\"o}bner bases of
                 zero-dimensional ideals; in the second section they
```

```
                recall the Gr{\"o}bner basis algorithm for solving
                systems of algebraic equations. The original results
                are then presented. The authors first take advantage of
                the obvious fact that density can be controlled
                performing 'small' changes of coordinates: they show
                that such approach is possible during a Gr{\"o}bner
                basis computation, in such a way that computations done
                before a change of coordinates are valid also after it;
                they propose a 'linear algebra' approach to obtain the
                Gr{\"o}bner basis w.r.t. the lexicographical ordering
                from the one w.r.t. the total-degree ordering; and
                finally they present a zero-dimensional radical
                algorithm and show how to apply it to the present
                problem.",
  acknowledgement = ack-nhfb,
  affiliation =  "Pisa Univ., Italy",
  classification = "C1110 (Algebra); C4140 (Linear algebra); C7310
                (Mathematics)",
  keywords =     "Coordinate changes; Polynomial equations; Gr{\"o}bner
                bases; Buchberger's algorithm; Gr{\"o}bner basis
                computation; Zeros; 0-Dimensional polynomial ideals;
                Lexicographical ordering; Elimination ideals;
                SCRATCHPAD II; Algebraic equations; Linear algebra;
                Total-degree ordering; Zero-dimensional radical
                algorithm",
  language =     "English",
  thesaurus =    "Equations; Linear algebra; Mathematics computing;
                Poles and zeros; Polynomials",
}
```

———-

— axiom.bib —

```
@inproceedings{Gilx92,
  author = "Gil, Isabelle",
  title = {{Computation of the Jordan canonical form of a square matrix
           (using the Axiom programming language)}},
  booktitle = "Proc ISSAC 1992",
  series = "ISSAC '92",
  year = "1992",
  publisher = "ACM",
  pages = "138-145",
  isbn = "0-89791-489-9 (soft cover), 0-89791-490-2 (hard cover)",
  abstract =
    "Presents an algorithm for computing: the Jordan form of a square
    matrix with coefficients in a field K using the computer algebra
    system Axiom. This system presents the advantage of allowing generic
    programming. That is to say, the algorithm can first be implemented
    for matrices with rational coefficients and then generalized to
    matrices with coefficients in any field.  Therefore the author
    presents the general method which is essentially based on the use of
    the Frobenius form of a matrix in order to compute its Jordan form;
```

```
    and then restricts attention to matrices with rational
    coefficients. On the one hand the author streamlines the algorithm
    froben which computes the Frobenius form of a matrix, and on the other
    she examines in some detail the transformation from the Frobenius form
    to the Jordan form, and gives the so called algorithm Jordform. The
    author studies in particular, the complexity of this algorithm and
    proves that it is polynomial when the coefficients of the matrix are
    rational. Finally the author gives some experiments and a conclusion.",
  keywords = "axiomref",
  beebe = "Gil:1992:CJC"
}
```

———-

— Gil:1992:CJC —

```
@InProceedings{Gil:1992:CJC,
  author =       "I. Gil",
  title =        {{Computation of the {Jordan} canonical form of a square
                 matrix (using the {Axiom} programming language)}},
  crossref =     "Wang:1992:ISS",
  pages =        "138--145",
  month =        "",
  year =         "1992",
  bibdate =      "Tue Sep 17 06:35:39 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Presents an algorithm for computing: the Jordan form
                 of a square matrix with coefficients in a field K using
                 the computer algebra system Axiom. This system presents
                 the advantage of allowing generic programming. That is
                 to say, the algorithm can first be implemented for
                 matrices with rational coefficients and then
                 generalized to matrices with coefficients in any field.
                 Therefore the author presents the general method which
                 is essentially based on the use of the Frobenius form
                 of a matrix in order to compute its Jordan form; and
                 then restricts attention to matrices with rational
                 coefficients. On the one hand the author streamlines
                 the algorithm froben which computes the Frobenius form
                 of a matrix, and on the other she examines in some
                 detail the transformation from the Frobenius form to
                 the Jordan form, and gives the so called algorithm
                 Jordform. The author studies in particular, the
                 complexity of this algorithm and proves that it is
                 polynomial when the coefficients of the matrix are
                 rational. Finally the author gives some experiments and
                 a conclusion.",
  acknowledgement = ack-nhfb,
  affiliation =  "LMC, IMAG, Grenoble, France",
  classification = "C4130 (Interpolation and function approximation);
                 C4140 (Linear algebra); C4240 (Programming and
                 algorithm theory); C7310 (Mathematics)",
  keywords =     "Axiom programming language; Complexity; Computer
```

```
                algebra system; Froben; Frobenius form; Generic
                programming; Jordan canonical form; Jordform;
                Polynomial; Rational coefficients; Square matrix",
  language =    "English",
  thesaurus =   "Computational complexity; Matrix algebra; Polynomials;
                Symbol manipulation",
}
```

———-

— **axiom.bib** —

```
@phdthesis{Gome92,
  author = "Gomez-Dias, Teresa",
  title = {{Quelques applications de l'\'evaluation dynamique}},
  school = "L'Universite de Limoges",
  year = "1992",
  month = "March",
  paper = "Gome92.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gome93,
  author = "Gomez-Diaz, Teresa",
  title = {{Examples of using Dynamic Constructible Closure}},
  booktitle = "IMACS Symposium SC-1993",
  year = "1993",
  abstract = "
    We present here some examples of using the ``Dynamic Constructible
    Closure'' program, which performs automatic case distinction in
    computations involving parameters over a base field $K$. This program
    is an application of the ``Dynamic Evaluation'' principle, which
    generalizes traditional evaluation and was first used to deal with
    algebraic numbers.",
  paper = "Gom93.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Gome94,
  author = "Gomez-Diaz, Teresa",
  title = {{The Possible Solutions to the Control Problem}},
  year = "1994"
}
```

———-

— **axiom.bib** —

```
@article{Gome96,
  author = "Gomez-Diaz, Theresa",
  title = {{Examples of using dynamic constructible closure}},
  journal = "Math. Comput. Simul.",
  volume = "42",
  number = "4-6",
  pages = "375-383",
  year = "1996",
  abstract =
    "We present here some examples of using the ``Dynamic Constructible
    Closure'' program, which performs automatic case distinctions in
    computations involving parameters over a base field ``K''. This
    program is an application of the ``Dynamic Evaluation'' principle
    which generalizes tradional evaluation and was first used to deal with
    algebraic numbers.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Gonn05,
  author = "Gonnet, Gaston and Haigh, Thomas",
  title = {{An Interview with Gaston Gonnet}},
  year = "2005",
  publisher = "SIAM",
  link = "\url{http://history.siam.org/pdfs2/Gonnet_final.pdf}",
  abstract =
    "Born in Uruguay, Gonnet was first exposed to computers while working
    for IBM in Montevideo as a young man. This led him to a position at
    the university computer center, and in turn to an undergraduate degree
    in computer science in 1973. In 1974, following a military coup, he
    left for graduate studies in computer science at the University of
    Waterloo. Gonnet earned an M.Sc. and a Ph.D. in just two and a half
    years, writing a thesis on the analysis of search algorithms under the
    supervision of Alan George. After one year teaching in Rio de Janeiro
    he returned to Waterloo, as a faculty member.

    In 1980, Gonnet began work with a group including Morven Gentleman and
    Keith Geddes to produce an efficient interactive computer algebra
    system able to work well on smaller computers: Maple. Gonnet discusses
    in great detail the goals and organization of the Maple project, its
    technical characteristics, the Maple language and kernel, the Maple
    library, sources of funding, the contributions of the various team
    members, and the evolution of the system over time. He compares the
    resulting system to MACSYMA, Mathematica, Reduce, Scratchpad and other
    systems. Gonnet also examines the licensing and distribution of Maple
    and the projects relations to its users. Maple was initially used for
```

```
    teaching purposes within the university, but soon found users in other
    institutions. From 1984, distribution was handled by Watcom, a company
    associated with the university, and 1988, Gonnet and Geddes created a
    new company, Waterloo Maple Software, Inc. to further commercialize
    Maple, which established itself as the leading commercial computer
    algebra system. However, during the mid-1990s the company ran into
    trouble and disagreements with his colleagues caused Gonnet to
    withdraw from managerial involvement. Since then, he feels that Maple
    has lost its battle with Mathematica. Gonnet also discusses Maples
    relation to Matlab and its creator, Cleve Moler.

    From 1984 onward with Frank Tompa, Tim Bray, and other Waterloo
    colleagues, Gonnet worked on the production of computer software to
    support the creation of the second edition of the Oxford English
    Dictionary. This led to the creation of another startup company, Open
    Text, producing software for the searching and indexing of textual
    information within large corporations. Gonnet explains his role in the
    firm, including his departure and his feeling that it made a strategic
    blunder by not exploiting its early lead in Internet search.

    Gonnet continued to work in a number of areas of computer science,
    including analysis of algorithms. In 1990, Gonnet moved from Waterloo
    to ETH in Switzerland. Among his projects since then have been Darwin,
    a bioinformatics system for the manipulation of genetic data, and
    leadership of the OpenMath project to produce a standard
    representation for mathematical objects. He has been involved in
    several further startup companies, including Aruna, a relational
    database company focused on business intelligence applications.",
  keywords = "axiomref",
  paper = "Gonn05.pdf"
}
```

———-

— axiom.bib —

```
@inproceedings{Good91,
  author = "Goodwin, B. M. and Buonopane, R. A. and Lee, A.",
  title = {{Using MathCAD in teaching material and energy balance concepts}},
  booktitle = "Challenges of a Changing World",
  comment = "Proc. 1991 Ann. Conf., Amer. Soc. for Engineering Education",
  pages = "345-349",
  year = "1991",
  abstract =
    "We show how PC-based applications software, specifically MathCAD, is
    used in the teaching of material and energy balance concepts. MathCAD
    is a microcomputer software package which acts as a mathematical
    scratchpad. It has proven to be a very useful instructional tool in
    introductory chemical engineering courses. MathCAD solutions to
    typical course problems are presented.",
  keywords = "axiomref",
  beebe = "Goodwin:1991:UMT"
}
```

———-

— **Goodwin:1991:UMT** —

```
@InProceedings{Goodwin:1991:UMT,
  author =       "B. M. Goodwin and R. A. Buonopane and A. Lee",
  title =        {{Using MathCAD in teaching material and energy
                 balance concepts}},
  crossref =     "Anonymous:1991:PAC",
  pages =        "345--349 (vol. 1)",
  year =         "1991",
  bibdate =      "Tue Sep 17 06:37:45 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The authors show how PC-based applications software,
                 specifically MathCAD, is used in the teaching of
                 material and energy balance concepts. MathCAD is a
                 microcomputer software package which acts as a
                 mathematical scratchpad. It has proven to be a very
                 useful instructional tool in introductory chemical
                 engineering courses. MathCAD solutions to typical
                 course problems are presented.",
  acknowledgement = ack-nhfb,
  affiliation =  "Northeastern Univ., Boston, MA, USA",
  classification = "C7450 (Chemical engineering); C7810C (Computer-aided
                 instruction)",
  keywords =     "Energy balance concepts; Instructional tool;
                 Introductory chemical engineering courses; MathCAD;
                 Mathematical scratchpad; PC-based applications
                 software",
  language =     "English",
  thesaurus =    "Chemical engineering computing; Computer aided
                 instruction; Microcomputer applications; Spreadsheet
                 programs",
}
```

———-

— **ignore** —

```
\bibitem[Golden 4]{GH84} Golden, V. Ellen; Hussain, M. A. (eds)
Proceedings of the 1984 MACSYMA Users' Conference:
Schenectady, New York, July 23-25, 1984, General Electric,
Schenectady, NY, USA, 1984
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Gonnet 96]{Gon96} Gonnet, Gaston H.
  title = {{Official verion 1.0 of the Meta Content Dictionary}},
```

```
  link = "\url{http://www.inf.ethz.ch/personal/gonnet/ContDict/Meta}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Good93,
  author = "Goodloe, A. and Loustaunau, Philippe",
  title = {{An abstract data type development of graded rings}},
  booktitle = "Design and Implementation of Symbolic Computation Systems",
  series = "Lecture Notes in Computer Science 721",
  pages = "193-202",
  isbn = "0-387-57272-4 (New York), 3-540-57272-4 (Berlin)",
  year = "1993",
  abstract =
    "Recently new computer algebra systems such as Scratchpad and Weyl have
    been developed with built in mechanisms for expressing abstract data types.
    These systems are object oriented in that they incorporate multiple
    inheritance and polymorphic types. Davenport and Trager have build much
    of the framework for basic commutative algebra in Scratchpad II
    utilizing its rich set of abstraction mechanisms. Davenport and Trager
    concentrated on developing factorization algorithms on domains which
    were abstract data types.

    We are taking a similar approach to the development of algorithms for
    computing in graded rings. The purpose of this paper is to develop the
    tools required to compute with polynomials with coefficients in a graded
    ring $R$. We focus on graded rings $R$ which are polynomial rings graded
    by a monoid, and we allow partial orders on the monomials. The ideas
    presented here can be applied to more general graded rings $R$, such as
    associated graded rings to filtered rings, as long as certain computational
    ``requirements'' are satisfied",
  keywords = "axiomref",
  beebe = "Goodloe:1993:ADT"
}
```

———-

— **Goodloe:1993:ADT** —

```
@InProceedings{Goodloe:1993:ADT,
  author =       "A. Goodloe and P. Loustaunau",
  title =        {{An abstract data type development of graded rings}},
  crossref =     "Fitch:1993:DIS",
  pages =        "193--202",
  month =        "",
  year =         "1993",
  bibdate =      "Tue Sep 17 06:37:45 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Novel computer algebra systems such as Scratchpad and
                 Weyl have been developed with built in mechanisms for
                 expressing abstract data types. These systems are
```

```
                object oriented in that they incorporate multiple
                inheritance and polymorphic types. The authors are
                taking a similar approach to the development of
                algorithms for computing in graded rings. They develop
                the tools required to compute with polynomials with
                coefficients in a graded ring R. They focus on graded
                rings R which are polynomial rings graded by a monoid,
                and allow partial orders on the monomials. The ideas
                presented can be applied to more general graded rings
                R, such as associated graded rings to filtered rings,
                as long as certain computational 'requirements' are
                satisfied.",
  acknowledgement = ack-nhfb,
  affiliation =  "Dept. of Math., George Mason Univ., Fairfax, VA, USA",
  classification = "C4130 (Interpolation and function approximation);
                C6120 (File organisation); C7310 (Mathematics)",
  keywords =     "Abstract data types; Computer algebra systems;
                Filtered rings; Graded rings; Monoid; Multiple
                inheritance; Object oriented; Partial orders;
                Polymorphic types; Polynomial rings; Scratchpad; Weyl",
  language =     "English",
  thesaurus =    "Abstract data types; Polynomials; Symbol
                manipulation",
}
```

———-

— **axiom.bib** —

```
@misc{Grab98,
  author = "Grabe, Hans-Gert",
  title = {{About the Polynomial System Solve Facility of Axiom, Macsyma,
           Maple Mathematica, MuPAD, and Reduce}},
  link = "\url{https://www.informatik.uni-leipzig.de/~graebe/ComputerAlgebra/Publications/WesterBook.pdf}",
  abstract =
    "We report on some experiences with the general purpose Computer
    Algebra Systems (CAS) Axiom, Macsyma, Maple, Mathematica, MuPAD, and
    Reduce solving systems of polynomial equations and the way they
    present their solutions. This snapshot (taken in the spring of 1996)
    of the current power of the different systems in a special area
    concentrates on both CPU-times and the quality of the output.",
  paper = "Grab98.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Grab02,
  author = "Grabe, Hans-Gert",
  title = {{The SymbolicData Benchmark Problems Collection of Polynomial
```

```
          Systems}},
  booktitle = "Workshop on Under- and Overdetermined Systems of Algebraic or
               Differential Equations",
  location = "Karlsruhe, Germany",
  pages = "57-76",
  year = "2002",
  link = "\url{http://symbolicdata.org/Papers/karlsruhe-02.pdf}",
  paper = "Grab02.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@misc{Grab06,
  author = "Grabe, Hans-Gert",
  title = {{The Groebner Factorizer and Polynomial System Solving}},
  year = "2006",
  report = "Special Semester on Groebner Bases",
  location = "Linz",
  link = "\url{https://www.ricam.oeaw.ac.at/specsem/srs/groeb/download/06\_02\_Solver.pdf}",
  abstract =
    "Let $S := k[x_1,\ldots, x_n]$ be the polynomial ring in the
    variables $x_1,\ldots,x_n$ over the field $k$ and
    $B := \{f_1,\ldots,f_m\} \subset S$
    be a finite system of polynomials. Denote by $I(B)$ the
    ideal generated by these polynomials. One of the major tasks of
    constructive commutative algebra is the derivation of information
    about the structure of
    \[V(B):=\{a \in K^n : \forall f \in B{\rm\ such\ that\ }f(a)=0\}\]
    the set of common zeroes of the system $B$ over an
    algebraically closed extension $K$ of $k$.  Splitting the system into
    smaller ones, solving them separately, and patching all solutions
    together is often a good guess for a quick solution of even highly
    nontrivial problems. This can be done by several techniques, e.g.,
    characteristic sets, resultants, the Groebner factorizer or some ad
    hoc methods. Of course, such a strategy makes sense only for problems
    that really will split, i.e., for reducible varieties of
    solutions. Surprisingly often, problems coming from 11real life''
    fulfill this condition.

    Among the methods to split polynomial systems into smaller pieces
    probably the Groebner factorizer method attracted the most
    theoretical attention, see Czapor ([4, 5]), Davenport ([6]), Melenk, M
    oller and Neun ([16, 17]) and Gr abe ([13, 14]). General purpose
    Computer Algebra Systems (CAS) are well suited for such an approach,
    since they make available both a (more or less) well tuned
    implementation of the classical Groebner algorithm and an effective
    multivariate polynomial factorizer.

    Furthermore it turned out that the Groebner factorizer is not only a
    good heuristic approach for splitting, but its output is also usually
```

```
    a collection of almost prime components. Their description allows a
    much deeper understanding of the structure of the set of zeroes
    compared to the result of a sole Groebner basis computation.

    Of course, for special purposes a general CAS as a multipurpose
    mathematical assistant cant offer the same power as specialized
    software with efficiently implemented and well adapted algorithms and
    data types. For polynomial system solving, such specialized software
    has to implement two algorithmically complex tasks, solving and
    splitting, and until recently none of the specialized systems (as
    e.g., GB, Macaulay, Singular, CoCoA, etc.) did both
    efficiently. Meanwhile, being very efficient computing (classical)
    Groebner bases, development efforts are also directed, not only
    for performance reasons, towards a better inclusion of factorization
    into such specialized systems.  Needless to remark that it needs some
    skill to force a special system to answer questions and the user will
    probably first try his ``home system'' for an answer. Thus the
    polynomial systems solving facility of the different CAS should behave
    especially well on such polynomial systems that are hard enough not to
    be done by hand, but not really hard to require special efforts. It
    should invoke a convenient interface to get the solutions in a form
    that is (correct and) well suited for further analysis in the familiar
    environment of the given CAS as the personal mathematical assistant.",
  paper = "Grab06.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@misc{Grab91,
  author = "Grabmeier, Johannes and Huber, K. and Krieger, U.",
  title = {{Das ComputeralgebraSystem AXIOM bei kryptologischen und
            verkehrstheoretischen Untersuchungen des Forschunginstituts
            der Deutschen Bundespost TELEKOM'}},
  type = "technical report",
  number = "TR 75.91.20",
  location = "Heidelberg, Germany",
  year = "1991",
  keywords = "axiomref",
  beebe = "Grabmeier:1991:CSA"
}
```

———-

— Grabmeier:1991:CSA —

```
@TechReport{Grabmeier:1991:CSA,
  author =       "J. Grabmeier and K. Huber and U. Krieger",
  title =        "{Das Computeralgebra-System AXIOM bei kryptologischen
                 und verkehrstheoretischen Untersuchungen des
                 Forschungsinstituts der Deutschen Bundespost TELEKOM}",
```

```
  type =         "Technischer Report",
  number =       "TR 75.91.20",
  institution =  "IBM Wissenschaftliches Zentrum",
  address =      "Heidelberg, Germany",
  pages =        "??",
  year =         "1991",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

— **axiom.bib** —

```
@article{Grab91b,
  author = "Grabmeier, Johannes",
  title = {{Axiom, ein Computeralgebrasystem mit abstrakten Datentypen}},
  journal = "mathPAD",
  volume = "1",
  number = "3",
  pages = "13-15",
  year = "1991",
  paper = "Grab91b.pdf",
  keywords = "axiomref"
}
```

— **ignore** —

```
\bibitem[Grabmeier 03]{GKW03} Grabmeier, Johannes; Kaltofen, Erich;
Weispfenning, Volker (eds)
Computer algebra handbook: foundations, applications, systems.
Springer-Verlag, Berlin, Germany / Heildelberg, Germany / London, UK / etc.,
2003. ISBN 3-540-65466-6 637pp Includes CDROM
  link = "\url{http://www.springer.com/sgw/cda/frontpage/0,11855,1-102-22-1477871-0,00.html}",
  keywords = "axiomref"
```

— **axiom.bib** —

```
@book{Gree01,
  author = "Green, Edward L.",
  title = {{Symbolic Computation: Solving Equations in Algebra, Geometry, and
           Engineering}},
  booktitle = "Proc. AMS-IMS-SIAM Joint Summer Research Conference on Symbolic
               Computation",
  volume = "232",
  publisher = "American Mathematical Society",
  year = "2001",
```

```
  abstract =
    "This volume contains papers related to the research conference,
    ``Symbolic Computation: Solving Equations in Algebra, Analysis, and
    Engineering,'' held at Mount Holyoke College (MA). It provides a broad
    range of active research areas in symbolic computation as it applies
    to the solution of polynomial systems. The conference brought together
    pure and applied mathematicians, computer scientists, and engineers,
    who use symbolic computation to solve systems of equations or who
    develop the theoretical background and tools needed for this
    purpose. Within this general framework, the conference focused on
    several themes: systems of polynomials, systems of differential
    equations, noncommutative systems, and applications.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Grie70,
  author = "Griesmer, James H. and Jenks, Richard D.",
  title = {{SCRATCHPAD/1 -- an interactive facility for symbolic mathematics}},
  booktitle = "Proc. second ACM Symposium on Symbolic and Algebraic
               Manipulation",
  series = "SYMSAC 71",
  year = "1971",
  pages = "42--58",
  doi = "http://dx.doi.org/10.1145806266",
  link =
    "\url{http://delivery.acm.org/10.1145/810000/806266/p42-griesmer.pdf}",
  abstract = "
    The SCRATCHPAD/1 system is designed to provide an interactive symbolic
    computational facility for the mathematician user. The system features
    a user language designed to capture the style and succinctness of
    mathematical notation, together with a facility for conveniently
    introducing new notations into the language. A comprehensive system
    library incorporates symbolic capabilities provided by such systems as
    SIN, MATHLAB, and REDUCE.",
  paper = "Grie70.pdf",
  keywords = "axiomref,printed",
  beebe = "Griesmer:1971:SIF"
}
```

———-

— **Griesmer:1971:SIF** —

```
@InProceedings{Griesmer:1971:SIF,
  author =       "J. H. Griesmer and R. D. Jenks",
  title =        {{SCRATCHPAD/1 --- an interactive facility for
                 symbolic mathematics}},
  crossref =     "Petrick:1971:PSS",
  pages =        "42--58",
```

```
  year =         "1971",
  DOI =          "http://dx.doi.org/10.1145806266",
  bibdate =      "Thu Jul 26 08:45:53 2001",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/obscure.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://delivery.acm.org/10.1145/810000/806266/p42-griesmer.pdf}",
}
```

———-

— **axiom.bib** —

```
@techreport{Grie72a,
  author = "Griesmer, James H. and Jenks, Richard D.",
  title = {{Experience with an online symbolic math system SCRATCHPAD}},
  institution = "IBM",
  year = "1972",
  isbn = "0-903796-02-3",
  keywords = "axiomref",
  beebe = "Griesmer:1972:EOSb"
}
```

———-

— **Griesmer:1972:EOSb** —

```
@InProceedings{Griesmer:1972:EOSb,
  author =       "J. Griesmer and R. Jenks",
  title =        {{Experience with an online symbolic math. system
                 SCRATCHPAD}},
  crossref =     "Online:1972:OCP",
  pages =        "??--??",
  year =         "1972",
  bibsource =    "/usr/local/src/bib/bibliography/Distributed/QLD.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  bydate =       "Le",
  byrev =        "Le",
  date =         "00/00/00",
  descriptors =  "Formula manipulation",
  enum =         "1209",
  language =     "English",
  location =     "PKI-OG: Li-Ord.Le",
  references =   "0",
  revision =     "21/04/91",
}
```

———-

— **axiom.bib** —

```
@article{Grie72,
  author = "Griesmer, James H. and Jenks, Richard D.",
```

```
  title = {{SCRATCHPAD: A capsule view}},
  journal = "ACM SIGPLAN Notices",
  volume = "7",
  number = "10",
  pages = "93-102",
  year = "1972",
  comment = "Proc. Symp. Two-dimensional man-machine communications",
  doi = "http://dx.doi.org/10.1145807019",
  abstract =
    "SCRATCHPAD is an interactive system for algebraic manipulation
    available under the CP/CMS time-sharing system at Yorktown Heights. It
    features an extensible declarative language for the interactive
    formulation of symbolic computations. The system is a large and
    complex body of LISP programs incorporating significant portions of
    other symbolic systems. Here we present a capsule view of SCRATCHPAD,
    its language and its capabilities. This is followed by an example
    which illustrates its use in an application involving the solution of
    an integral equation.",
  paper = "Grie72.pdf",
  keywords = "axiomref, printed, DONE",
  beebe = "Griesmer:1972:SCV"
}
```

———-

— Griesmer:1972:SCV —

```
@Article{Griesmer:1972:SCV,
  author =       "James H. Griesmer and Richard D. Jenks",
  title =        {{SCRATCHPAD: A capsule view}},
  journal =      j-SIGPLAN,
  volume =       "7",
  number =       "10",
  pages =        "93--102",
  year =         "1972",
  CODEN =        "SINODQ",
  DOI =          "http://dx.doi.org/10.1145807019",
  ISSN =         "0362-1340 (print), 1523-2867 (print), 1558-1160
                 (electronic)",
  ISSN-L =       "0362-1340",
  bibdate =      "Thu Jul 26 10:33:16 2001",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "Proceedings of the symposium on Two-dimensional
                 man-machine communication, Mark B. Wells and James B.
                 Morris (eds.)",
  acknowledgement = ack-nhfb,
  bookpages =    "iii + 160",
  fjournal =     "ACM SIGPLAN Notices",
  link =  "\url{http://portal.acm.org/browse_dl.cfm?idx=J706}",
}
```

———-

— **axiom.bib** —

```
@article{Grie74,
  author = "Griesmer, James H. and Jenks, Richard D.",
  title = {{A solution to problem \#4: the lie transform}},
  journal = "SIGSAM Bulletin",
  volume = "8",
  number = "4",
  pages = "12-13",
  year = "1974",
  abstract =
    "The following SCRATCHPAD conversation for carrying out the Lie
    Transform computation represents a slight modification of one written
    by Dr. David Barton, when he was a summer visitor during 1972 at the
    Watson Research Center.",
  paper = "Grie74.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Grie75,
  author = "Griesmer, James H. and Jenks, Richard D. and Yun, David Y.Y",
  title = {{SCRATCHPAD User's Manual}},
  institution = "IBM",
  year = "1975",
  type = "Research Report",
  number = "RA70",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Grie75a,
  author = "Griesmer, James H. and Jenks, Richard D. and Yun, David Y.Y.",
  title = {{A SCRATCHPAD solution to problem \#7}},
  journal = "SIGSAM",
  volume = "9",
  number = "3",
  pages = "13-17",
  year = "1975",
  paper = "Grie75a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Grie75b,
  author = "Griesmer, James H. and Jenks, Richard D. and Yun, David Y.Y.",
  title = {{A FORMAT statement in SCRATCHPAD}},
  journal = "SIGSAM",
  volume = "9",
  number = "3",
  pages = "24-25",
  year = "1975",
  abstract =
    "Algebraic manipulation covers branches of software, particularly list
    processing, mathematics, notably logic and number theory, and
    applications largely in physics. The lectures will deal with all of these
    to a varying extent.",
  paper = "Grie75b.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Griesmer 76]{GJY76} Griesmer, J.H.; Jenks, R.D.; Yun, D.Y.Y
  title = {{A Set of SCRATCHPAD Examples}},
April 1976 (private copy)
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Grie79,
  author = "Griesmer, James H.",
  title = {{The state of symbolic computation}},
  journal = "SIGSAM Bulletin",
  volume = "13",
  number = "3",
  pages = "25-28",
  year = "1979",
  abstract =
    "When I was first asked to give this banquet talk, I was somewhat
    hesitant to accept. I have not been very heavily involved in the field
    of symbolic and algebraic manipulation for the past three years. I
    have contented myself with such behind-the-scenes activities as
    serving as an associate editor for the ACM Transactions on
    Mathematical Software and as a member of the SIGSAM Nominating
    Committees two years ago and again this year.",
  paper = "Grie79.pdf",
  keywwords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Grun94,
  author = "Gruntz, Dominik and Monagan, Michael B.",
  title = {{Introduction to Gauss}},
  journal = "SIGSAM Bulletin",
  volume = "28",
  number = "3",
  pages = "3-19",
  year = "1994",
  link = "\url{http://ftp.cecm.sfu.ca/personal/monaganm/papers/Gauss.pdf}",
  abstract =
    "The Gauss package offers Maple users a new approach to programming
    based on the idea of parameterized types (domains) which is central to
    the AXIOM system. This approach to programming is now regarded by many
    as the right way to go in computer algebra systems design. In this
    article, we describe how Gauss is designed and show examples of usage.
    We end with some comments about how Gauss is being used in Maple.",
  paper = "Grun94.pdf",
  keywords = "axiomref",
  beebe = "Gruntz:1994:IG"
}
```

———-

— **Gruntz:1994:IG** —

```
@Article{Gruntz:1994:IG,
  author =       "D. Gruntz and M. Monagan",
  title =        {{Introduction to Gauss}},
  journal =      j-SIGSAM,
  volume =       "28",
  number =       "2",
  pages =        "3--19",
  month =        aug,
  year =         "1994",
  CODEN =        "SIGSBZ",
  ISSN =         "0163-5824 (print), 1557-9492 (electronic)",
  ISSN-L =       "0163-5824",
  bibdate =      "Tue Dec 12 09:33:35 MST 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The Gauss package offers Maple users a new approach to
                 programming based on the idea of parameterized types
                 (domains) which is central to the AXIOM system. This
                 approach to programming is now regarded by many as the
                 right way to go in computer algebra systems design. We
                 describe how Gauss is designed and show examples of
                 usage. We end with some comments about how Gauss is
                 being used in Maple.",
  acknowledgement = ack-nhfb,
  affiliation =  "Inst. for Sci. Comput., Eidgenossische Tech.
```

```
                Hochschule, Zurich, Switzerland",
  classification = "C6110 (Systems analysis and programming); C6130
                (Data handling techniques); C7310 (Mathematics
                computing)",
  fjournal =    "SIGSAM Bulletin",
  keywords =    "AXIOM system; Computer algebra systems design; Gauss
                package; Maple users; Parameterized types;
                Programming",
  language =    "English",
  pubcountry =  "USA",
  thesaurus =   "Programming environments; Software packages; Symbol
                manipulation; Systems analysis; Type theory",
}
```

———-

— **axiom.bib** —

```
@phdthesis{Grun96,
  author = "Gruntz, Dominik",
  title = {{On Computing Limits in a Symbolic Manipulation System}},
  school = "Swiss Federal Institute of Technology Zurich",
  year = "1996",
  link = "\url{http://www.cybertester.com/data/gruntz.pdf}",
  abstract = "
    This thesis presents an algorithm for computing (one-sided) limits
    within a symbolic manipulation system. Computing limtis is an
    important facility, as limits are used both by other functions such as
    the definite integrator and to get directly some qualitative
    information about a given function.

    The algorithm we present is very compact, easy to understand and easy
    to implement. It overcomes the cancellation problem other algorithms
    suffer from. These goals were achieved using a uniform method, namely
    by expanding the whole function into a series in terms of its most
    rapidly varying subexpression instead of a recursive bottom up
    expansion of the function. In the latter approach exact error terms
    have to be kept with each approximation in order to resolve the
    cancellation problem, and this may lead to an intermediate expression
    swell. Our algorithm avoids this problem and is thus suited to be
    implemented in a symbolic manipulation system.",
  paper = "Grun96.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Gute16,
  author = "Gutenberg Self-Publishing Press",
  title = {{OpenAxiom}},
```

```
  link = "\url{http://self.gutenberg.org/articles/openaxiom}",
  year = "2016",
  keywords = "axiomref"
}
```

———-

1.40.8 H

— **axiom.bib** —

```
@article{Hall96,
  author = "Hall, Cordelia V. and Hammond, Kevin and Jones, Simon L. Peyton
            and Wadler, Philip L.",
  title = {{Type Classes in Haskell}},
  journal = "Trans. on Programming Langues and Systems",
  volume = "18",
  number = "2",
  pages = "109-138",
  year = "1996",
  abstract =
    "This article de nes a set of type inference rules for resolving
    overloading introduced by type classes, as used in the functional
    programming language Haskell.  Programs including type classes are
    transformed into ones which may be typed by standard Hindley-Milner
    inference rules. In contrast to other work on type classes, the rules
    presented here relate directly to Haskell programs.  An innovative
    aspect of this work is the use of second-order lambda calculus to
    record type information in the transformed program.",
  paper = "Hall96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Harr98,
  author = "Harrison, J. and Thery, L.",
  title = {{A Skeptic's approach to combining HOL and Maple}},
  journal = "J. Autom. Reasoning",
  volume = "21",
  number = "3",
  pages = "279-294",
  year = "1998",
  link = "\url{http://www.cl.cam.ac.uk/~jrh13/papers/cas.ps.gz}",
  abstract =
    "We contrast theorem provers and computer algebra systems, pointing
    out the advantages and disadvantages of each, and suggest a simple way
    to achieve a synthesis of some of the best features of both. Our
    method is based on the systematic separation of search for a solution
```

```
    and checking the solution, using a physical connection between
    systems. We describe the separation of proof search and checking in
    some detail, relating it to proof planning and to the complexity class
    NP, and discuss different ways of exploiting a physical link between
    systems. Finally, the method is illustrated by some concrete examples
    of computer algebra results proved formally in the HOL theorem prover
    with the aid of Maple.",
  paper = "Harr98.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Harr12,
  author = "Harriss, Edmund and Daly, Timothy",
  title = {{Have we ever lost mathematics?}},
  link = "\url{https://maxwelldemon.com/2012/05/09/have-we-ever-lost-mathematics/}",
  year = "2012",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Hassner 87]{HBW87} Hassner, Martin; Burge,  William H.;
Watt,  Stephen M.
  title = {{Construction of Algebraic Error Control Codes (ECC) on the
            Elliptic Riemann Surface}},
in [Wit87], pp5-8
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Kead93a,
  author = "Keady, G. and Richardson, M.G.",
  title = {{An application of IRENA to systems of nonlinear equations arising
           in equilibrium flows in networks}},
  booktitle = "Proc. ISSAC 1993",
  series = "ISSAC '93",
  year = "1993",
  abstract =
    "IRENA - an $I$nterface from $RE$DUCE to $NA$G - runs under the REDUCE
    Computer Algebra (CA) system and provides an interactive front end to
    the NAG Fortran Library.

    Here IRENA is tested on a problem closer to an engineering problem
```

```
    than previously publised examples. We also illustrate the use of the
    {\tt codeonly} switch, which is relevant to larger scale problems. We
    describe progress on an issue raised in the 'Future Developments'
    section in our {\sl SIGSAM Bulletin} article [2]: the progress improves
    the practical effectiveness of IRENA.",
  paper = "Kead93a.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@inproceedings{Hawk95,
  author = "Hawkes, Evatt and Keady, Grant",
  title = {{Two more links to NAG numerics involving CA systems}},
  booktitle = "IMACS Applied Computer Algebra Conference",
  location = "University of New Mexico",
  year = "1995",
  algebra =
   "\newline\refto{domain ASP1 Asp1}
    \newline\refto{domain ASP10 Asp10}
    \newline\refto{domain ASP12 Asp12}
    \newline\refto{domain ASP19 Asp19}
    \newline\refto{domain ASP20 Asp20}
    \newline\refto{domain ASP24 Asp24}
    \newline\refto{domain ASP27 Asp27}
    \newline\refto{domain ASP28 Asp28}
    \newline\refto{domain ASP29 Asp29}
    \newline\refto{domain ASP30 Asp30}
    \newline\refto{domain ASP31 Asp31}
    \newline\refto{domain ASP33 Asp33}
    \newline\refto{domain ASP34 Asp34}
    \newline\refto{domain ASP35 Asp35}
    \newline\refto{domain ASP4 Asp4}
    \newline\refto{domain ASP41 Asp41}
    \newline\refto{domain ASP42 Asp42}
    \newline\refto{domain ASP49 Asp49}
    \newline\refto{domain ASP50 Asp50}
    \newline\refto{domain ASP55 Asp55}
    \newline\refto{domain ASP6 Asp6}
    \newline\refto{domain ASP7 Asp7}
    \newline\refto{domain ASP73 Asp73}
    \newline\refto{domain ASP74 Asp74}
    \newline\refto{domain ASP77 Asp77}
    \newline\refto{domain ASP78 Asp78}
    \newline\refto{domain ASP8 Asp8}
    \newline\refto{domain ASP80 Asp80}
    \newline\refto{domain ASP9 Asp9}",
  abstract =
    "The 'more' in the title is because this paper is a sequel to papers
    by Keving Broughan, [BKRRD,BK]. For some years GK has had interests in
    (i) interactive front-ends to numeric computation, such as the
```

```
    NAG/IMSL library computation, and (ii) Fortran code generation for
    Argument SubPrograms (ASPs), such as those neede by some NAG/IMSL
    routines. Demonstrations of three links to the NAG library are
    described in [BKRRD]. A description of a link to NAG from Macsyma
    which was mentioned, but not in a sufficiently advanced state to
    demonstrate in early 1991, is given in [BK]. The situation at the end
    of 1991 was that there were links to NAG involving each of Macsyma,
    REDUCE and Mathematica. The links are called Naglink, IRENA and
    InterCall, respectively. The principal authors of IRENA are Mike Dewar
    and Mike Richardson. InterCall is not specific to the NAG library;
    indeed InterCall is used with calls to IMSL and to elsewhere at the
    conference venue, the University of New Mexico.

    The two futher links to NAG library treated in this paper are AXIOM2.0
    and genmex/ESC, genmex allows calls to NAG from Matlab. genmex can be
    regarded as similar to InterCall: genmes uses Matlab's mex files in a
    similar way to InterCall's use of Mathematica's MathLink. Again genmex
    is not specific to the NAG library. Mike Dewar is an author both of
    IRENA and the AXIOM2.0 link to the NAG library: see [D] foe discussion
    of the differences between the IRENA project and the AXIOM-NAG link
    project.",
  paper = "Hawk95.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@inproceedings{Hear80,
  author = "Hearn, Anthony C.",
  title = {{Symbolic Computation and its Application to High Energy Physics}},
  booktitle = "Proc. 1980 CERN School of Computing",
  pages = "390-406",
  year = "1980",
  link = "\url{http://www.iaea.org/inis/collection/NCLCollectionStore/\_Public/12/631/12631585.pdf}",
  abstract =
    "It is clear that we are in the middle of an electronic revolution
    whose effect will be as profound as the in dustrial revolution.  The
    continuing advances in computing technology will provide us with
    devices which will make present day computers appear primitive.  In
    this environment, the algebraic and other non-numerical capabilities
    of such devices will become increasingly important. These lectures
    will review the present state of the field of algebraic computation
    and its potential for problem solving in high energy physics and
    related areas.  We shall begin with a brief description of the
    available systems and examine the data objects which they consider.
    As an example of the facilities which these systems can offer, we
    shall then consider the problem of analytic integration, since this
    is so fundamental to many of the calculational techniques used by high
    energy physicists.  Finally, we shall study the implications which the
    current developments in hardware technology hold for scientific
    problem solving.",
```

```
  paper = "Hear80.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Hear82,
  author = "Hearn, Anthony C.",
  title = {{REDUCE - A Case Study in Algebra System Development}},
  journal = "Lecture Notes in Computer Science",
  volume = "144",
  pages = "263-272",
  year = "1982",
  paper = "Hear82.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Hear95,
  author = "Hearn, Anthony C. and Eberhard, Schrufer",
  title = {{A computer algebra system based on order-sorted algebra}},
  journal = "J. Symbolic Computing",
  volume = "19",
  number = "1-3",
  pages = "65-77",
  year = "1995",
  abstract =
    "This paper presents the prototype design of an algebraic computation
    system that manipulates algebraic quantities as generic objects using
    order-sorted algebra as the underlying model. The resulting programs
    have a form that is closely related to the algorithmic description of
    a problem, but with the security of full type checking in a compact,
    natural style.",
  paper = "Hear95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Heck93,
  author = "Heck, Andre",
  title = {{Introduction to Maple}},
  year = "1993",
  publisher = "Springer-Verlag",
  abstract =
```

```
    "This is an introductory book on one of the most powerful computer
    algebra systems, viz, Maple: The primary emphasis in this book is on
    learning those things that can be done with Maple and how it can be
    used to solve mathematical problems. In this book usage of Maple as a
    programming language is not discussed at a higher level than that of
    defining simple procedures and using simple language constructs.
    However, the Maple data structures are discussed in detail.

    This book is divided into eighteen chapters spanning a variety of
    topics. Starting with an introduction to symbolic computation and
    other similar computer algebra systems, this book covers several
    topics like polynomials and rational functions, series,
    differentiation and integration, differential equations, linear
    algebra, 2-D and 3-D graphics, etc. The applications covered include
    kinematics of the Stanford manipulator, a 3-component model for
    cadmium transfer through the human body, molecular-orbital Hckel
    theory, prolate spheroidal coordinates and Moore-Penrose inverses.

    At the end of each chapter, a good number of excercises is given. A
    list of relevant references is also given at the end of the book.
    This book is very useful to all users of Maple package.",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Heck 01]{Hec01} Heck, A.
  title = {{Variables in computer algebra, mathematics and science}},
The International Journal of Computer Algebra in Mathematics Education
Vol. 8 No. 3 pp195-210 (2001)
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@phdthesis{Hemm03,
  author = "Hemmecke, Ralf",
  title = {{Involutive Bases for Polynomial Ideals}},
  school = "Johannes Kepler University, RISC",
  year = "2003",
  abstract =
    "This thesis contributes to the theory of polynomial involutive
    bases. Firstly, we present the two existing theories of involutive
    divisions, compare them, and come up with a generalised approach of
    {\sl suitable partial divisions}. The thesis is built on this
    generalized approach. Secondly, we treat the question of choosing a
    ``good'' suitable partial division in each iteration of the involutive
    basis algorithm. We devise an efficient and flexible algorithm for
    this purpose, the {\sl Sliced Division} algorithm. During the
```

```
    involutive basis algorithm, the Sliced Division algorithm contributes
    to an early detection of the involutive basis property and a
    minimisation of the number of critical elements. Thirdly, we give new
    criteria to avoid unnecessary reductions in an involutive basis
    algorithm. We show that the termination property of an involutive
    basis algorithm which applies our criteria is independent of the
    prolongation selection strategy used during its run. Finally, we
    present an implementation of the algorithm and results of this thesis
    in our software package CALIX."
}
```

———-

— **axiom.bib** —

```
@misc{RISC06,
  author = "Hemmecke, Ralf and Rubey, Martin",
  title = {{AXIOM Workshop 2006}},
  link = "\url{http://axiom-wiki.newsynthesis.org/WorkShopRISC2006}",
  year = "2006",
  location = "Hagenberg, Austria",
  abstract =
    "Axiom is a computer algebra system with a long tradition. It recently
    became free software.

    The workshop aims at a cooperation of Axiom developers with developers
    of packages written for other Computer Algebra Systems or developers
    of stand-alone packages. Furthermore, the workshop wants to make the
    potential of Axiom and Aldor more widely known in order to attract new
    users and new developers.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{RISC07,
  author = "Hemmecke, Ralf and Rubey, Martin",
  title = {{AXIOM Workshop 2007}},
  link = "\url{http://axiom-wiki.newsynthesis.org/WorkShopRISC2007}",
  year = "2007",
  location = "Hagenberg, Austria",
  abstract =
    "The workshop aims at a cooperation of Axiom developers with deelopers
    of packages written for other Computer Algebra Systems, and
    mathematicians that would like to use a computer algebra system to
    perform experiments.

    One goal of the workshop is to learn about the mathematical theory,
    the design of packages written for other CAS and to make those
    functionalities available in Axiom." ,
```

```
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Hera16,
  author = "Heras, Jonathan and Martin-Mateos, Franciso Jesus and
            Pascual, Vico",
  title = {{A Hierarchy of Mathematical Structures in ACL2}},
  link = "\url{http://staff.computing.dundee.ac.uk/jheras/papers/ahomsia.pdf}",
  abstract =
    "In this paper, we present a methodology which allows one to deal with
    {\sl mathematical structures} in the ACL2 theorem prover. Namely, we
    cope with the representation of mathematical structures, the
    certification that an object fulfills the axioms characterizing an
    algebraic structure and the generation of generic theories about
    concrete structures. As a by-product, an {\sl ACL2 algebraic
    hierarchy} has been obtained. Our framework has been tested with the
    definition of {\sl homology groups}, an example coming from
    Homological Algebra which involves several notions related to
    Universal Algebra. The method presented here, when compared to a
    from-scratch approach, is preferred when working with complex
    mathematical structures; for instance, the ones coming from Algebraic
    Topology. The final aim of this work is the verification of Computer
    Algebra systems, a field where our hierarchy fits better than the ones
    developed in other systems.",
  paper = "Hera16.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Hera15,
  author = "Heras, Jonathan and Martin-Mateos, Franciso Jesus and
            Pascual, Vico",
  title = {{Modelling algebraic structures and morphisms in ACL2}},
  journal = "Appl. Algebra Eng. Commun. Comput.",
  volume = "26",
  number = "3",
  pages = "277-303",
  year = "2015",
  abstract =
    "In this paper, we present how algebraic structures and morphisms can
    be modelled in the ACL2 theorem prover. Namely, we illustrate a
    methodology for implementing a set of tools that facilitates the
    formalisations related to algebraic structures -- as a result, an
    algebraic hierarchy ranging from setoids to vector spaces has been
    developed. The resultant tools can be used to simplify the development
```

```
    of generic theories about algebraic structures. In particular, the
    benefits of using the tools presented in this paper, compared to a
    from-scratch approach, are especially relevant when working with
    complex mathematical structures; for example, the structures employed
    in Algebraic Topology. This work shows that ACL2 can be a suitable
    tool for formalising algebraic concepts coming, for instance, from
    computer algebra systems.",
  paper = "Hera15.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Here96,
  author = "Hereman, Willy",
  title = {{The Incredible World of Symbolic Mathematics
            A Review of Computer Algebra Systems}},
  year = "1996",
  link = "\url{https://inside.mines.edu/~whereman/papers/Hereman-PhysicsWorld-9-March1996.pdf}",
  paper = "Here96.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Here97,
  author = "Hereman, Willy",
  title = {{Review of Symbolic Software for Lie Symmetry Analysis}},
  journal = "Math. Comput. Modelling",
  volume = "25",
  number = "8/9",
  pages = "115-132",
  year = "1997",
  abstract =
    "Sophus Lie (1842-1899) pioneered the study of continuous
    transformation groups that leave systems of differential equations
    invariant.  Lies work [1-3] brought diverse and ad hoc integration
    methods for solving special classes of differential equations under a
    common conceptual umbrella.  Indeed, Lies infinitesimal
    transformation method provides a widely applicable technique to find
    closed form solutions of ordinary differential equations (ODES).
    Standard solution methods for first-order or linear ODES can be
    characterized in terms of symmetries.  Through the group
    classification of ODES, Lie succeeded in identifying all ODES that can
    either be reduced to lower-order ones or be completely integrated via
    group theoretic techniques.

    Applied to partial differential equations (PDEs), Lies method [2]
```

```
   leads to group-invariant solutions and conservation laws.  Exploiting
   the symmetries of PDEs, new solutions can be derived from known ones,
   and PDEs can be classified into equivalence classes.  Furthermore,
   group-invariant solutions obtained via Lies approach may provide
   insight into the physical models themselves, and explicit solutions
   can serve as benchmarks in the design, accuracy testing, and
   comparison of numerical algorithms.

   Nowadays, the concept of symmetry plays a key role in the study and
   development of mathematics and physics.  Indeed, the theory of Lie
   groups and Lie algebras is applied to diverse fields of mathematics
   including differential geometry, algebraic topology, bifurcation
   theory, to name a few.  Lies original ideas greatly influenced the
   study of physically important systems of differential equations in
   classical and quantum mechanics, fluid dynamics, elasticity, and many
   other applied areas [4-81].

   The application of Lie group methods to concrete physical systems
   involves tedious computations.  Even the calculation of the
   continuous symmetry group of a modest system of differential equations
   is prone to errors, if done with pencil and paper.  Computer algebra
   systems (CAS) such as Mathematica, MACSYMA, Maple, REDUCE, AXIOM and
   MuPAD are extremely useful for such computations.  Symbolic packages
   [9-11], written in the language of these GAS, can find the determining
   equations of the Lie symmetry group.  The most sophisticated packages
   then reduce these into an equivalent but more suitable system,
   subsequently solve that system in closed form, and go on to calculate
   the infinitesimal generators that span the Lie algebra of symmetries.

   In Section 2, we discuss methods and algorithms used in the
   computation of Lie symmetries.  We address the computation of
   determining systems, their reduction to standard form, solution
   techniques, and the computation of the size of the symmetry group.
   In Section 3, we look beyond Lie-point symmetries, addressing contact
   and generalized symmetries, as well as nonclassical or conditional
   symmetries.

   Section 4 is devoted to a review of modern Lie symmetry
   programs, classified according to the underlying CAS.  The review
   focuses on Lie symmetry software for classical Lie-point symmetries,
   contact (or dynamical), generalized (or Lie-Backlund) symmetries,
   nonclassical (or conditional) symmetries.  Most of these packages were
   written in the last decade.  Researchers interested in details about
   pioneering work should consult [9,10,12].  In Section 5, two examples
   illustrate results that can be obtained with Lie symmetry software.
   In Section 6 we draw some conclusions.

   Lack of space forces us to give only a few key references for the Lie
   symmetry packages.  A comprehensive survey of the literature devoted
   to theoretical as well as computational aspects of Lie symmetries,
   with over 300 references, can be found elsewhere [11].",
  paper = "Here97.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Hive04,
  author = "Hivert, Florent and Thiery, Nicolas M.",
  title = {{MuPAD-Combinat, an open-source package for research in
           algebraic combinatorics}},
  journal = "Seminaire Lotharingien de Combinatoire",
  volume = "51",
  number = "B51z",
  year = "2004",
  link = "\url{http://www.emis.de/journals/SLC/wpapers/s51thiery.pdf}",
  abstract =
    "In this article we give an overview of the MuPAD-Combinat open-source
    algebraic combinatorics package for the computer algebra system MuPAD
    2.0.0 and higher. This includes our motivations for developing yet
    another combinatorial software, a tutorial introduction with lots of
    examples, as well as notes on the general design. The material
    presented here is also available as a part of the MuPAD-Combinat
    handbook; further details and references on the algorithms used can be
    found there. The package and the handbook are available from the web
    page, together with download and installation instructions, mailing
    lists, etc.",
  paper = "Hive04.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Hoan00,
  author = "Hoang, Ngoc Minh and Petitot, Michel and Van er Hoeven, Joris",
  title = {{Shuffle algebra and polylogarithms}},
  journal = "Discrete Math.",
  volume = "225",
  number = "1-3",
  pages = "217-230",
  year = "2000",
  abstract =
    "Generalized polylogarithms are defined as iterated integrals with
    respect to the two differential forms $\omega_0=\frac{dz}{z}$ and
    $\omega_1=\frac{dz}{(1-z)}$. We give an algorithm which computes the
    monodromy of these special functions. This algorithm, implemented in
    AXIOM, is based on the computation of the associator $\Phi_{KZ}$ of
    Drinfeld, in factorized form. The monodromy formulae involve special
    constants, called multiple zeta values. We prove that the algebra of
    polylogarithms is isomorphic to a shuffle algebra.",
  paper = "Hoan00.pdf",
  keywords = "axiomref"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Hoar08,
  author = "Hoarau, Emma and David, Claire",
  title = {{Lie group computation of finite difference schemes}},
  year = "2008",
  journal = "math.NA",
  link = "\url{http://arxiv.org/pdf/math/0611895.pdf}",
  abstract =
    "A Mathematica based program has been elaborated in order to determine
    the symmetry group of a finite difference equation. The package
    provides functions which enabel use to solve the determining equations
    of the related Lie group.",
  paper = "Hoar08.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Hodo11,
  author = "Hodorog, Madalina",
  title = {{Symbolic-Numeric Algorithms for Plane Algebraic Curves}},
  year = "2011",
  school = "RISC Research Institute for Symbolic Computation",
  abstract =
    "In computer algebra, the problem of computing topological invariants
    (i.e. delta-invariant, genus) of a plan complex algebraic curve is
    well-understood if the coefficients of the defining polynomial of the
    curve are exact data (i.e. integer numbers or rational numbers). The
    challenge is to handle this problem if the coefficients are inexact
    (i.e. numerical values).

    In this thesis, we approach the algebraic problem of computing
    invariants of a plane complex algebraic curve defined by a polynomial
    with both exact and inexact data. For the inexact data, we associate a
    positive real number called {\sl tolerance} or {\sl noise}, which
    measures the error level in the coefficients. We deal with an {\sl
    ill-posed} problem in the sense that, tiny changes in the input data
    lead to dramatic modifications in the output solution.

    For handling the ill-posedness of the problem we present a {\sl
    regularization} method, which estimates the invariants of a plane
    complex algebraic curve. Our regularization method consists of a set
    of {\sl symbolic-numeric algorithms} that extract structural
    information on the input curve, and of a {\sl parameter choice rule},
    i.e. a function in the noise level. We first design the following
```

```
    symbolic-numeric algorithms for computing the invariants of a plane
    complex algebraic curve:
    \begin{itemize}
    \item we compute the link of each singularity of the curve by numerical
    equation solving
    \item we compute the Alexander polynomial of each link by using
    algorithms from computational geometry (i.e. an adapted version of
    the Bentley-Ottmann algorithm) and combinatorial objects from knot
    theory.
    \item we derive a formula for the delta-invariant and for the genus
    \end{itemize}

    We then prove that the symbolic-numeric algorithms together with the
    parameter choice rule compute approximate solutions, which satisfy the
    {\sl convergence for noisy data property}. Moreover, we perform
    several numerical experiments, which support the validity for the
    convergence statement.

    We implement the designed symbolic-numeric algorithms in a new
    software package called {\sl Genom3ck}, developed using the {\sl Axel}
    free algebraic modeler and the {\sl Mathemagix} free computer algebra
    system. For our purpose, both of these systems provide modern
    graphical capabilities, and algebraic and geometric tools for
    manipulating algebraic curves and surfaces defined by polynomials with
    both exact and inexact data. Together with its main functionality to
    compute the genus, the package {\sl Genom3ck} computes also other type
    of information on a plane complex algebraic curve, such as the
    singularities of the curve in the projective plane and the topological
    type of each singularity.",
  paper = "Hodo11.pdf"
}
```

———-

— axiom.bib —

```
@misc{Hoep95,
  author = "Hoeppner, Sabine",
  title = {{Linear differential equations of second order in fields of
           positive characteristic}},
  comment = "Essen: Univ. Essen, FB Math 71 S.",
  year = "1995",
  abstract =
    "Let be a differential field of characteristic $p>2$ of the type
    $(\mathbb{F}_p(x),\frac{d}{dx})$. The equation studied is
    $y^{\prime\prime}+ay^{\prime}+by=0$ with $a,b \in K$.  The goal is to
    produce a Liouvillian extension $L \supset K$ which contains two
    independent solutions. This is done by solving the associated Riccati
    equation $u^{\prime}+u^2+au+b=0$. Unlike the characteristic zero
    situation, the Riccati equation has one or two solutions in an
    algebraic extension of degree $\le 2$ of $K$.  Full solutions are
    given for the various cases that do occur. The paper ends with a
    program in AXIOM which computes the solutions of the Riccati equation
```

```
    and the Liouvillian extensions.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Hoev13,
  author = "Hoeven, Joris van der and Lecerf, Gregoire",
  title = {{Interfacing Mathemagix with C++}},
  link = "\url{http://www.texmacs.org/joris/mmxcpp/mmxcpp.pdf}",
  abstract =
    "In this paper, we give a detailed description of the interface
    between the Mathemagix language and C++. In particular, we describe
    the mechanism which allows us to import a C++ template library
    (which only permits static instantiation) as a fully generic
    Mathemagix template library.",
  paper = "Hoev13.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Hoev15,
  author = "Hoeven, Joris van der and Lecerf, Gregoire",
  title = {{Interfacing Mathemagix with C++}},
  link = "\url{http://www.texmacs.org/joris/mmxcpp/mmxcpp.html}",
  abstract =
    "In this paper, we give a detailed description of the interface
    between the Mathemagix language and C++. In particular, we describe
    the mechanism which allows us to import a C++ template library
    (which only permits static instantiation) as a fully generic
    Mathemagix template library.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Hold11,
  author = "Hohold, Tom and van Lint, Jacobus H. and Pellikaan, Ruud",
  title = {{Algebraic Geometry Codes}},
  year = "2011",
  pages = "871-961",
  booktitle = "Handbook of Coding Theory",
  publisher = "V.S Pless, W.C. Huffman and R.A. Brualdi (eds)",
  volume = "I",
```

```
  link = "\url{http://www.win.tue.nl/~ruudp/paper/31.pdf}",
  algebra = "\newline\refto{category PRSPCAT ProjectiveSpaceCategory}",
  isbn = "9780444814722",
  paper = "Hold11.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Huguet 89]{HP89} Huguet, L.; Poli, A.  (eds).
Applied Algebra, Algebraic Algorithms and Error-Correcting Codes.
5th International Conference AAECC-5 Proceedings.
Springer-Verlag, Berlin, Germany / Heildelberg, Germany / London, UK / etc.,
1989. ISBN 3-540-51082-6. LCCN QA268.A35 1987
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@book{Hous92,
  author = "Houstis, E.N. and Gaffney, P.W.",
  title = {{Programming environments for high-level scientific problem
           solving}},
  year = "1992",
  publisher = "Elsevier",
  isbn = "978-0444891761",
  abstract =
    "Programming environments, as the name suggests, are intended to
    provide a unified, extensive range of capabilities for a person
    wishing to solve a problem using a computer. In this particular
    proceedings volume, the problem considered is a high-level scientific
    computation. In other words, a scientific problem whose solution
    usually requires sophisticated computing techniques and a large
    allocation of computing resources.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Hous00,
  author = "Houstis, Elias N. and Rice, John R.",
  title = {{Future problem solving environments for computational science}},
  journal = "Math. Comput. Simul.",
  volume = "54",
  number = "4-5",
  pages = "243-257",
  year = "2000",
```

```
  abstract =
    "We review the current state of the Problem Solving Environment (PSE)
    field and make projections for the future. First, we describe the
    computing context, the definition of a PSE and the goals of a PSE. The
    state-of-the-art is summarized along with the principal components and
    paradigms for building PSEs. The discussion of the future is given in
    three parts: future trends, scenarios for 2010/2025, and research
    issues to be addressed.",
  paper = "Hous00.pdf",
  keywords = "axiomref"
}
```

———-

1.40.9 I

— **axiom.bib** —

```
@article{Igar75,
  author = "Igarashi, Shigeru and London, Ralph L. and Luckham, David C.",
  title = {{Automatic Program Verification I: A Logical Basis and Its
            Implementation}},
  journal = "Acta Informatica",
  volume = "4",
  number = "2",
  pages = "145-182",
  year = "1975",
  abstract =
    "Defining the semantics of programming languages by axioms and rules
    of inference yields a deduction system within which proofs may be
    given that programs satisfy specifications. The deduction system
    herein is shown to be consistent and also deduction complete with
    respect to Hoare's system. A subgoaler for the deduction system is
    described whose input is a significant subset of Pascal programs plus
    inductive assertions. The output is a set of verification conditions
    or lemmas to be proved. Several non-trivial arithmetic and sorting
    programs have been shown to satisfy specifications by using an
    interactive theorem prover to automatically generate proofs of the
    verification conditions. Additional components for a more powerful
    verification system are under construction.",
  paper = "Igar75.pdf"
}
```

———-

1.40.10 J

— **ignore** —

```
\bibitem[Jacob 93]{JOS93} Jacob, G.; Oussous, N. E.; Steinberg, S. (eds)
```

```
Proceedings SC 93
International IMACS Symposium on Symbolic Computation. New Trends and
Developments. LIFL Univ. Lille, Lille France, 1993
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Jacq97,
  author = "Jacquemard, Alain and Khechichine-Mourtada, F.Z. and Mourtada, A.",
  title = {{Formal algorithms applied to the study of the cyclicity of a
           generic algebraic polycycle with four hyperbolic crests}},
  journal = "Nonlinearity",
  volume = "10",
  number = "1",
  pages = "19-53",
  year = "1997",
  comment = "french",
  abstract =
    "Drawing on the work of Mourtada, we show that a family of vector
    fields with a generic algebraic polycycle of four hyperbolic apices
    possesses a maximum capacity of four limit cycles. This cyclicity is
    attained in an opening connecting the parameters which the edge
    contains, in particular a generic line of singularities of dovetail
    type. We also give an asymptotic estimation of the volume of this
    opening, as well as an explicit example of a family of polynomial
    vector fields replicating the above-described conditions and
    possessing five limit cycles. The methods employed are very diverse:
    geometrical arguments (Thoms theory of catastrophes and the theory of
    algebraic singularities), developments from Puiseux, the number of
    major roots by Descartes law and calculated exactly by Sturm series,
    and other specific methods for formal calculus, such as for example
    the cylindrical algebraic decomposition and the resolution of
    algebraic systems via the construction of Grbner bases. The
    calculations have been executed formally, that is to say without
    making the least appeal to numerical approximation, in using the
    formal calculus system AXIOM.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Jacq02,
  author = "Jacquemard, Alain and Teixeira, M.A.",
  title = {{Effective algebraic geometry and normal forms of reversible
           mappings}},
  journal = "Rev. Mat. Complut.",
  volume = "15",
  number = "1",
```

```
  pages = "31-55",
  year = "2002",
  abstract =
    "The authors consider a problem coming from the theory of discrete
    dynamical systems (iterations of diffeomorphisms in
    $\mathbb{R}^3$). Namely, how to characterize the behaviour of the
    trajectories near the fixed points and the stability of this behaviour
    under perturbations. The authors use the computer in the context of
    reversible mappings. When such a mapping has some degree of degeneracy
    they apply Grbner bases techniques and the theory of (formal) normal
    forms in the space of the coefficients of the jets of the
    mapping. They show that a language with typed objects like AXIOM is
    very convenient to solve such problems.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Jaes93,
  author = "Jaeschke, Gerhard",
  title = {{On Strong Pseudoprimes to Several Bases}},
  journal = "Mathematics of Computation",
  volume = "61",
  number = "204",
  year = "1993",
  pages = "915-926",
  algebra = "\newline\refto{package PRIMES IntegerPrimesPackage}",
  abstract =
    "With $\psi_k$ denoting the smallest strong pseudoprime to all of the
    first $k$ primes taken as bases we determine the exact values for
    $\psi_5$, $\psi_6$, $\psi_7$, $\psi_8$, and give upper bounds for
    $\psi_9$, $\psi_{10}$, $\psi_{11}$. We discuss the methods and
    underlying facts for obtaining these results",
  paper = "Jaes93.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Jage96,
  author = "Jager, Bram De and van Asch, Bram",
  title = {{Symbolic Solutions for a Class of Partial Differential Equations}},
  journal = "J. Symbolic Computation",
  volume = "22",
  year = "1996",
  pages = "459-468",
  link = "\url{http://www.mate.tue.nl/mate/pdfs/1610.pdf}",
  abstract =
```

```
    "An algorithm to generate solutions for members of a class of
    completely integrable partial differential equations has been derived
    from a constructive proof of Frobenius' Theorem. The algorithm is
    implemented as a procedure in the computer algebra system
    Maple. Because the implementation uses the facilities of Maple for
    solving sets of ordinary differential equations and for sets of
    nonlinear equations, and those facilities are limited, the problems
    that actually can be solved are restricted in size and
    complexity. Several examples, some derived from industrial practice,
    are presented to illustrate the use of the algorithm and to
    demonstrate the advantages and shortcomings of the implementation.",
  paper = "Jage96.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Janssen 88]{Jan88} Jan{\ss}en, R. (ed)
Trends in Computer Algebra, International Symposium
Bad Neuenahr, May 19-21, 1987, Proceedings, volume 296 of Lecture Notes in
Computer Science.
Springer-Verlag, Berlin, Germany / Heildelberg, Germany / London, UK / etc.,
1988 ISBN 3-540-18928-9, 0-387-18928-9 LCCN QA155.7.E4T74 1988
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@techreport{Jenk70,
  author = "Jenks, Richard D.",
  title = {{META/LISP: An interactive translator writing system}},
  type = "research report",
  number = "RC2968",
  year = "1970",
  institution = "IBM Research",
  abstract =
    "META/LISP is a general purpose translator writing system for IBM
    System/360 currently running on TSS, CP/CMS, and OS/360. The input
    to the system is a source program which simultaneously describes
    1) the syntax of some input data to be translated and
    2) algorithms which operate on the input data and a pushdown stack
    to accomplish the desired translation; the output of the system is
    a compiled program for translating that input data. In particular
    when the input data are statements of a higher-level language to
    be translated into assembly language, META/LISP serves as a
    compiler-compiler. META/LISP uses the top-down syntax-directed
    approach which makes the system extremely attractive for the
    design and implementation of experimental languages; using
    META/LISP such compilers are easy to write, easy to check out, and
```

```
    - most importantly - easy to modify interactively. The appendices
    which follow a rather complete description of the system including
    a self-description of the META/LISP compiler.",
  paper = "Jenk70.pdf",
  keywords = "axiomref, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Jenk71a,
  author = "Jenks, Richard D.",
  title = {{META LISP and META PLUS:: Tools for Rapidly Implementing
            extendable language translators}},
  booktitle = "Proc. 2nd ACM Symposium on Symbolic and Algebraic
               Manipulation",
  publisher = "ACM",
  pages = "281",
  year = "1971",
  abstract =
    "A unique feature of the SCRATCHPAD system for symbolic
    manipulation is its powerful translation facilities. The essential
    components are META/LISP, a translator writing system, and
    META/PLUS, a facility for immediately extending the syntax of any
    translator produced through META/LISP. This talk will illustrate
    how these facilities may be used to produce a conversational
    higher-level LISP system. The language chosen is called ALPL
    because of its similarity to APL and LPL, a language resident in
    the SCRATCHPAD system. The essential characteristics of ALPL are
    described by eight syntax rules together with brief examples of
    corresponding ALPL and LISP programs. It is shown how a META/LISP
    program may be easily written to produce a conversation ALPL
    system. The ALPL language is then incrementally extended by
    introducing new notations defined in terms of existing ALPL
    constructs through calls to META/PLUS.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techReport{Jenk71,
  author = "Jenks, Richard D.",
  title = {{META/PLUS: The syntax extension facility for SCRATCHPAD}},
  type = "Research Report",
  number = "RC 3259",
  institution = "IBM Research",
  year = "1971",
  keywords = "axiomref",
  beebe = "Jenks:1971:MPS"
```

```
}
```

———-

— **Jenks:1971:MPS** —

```
@TechReport{Jenks:1971:MPS,
  author =       "R. D. Jenks",
  title =        {{META\slash PLUS: The Syntax Extension Facility for
                 SCRATCHPAD}}",
  type =         "Research Report",
  number =       "RC 3259",
  institution =  "International Business Machines Inc., Thomas J. Watson
                 Research Center",
  address =      "Yorktown Heights, NY, USA",
  pages =        "??",
  month =        feb,
  year =         "1971",
  bibdate =      "Sat Dec 30 08:53:02 1995",
  bibsource =    "/usr/local/src/bib/bibliography/Ai/lisp.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— **axiom.bib** —

```
@misc{Jenk72,
  author = "Jenks, Richard D.",
  title = {{SCRATCHPAD}},
  volume = "??",
  number = "24",
  pages = "16-17",
  month = "October",
  year = "1972",
  abstract =
    "The following SCRATCHPAD solution of Problem \#2 was run on a 1280K
    virtual machine under CP/CMS time sharing system on a System/360
    model 67. The conversation below is a modification of a program
    originally written by Yngve Sundblad, August 1972. The program uses
    symmetrised formulae, saves certain intermediate results, but does not
    eliminate numerical factors in denominators",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Jenk74,
  author = "Jenks, Richard D.",
  title = {{The SCRATCHPAD language}},
```

```
  journal = "ACM SIGPLAN Notices",
  comment = "reprinted in SIGSAM Bulletin, Vol 8, No. 2, pp 20-30 May 1974",
  volume = "9",
  number = "4",
  pages = "101-111",
  year = "1974",
  doi = "http://dx.doi.org/10.1145807051",
  abstract =
    "SCRATCHPAD is an interactive system for symbolic mathematical
    computation. Its user language, originally intended as a
    special-purpose non-procedural language, was designed to capture the
    style and succinctness of common mathematical notations, and to serve
    as a useful, effective tool for on-line problem solving. This paper
    describes extensions to the language which enable it to serve also as
    a high-level programming language, both for the formal description of
    mathematical algorithms and their efficient implementation.",
  paper = "Jenk74.pdf",
  keywords = "axiomref, printed",
  beebe = "Jenks:1974:SL"
}
```

———-

— Jenks:1974:SL —

```
@Article{Jenks:1974:SL,
  author =       "R. D. Jenks",
  title =        {{The SCRATCHPAD language}},
  journal =      j-SIGPLAN,
  volume =       "9",
  number =       "4",
  pages =        "101--111",
  month =        apr,
  year =         "1974",
  CODEN =        "SINODQ",
  DOI =          "http://dx.doi.org/10.1145807051",
  ISSN =         "0362-1340 (print), 1523-2867 (print), 1558-1160
                 (electronic)",
  ISSN-L =       "0362-1340",
  bibdate =      "Sat Apr 25 11:46:37 MDT 1998",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  classification = "C6140D (High level languages); C7310 (Mathematics
                 computing)",
  conflocation = "Santa Monica, CA, USA; 28-29 March 1974",
  conftitle =    "ACM SIGPLAN Symposium on Very High Level Languages",
  corpsource =   "IBM Thomas J. Watson Res. Center, Yorktown Heights,
                 NY, USA",
  fjournal =     "ACM SIGPLAN Notices",
  link =  "\url{http://portal.acm.org/browse_dl.cfm?idx=J706}",
  keywords =     "formal description; formal programming language; high
                 level programming language; interactive system;
                 mathematical algorithms; natural sciences applications
```

```
                of computers; online problem solving; problem oriented
                languages; SCRATCHPAD language; symbolic mathematical
                computation; user language",
  sponsororg =  "ACM",
  treatment =   "A Application; P Practical",
}
```

———-

— **axiom.bib** —

```
@article{Jenk76a,
  author = "Jenks, Richard D.",
  title = {{Problem \#11: generation of Runge-Kutta equations}},
  journal = "SIGSAM Bulletin",
  volume = "10",
  number = "1",
  year = "1976",
  abstract =
    "Generate a set of equations for an explicit k-th order, m stage,
    Runge-Kutta method for integrating an autonomous system of ordinary
    differential equations, k and m as large as possible. The number of
    conditions and variables for various k and m are given in Table
    1. Tabulate the costs C(m,k)."
}
```

———-

— **axiom.bib** —

```
@inproceedings{Jenk76,
  author = "Jenks, Richard D.",
  title = {{A pattern compiler}},
  booktitle = "Proc. 1976 ACM Symposium on Symbolic and Algebraic Computation",
  series = "SYMSAC '76",
  year = "1976",
  publisher = "ACM Press",
  doi = "http://dx.doi.org/10.1145806324",
  abstract =
    "A pattern compiler for the SCRATCHPAD system provides an efficient
    implementation of sets of user-defined pattern-replacement rules for
    symbolic mathematical computation such as tables of integrals or
    summation identities. Rules are compiled together, with common search
    paths merged and factored out and with the resulting code optimized
    for efficient recognition over all patterns. Matching principally
    involves structural comparison of expression trees and evaluation of
    predicates. Pattern recognizers are ``fully compiled''; if values of
    match variables can be determined by solving equations at compile time.
    Recognition times for several pattern matchers are compared.",
  paper = "Jenk76.pdf",
  keywords = "axiomref, printed",
  beebe = "Jenks:1976:PC"
```

```
}
```

———-

— **Jenks:1976:PC** —

```
@InProceedings{Jenks:1976:PC,
  author =       "Richard D. Jenks",
  editor =       "Richard D. Jenks",
  booktitle =    "Symsac '76: proceedings of the 1976 ACM Symposium on
                 Symbolic and Algebraic Computation, August 10--12,
                 1976, Yorktown Heights, New York",
  title =        {{A pattern compiler}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "60--65",
  year =         "1976",
  DOI =          "http://dx.doi.org/10.1145806324",
  ISBN =         "????",
  ISBN-13 =      "????",
  LCCN =         "QA155.7.E4 .A15 1976; QA9.58 .A11 1976",
  bibdate =      "Thu Jul 26 08:56:43 2001",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  bookpages =    "384",
  keywords =     "Scratchpad",
}
```

———-

— **axiom.bib** —

```
@article{Jenk77,
  author = "Jenks, Richard D.",
  title = {{On the Design of a Mode-Based Symbolic System}},
  journal = "SIGGAM Bulletin",
  volume = "11",
  number = "1",
  pages = "16-19",
  year = "1977",
  abstract =
    "This paper is a preliminary report on the design and implementation
    of a mode-based symbolic programming system and compiler which allows
    programming with rewrite rules and LET and IS pattern-match constructs.
    An important feature of this design is the provision for mode-valued
    variables which allow algebraic domains to be run-time parameters.",
  paper = "Jenk77.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Jenk79,
  author = "Jenks, Richard D.",
  title = {{MODLISP: An Introduction}},
  booktitle = "Proc. ISSAC 1979",
  series = "EUROSAM 79",
  pages = "466-480",
  year = "1979",
  publisher = "Springer-Verlag",
  isbn = "3-540-09519-5",
  comment = "IBM Research Report RC 8073 Jan 1980",
  paper = "Jenk79.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@article{Jenk79a,
  author = "Jenks, Richard D.",
  title = {{SCRATCHPAD/360: reflections on a language design}},
  journal = "SIGSAM",
  volume = "13",
  number = "1",
  pages = "16-26",
  year = "1979",
  comment = "IBM Research RC 7405",
  abstract =
    "The key concepts of the SCRATCHPAD language are described, assessed,
    and illustrated by an example. The language was originally intended as
    an interactive problem solving language for symbolic mathematics.
    Nevertheless, as this paper intends to show, it can be used as a
    programming language as well.",
  paper = "Jenk79a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Jenk81,
  author = "Jenks, Richard D. and Trager, Barry M.",
  title = {{A Language for Computational Algebra}},
  year = "1981",
  booktitle = "Proc. Symp. on Symbolic and Algebraic Manipulation",
  series = "SYMSAC 1981",
  location = "Snowbird, Utah",
  comment = "IBM Research Report 8930",
  abstract =
```

```
    "This paper reports ongoing research at the IBM Research Center on the
    development of a language with extensible parameterized types and
    generic operators for computational algebra. The language provides an
    abstract data type mechanism for defining algorithms which work in as
    general a setting as possible. The language is based on the notions of
    domains and categories. Domains represent algebraic
    structures. Categories designate collections of domains having common
    operations with stated mathematical properties. Domains and categories
    are computed objects which may be dynamically assigned to variables,
    passed as arguments, and returned by functions. Although the language
    has been carefully tailored for the application of algebraic
    computation, it actually provides a very general abstract data type
    mechanism. Our notion of a category to group domains with common
    properties appears novel among programming languages (cf. image
    functor of RUSSELL) and leads to a very powerful notion of abstract
    algorithms missing from other work on data types known to the authors.",
  paper = "Jenk81.pdf",
  keywords = "axiomref, printed"
}
```

———-

— ignore —

```
\bibitem[Jenks 81a]{JT81a} Jenks, R.D.; Trager, B.M.
  title = {{A Language for Computational Algebra}},
SIGPLAN Notices, New York: Association for Computing Machiner, Nov 1981
  keywords = "axiomref"
```

———-

— axiom.bib —

```
@inproceedings{Jenk84a,
  author = "Jenks, Richard D.",
  title = {{The New SCRATCHPAD Language and System for Computer Algebra}},
  booktitle = "Proc. 1984 MACSYMA Users Conference",
  year = "1984",
  pages = "409-??",
  keywords = "axiomref",
  beebe = "Jenks:1984:NSL"
}
```

———-

— Jenks:1984:NSL —

```
@InProceedings{Jenks:1984:NSL,
  author =       "Richard D. Jenks",
  title =        {{The New {SCRATCHPAD} Language and System for Computer
                 Algebra}},
  crossref =     "Golden:1984:PMU",
```

```
  pages =        "409--??",
  year =         "1984",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.1.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— axiom.bib —

```
@inproceedings{Jenk84b,
  author = "Jenks, Richard D.",
  title = {{A primer: 11 keys to New Scratchpad}},
  booktitle = "Proc. EUROSAM ISSAC 1984",
  year = "1984",
  publisher = "Springer-Verlag",
  pages = "123-147",
  isbn = "0-387-13350-X",
  abstract =
    "This paper is an abbreviated primer for the language of new
    SCRATCHPAD, a new implementation of SCRATCHPAD which has been under
    design and development by the Computer Algebra Group at the IBM
    Research Center during the past 6 years. The basic design goals of the
    new SCRATCHPAD language and interface to the user are to provde:
    \begin{itemize}
    \item a ``typeless'' interactive language suitable for on-line solution
    of mathematical problems by novice users with little or no programming
    required, and
    \item a programming language suitable for the formal description of
    algorithms and algebraic structures which can be compiled into run-time
    efficient object code.
    \end{itemize}

    The new SCRATCHPAD language is introduced by 11 keys with each
    successive key introducing an additional capability of the
    language. The language is thus described as a ``concentric'' language
    with each of the 11 levels corresponding to a language subset. These
    levels are more than just a pedagogic device, since they correspond to
    levels at which the system can be effectively used. Level 1 is
    sufficient for naive interactive use; levels 2-8 progressively
    introduce interactive users to capabilities of the language; levels
    9-11 are for system programmers and advanced users. Levesl 2, 4, 6,
    and 7 give users the full power of LISP with a high-level language;
    level 8 introduces ``type declarations;'' level 9 allows polymorphic
    functions to be defined and compiled; levels 10-11 give users an
    Ada-like facility for defining types and packages (those of new
    SCRATCHPAD are dynamically constructable, however). One language is
    used for both interactive and system programming language use,
    although several freedoms such as abbreviation and optional
    type-declarations allowed at top-level are not permitted in system
    code. The interactive language (levels 1-8) is a blend of original
    SCRATCHPAD [GRJY75], some proposed extensions [JENK74], work by Loos
    [LOOS74], SETL [DEWA79], SMP [COWO81], and new ideas; the system
```

```
    programming language (levels 1-11) superficially resembles Ada but is
    more similar to CLU [LISK74] in its semantic design.

    This presentation of the language in this paper omits many details to
    be covered in the SCRATCHPAD System Programming Manual [SCRA84] and an
    expanded version of this paper will serve as a primer for SCRATCHPAD
    users [JESU84].",
  paper = "Jenk84.pdf",
  keywords = "axiomref, printed",
  beebe = "Jenks:1984:PKN"
}
```

———-

— **Jenks:1984:PKN** —

```
@InProceedings{Jenks:1984:PKN,
  author =       "Richard D. Jenks",
  title =        {{A primer: 11 keys to New Scratchpad}},
  crossref =     "Fitch:1984:E",
  pages =        "123--147",
  year =         "1984",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.1.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— **axiom.bib** —

```
@misc{Jenk84c,
  author = "Jenks, Richard D. and Sundaresan, Christine J.",
  title = {{The 11 Keys to SCRATCHPAD: A Primer}},
  year = "1984",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Jenk87a,
  author = "Jenks, Richard D.",
  title = {{1962-1992: The First 30 Years of Symbolic Mathematical
           Programming Systems}},
  journal = "Lecture Notes in Computer Science",
  volume = "296",
  year = "1987",
  pages = "1-1",
  abstract =
    "This talk examines the history and future of symbolic mathematical
    computer systems. This talk will trace the development of three
```

```
    generations of computer algebra systems as typified by an early system
    of 60's: FORMAC, the standalone systems of the 70's: REDUCE and
    MACSYMA, and those developed in the 80's: muMATH, MAPLE, SMP, with
    particular emphasis on Scratchpad II, a system of revolutionary design
    currently under development by IBM Research.

    This talk will trace the progress of algebraic algorithm research in
    the past 25 years, advances in hardware and software technology over
    the same period, and the impact of such progress on the design issues
    of such systems. The talk will conclude with a description of the
    workstation of the future and its anticipated impact on the research
    and educational communities.",
  paper = "Jenk87a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Acad16,
  author = "Academic Search",
  title = {{A Primer: 11 Keys to New Scratchpad}},
  link = "\url{http://libra.msra.cn/publication/645035/a-primer-11-keys-to-new-scratchpad}",
  year = "2016",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Jenk86a,
  author = "Jenks, Richard D.",
  title = {{Basic Algebraic Facilities of the Scratchpad II Computer
           Algebra System}},
  institution = "IBM Research",
  year = "1986",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Jenk86b,
  author = "Jenks, Richard D.",
  title = {{Scratchpad II Examples from INPUT files}},
  institution = "IBM Research",
  year = "1986",
  keywords = "axiomref"
```

```
}
```

———-

— axiom.bib —

```
@techreport{Jenk86,
  author = "Jenks, Richard D. and Sutor, Robert S. and Watt, Stephen M.",
  title = {{Scratchpad II: An Abstract Datatype System for Mathematical
           Computation}},
  institution = "IBM Research",
  year = "1986",
  type = "Research Report",
  number = "RC 12327 (\#55257)",
  link = "\url{http://www.csd.uwo.ca/~watt/pub/reprints/1987-ima-spadadt.pdf}",
  abstract = "
    Scratchpad II is an abstract datatype language and system that is
    under development in the Computer Algebra Group, Mathematical Sciences
    Department, at the IBM Thomas J. Watson Research Center. Some features
    of APL that made computation particularly elegant have been borrowed.
    Many different kinds of computational objects and data structures are
    provided. Facilities for computation include symbolic integration,
    differentiation, factorization, solution of equations and linear
    algebra.  Code economy and modularity is achieved by having
    polymorphic packages of functions that may create datatypes. The use
    of categories makes these facilities as general as possible.",
  paper = "Jenk86.pdf",
  keywords = "axiomref, printed",
  beebe = "Jenks:1986:SIA"
}
```

———-

— Jenks:1986:SIA —

```
@TechReport{Jenks:1986:SIA,
  author =       "Richard D. Jenks and Robert S. Sutor and Stephen M.
                 Watt",
  title =        {{Scratchpad {II}: an abstract datatype system for
                 mathematical computation}},
  type =         "Research Report",
  number =       "RC 12327 (\#55257)",
  institution =  "International Business Machines Inc., Thomas J. Watson
                 Research Center",
  address =      "Yorktown Heights, NY, USA",
  pages =        "23",
  year =         "1986",
  bibdate =      "Thu Oct 31 17:23:28 2002",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  keywords =     "Abstract data types (Computer science); Operating
                 systems (Computers)",
}
```

———-

— **axiom.bib** —

```
@inproceedings{Jenk87,
  author = "Jenks, Richard D. and Sutor, Robert S. and Watt, Stephen M.",
  title = {{Scratchpad II: an Abstract Datatype System for
           Mathematical Computation'}},
  booktitle = "Proceedings Trends in Computer Algebra",
  series = "Lecture Notes in Computer Science 296",
  pages = "157-182",
  publisher = "Springer-Verlag",
  isbn = "0-387-18928-9",
  year = "1987",
  comment = "IBM Research Report RC 12327 (\#55257) See Jenks86.pdf",
  abstract =
    "Scratchpad II is an abstract datatype language and system that is
    under development in the Computer Algebra Group, Mathematical Sciences
    Department, at the IBM Thomas J. Watson Research Center. Many
    different kinds of computational objects and data structures are
    provided. Facilities for computation include symbolic integration,
    differentation, factorization, solution of equations and linear
    algebra. Code economy and modularity is achieved by having polymorphic
    packages of functions that may create datatypes. The use of categories
    makes these facilities as general as possible.",
  paper = "Jenk87.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Jenk88,
  author = "Jenks, Richard D. and Sutor, Robert S. and Watt, Stephen M.",
  title = {{Scratchpad II: An Abstract Datatype System for Mathematical
           Computation}},
  booktitle = "Mathematical Aspects of Scientific Software",
  year = "1988",
  pages = "157-182",
  publisher = "Springer",
  isbn = "0-387-18928-9",
  abstract =
    "Scratchpad II is an abstract datatype language and system that is
    under development in the Computer Algebra Group, Mathematical Sciences
    Department, at the IBM Thomas J. Watson Research Center. Many
    different kinds of computational objects and data structures are
    provided. Facilities for computation include symbolic integration,
    differentation, factorization, solution of equations and linear
    algebra. Code economy and modularity is achieved by having polymorphic
    packages of functions that may create datatypes. The use of categories
```

```
    makes these facilities as general as possible.",
  keywords = "axiomref",
  beebe = "Jenks:1988:SIA"
}
```

———-

— **Jenks:1988:SIA** —

```
@InProceedings{Jenks:1988:SIA,
  author =       "R. D. Jenks and R. S. Sutor and S. M. Watt",
  title =        {{Scratchpad II: An Abstract Datatype System for
                 Mathematical Computation}},
  crossref =     "Janssen:1988:TCA",
  pages =        "12--37",
  year =         "1988",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— **ignore** —

```
\bibitem[Jenks 88a]{Jen88a} Jenks, R. D.
  title = {{A Guide to Programming in BOOT}},
Computer Algebra Group, Mathematical Sciences Department, IBM Research
Draft September 5, 1988
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Jenks 88b]{Jen88b} Jenks, Richard
  title = {{The Scratchpad II Computer Algebra System Interactive
            Environment Users Guide}},
 Spring 1988
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Jenk88d,
  author = "Jenks, Richard D.",
  title = {{Scratchpad II: A computer algebra language and system}},
  journal = "The Journal of the Acoustical Society of America",
  link = "\url{https://asa.scitation.org/doi/pdf/10.1121/1.2025115}",
  year = "1988",
  volume = "83",
```

```
  number = "S1",
  pages = "S106",
  abstract =
    "The Scratchpad II system represents a new generation of systems for
    doing symbolic mathematics, based on modern algebra and abstract data
    types. A large number of facilities are provided, for example:
    symbolic integration, ``infinite'' power series, differential operators,
    Cartesian tensors, and solution of nonlinear systems. Scratchpad II
    has been designed from the outset to be extendible. The system
    introduces a new data abstraction notion, the ``category,'' to express
    intricate interrelationships between data types. The result design
    permits the compilation of algorithms described in their most natural
    mathematical setting. The use of categories guarantees user defined
    types and packages are compatible with each other and with built in
    facilities. This system provides a single highlevel language with an
    intepreter and compiler. The language can be used by the naive user
    for convenient interactive mathematics calculations and by the
    advanced user for the efficient implementation of
    algorithms. Scratchpad II is built on Lisp/VM and runs on IBM/370
    class mainframes. An implementation of the system on the RT/PC is
    expected soon.",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@book{Jenk92,
  author = "Jenks, Richard D. and Sutor, Robert S.",
  title = {{AXIOM: The Scientific Computation System}},
  publisher = "Springer-Verlag, Berlin, Germany",
  year = "1992",
  isbn = "0-387-97855-0",
  keywords = "axiomref",
  beebe = "Jenks:1992:ASC"
}
```

———-

— Jenks:1992:ASC —

```
@Book{Jenks:1992:ASC,
  author =       "Richard D. Jenks and Robert S. Sutor",
  title =        {{AXIOM: The Scientific Computation System}},
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "xxiv + 742",
  year =         "1992",
  ISBN =         "0-387-97855-0 (New York), 3-540-97855-0 (Berlin)",
  ISBN-13 =      "978-0-387-97855-0 (New York), 978-3-540-97855-8
                 (Berlin)",
  LCCN =         "QA76.95.J46 1992",
```

```
  bibdate =      "Fri Dec 29 18:16:15 1995",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— axiom.bib —

```
@InProceedings{Jenk94,
  author = "Jenks, Richard D. and Trager, Barry M.",
  title = {{How to make AXIOM into a Scratchpad}},
  booktitle = "Proceedings of the ACM-SIGSAM 1989 International
           Symposium on Symbolic and Algebraic Computation, ISSAC '94",
  series = "ISSAC 94",
  year = "1994",
  pages = "32-40",
  isbn = "0-89791-638-7",
  publisher = "ACM Press",
  address = "New York, NY, USA",
  abstract =
    "Scratchpad [GrJe71] was a computer algebra system developed in the
    early 1970s.  Like M\&M (Maple [CGG91ab] and Mathematical [W01S92]) and
    other systems today, Scratchpad had one principal representation for
    mathematical formulae based on ``expression trees''.  Its user interface
    design was based on a pattern-matching paradigm with infinite rewrite-
    rule semantics, providing what we believe to be the most natural
    paradigm for interactive symbolic problem solving.  Like M\&M, however,
    user programs were interpreted, often resulting in poor performance
    relative to similar facilities coded in standard programming languages
    such as FORTRAN and C.

    Scratchpad development stopped in 1976 giving way to a new system
    design ([JenR79], [JeTr81]) that evolved into AXIOM [JeSu92].
    AXIOM has a strongly-typed programming language for building a library
    of parameterized types and algorithms, and a type-inferencing
    interpreter that accesses the library and can build any of an infinite
    number of types for interactive use.

    We suggest that the addition of an expression tree type to AXIOM can
    allow users to operate with the same freedom and convenience of
    untyped systems without giving up the expressive power and run-time
    efficiency provided by the type system.  We also present a design that
    supports a multiplicity of programming styles, from the Scratchpad
    pattern-matching paradigm to functional programming to more
    conventional procedural programming.  The resulting design seems to us
    to combine the best features of Scratchpad with current AXIOM and to
    offer a most attractive, flexible, and user-friendly environment for
    interactive problem solving.

    Section 2 is a discussion of design issues contrasting AXIOM with
    other symbolic systems.  Sections 3 and 4 is an assessment of AXIOMs
    current design for building libraries and interactive use.  Section 5
```

```
    describes a new interface design for AXIOM, its resulting paradigms,
    and its underlying semantic model.  Section 6 compares this work with
    others.",
  paper = "Jenk94.pdf",
  keywords = "axiomref",
  beebe = "Jenks:1994:HMA"
}
```

———-

— Jenks:1994:HMA —

```
@InProceedings{Jenks:1994:HMA,
  author =       "Richard D. Jenks and Barry M. Trager",
  title =        {{How to make {AXIOM} into a Scratchpad}},
  crossref =     "ACM:1994:IPI",
  pages =        "32--40",
  year =         "1994",
  MRclass =      "68W30 (Symbolic computation and algebraic
                 computation)",
  bibdate =      "Tue Sep 17 06:29:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  ZMnumber =     "0945.68543",
  abstract =     "Scratchpad (Griesmer and Jenks, 1971) was a computer
                 algebra system that had one principal representation
                 for mathematical formulae based on expression trees.
                 Its user interface design was based on a
                 pattern-matching paradigm with infinite rewrite rule
                 semantics, providing what we believe to be the most
                 natural paradigm for interactive symbolic problem
                 solving. Like M and M, however, user programs were
                 interpreted, often resulting in poor performance
                 relative to similar facilities coded in standard
                 programming languages such as FORTRAN and C. Scratchpad
                 development stopped in 1976 giving way to a new system
                 design that evolved into AXIOM. AXIOM has a
                 strongly-typed programming language for building a
                 library of parameterized types and algorithms, and a
                 type-inferencing interpreter that accesses the library
                 and can build any of an infinite number of types for
                 interactive use. We suggest that the addition of an
                 expression tree type to AXIOM can allow users to
                 operate with the same freedom and convenience of
                 untyped systems without giving up the expressive power
                 and run-time efficiency provided by the type system. We
                 also present a design that supports a multiplicity of
                 programming styles, from the Scratchpad
                 pattern-matching paradigm to functional programming to
                 more conventional procedural programming.",
  acknowledgement = ack-nhfb,
  affiliation =  "IBM Thomas J. Watson Res. Center, Yorktown Heights,
                 NY, USA",
  classification = "C6180 (User interfaces); C7310 (Mathematics
```

```
                computing)",
  keywords =    "AXIOM; C; Computer algebra system; Expression trees;
                FORTRAN; Functional programming; Infinite rewrite rule
                semantics; Library; Mathematical formulae;
                Pattern-matching; Procedural programming; Run-time
                efficiency; Scratchpad; Strongly-typed programming
                language; Symbolic problem solving; Type-inferencing
                interpreter; Untyped systems; User interface design;
                User programs",
  language =    "English",
  thesaurus =   "Mathematics computing; Pattern matching; Program
                interpreters; Programming; Symbol manipulation; User
                interfaces",
}
```

— axiom.bib —

```
@phdthesis{Joha14,
  author = "Johansson, Fredrik",
  title = {{Fast and Rigorous Computation of Special Functions to High
           Precision}},
  school = "Johannes Kepler University, Linz, Austria RISC",
  year = "2014",
  abstract =
    "The problem of efficiently evaluating special functions to high
    precision has been considered by numerous authors. Important tools
    used for this purpose include algorithms for evaluation of linearly
    recurrent sequences, and algorithms for power series arithmetic.

    In this work, we give new baby-step, giant-step algorithms for
    evaluation of linearly recurrent sequences involving an expensive
    parameter (such as a high-precision real number) and for computing
    compositional inverses of power series. Our algorithms do not have the
    best asymptotic complexity, but they are faster than previous
    algorithms in practice over a large input range.

    Using a combination of techniques, we also obtain efficient new
    algorithms for numerically evaluating the gamma function $\Gamma(z)$
    and the Hurwitz zeta function $\zeta(s,a)$, or Taylor series
    expansions of those functions, with rigorous error bounds. Our methods
    achieve softly optimal complexity when computing a large number of
    derivatives to proportionally high precision.

    Finally, we show that isolated values of the integer partition
    function $p(n)$ can be computed rigorously with softly optimal
    complexity by means of the Hardy-Ramanujan-Rademacher formula and
    careful numerical evaluation.

    We provide open source implementations which run significantly faster
    than previous published software. The implementations are used for
    record computations of the partition function, including the
```

```
    tabulation of several billion Ramanujan-type congruences, and of
    Taylor series associated with the Reimann zeta function.",
  paper = "Joha14.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Joha02,
  author = "Johansson, Leif and Lambe, Larry and Skoldberg, Emil",
  title = {{On Constructing Resolutions over the Polynomial Algebra}},
  journal = "Homology, Homotopy and Applications",
  volume = "4",
  number = "2",
  year = "2002",
  pages = "315-336",
  link = "\url{http://projecteuclid.org/download/pdf\_1/euclid.hha/1139852468}",
  abstract =
    "Let $k$ be a field, and $A$ be a polynomial algebra over $k$.
    Let $I \subseteq A$ be an ideal. We present a novel method for
    computing resolutions of $A/I$ over $A$. The method is a synthesis
    of Groebner basis techniques and homological perturbation theory.
    The examples in this paper were computed using computer algebra.",
  paper = "Joha02.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{John94,
  author = "Johnson, M.E. and Rogers, C. and Schief, W.K. and Seiler, W.M.",
  title = {{On moving pseudospherical surfaces: a generalised Weingarten
           system and its formal analysis}},
  journal = "Lie Groups Appl.",
  volume = "1",
  pages = "124-136",
  year = "1994",
  abstract =
    "The connection between the motion of certain curves in $\mathbb{R}^3$
    and $1+1$-dimensional soliton equations is by now well-established. On the
    other hand, the sine-Gordon and other integrable equations may be
    readily derived via the classical geometry of stationary
    pseudospherical surfaces. Here, the motion of pseudospherical surfaces
    $S$ is considered in a natural orthonormal triad formulation. In one case,
    in a motion in which the Gaussian curvature of $S$ remains constant in
    time, an integrable nonlinear evolution equation is derived which has
    its origin in the description of wave propagation in an anharmonic
    crystal. In a second case, wherein the Gaussian curvature is allowed
    to vary in time, a classical generalised Weingarten system is derived
```

```
    in connection with the purely normal propagation of a pseudospherical
    surface. This is linked to triply orthogonal coordinate systems of
    Bianchi type. The generalised Weingarten system incorporates an
    integrable $2+1$-dimensional sine-Gordon equation. The arbitrariness of the
    solutions of the generalised Weingarten system is determined via a
    completion procedure.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Joll13,
  author = "Jolly, Raphael",
  title = {{Category as Type Classes in the Scala Algebra System}},
  journal = "LNCS",
  pages = "209-218",
  year = "2013",
  abstract =
    "A characterization of the categorical view of computer algebra is
    proposed. Some requirements on the ability for abstraction that
    programming languages must have in order to allow a categorical
    approach is given. Object-oriented inheritance is presented as a
    suitable abstraction scheme and exemplified by the Java Algebra
    System. Type classes are then introduced as an alternative abstraction
    scheme and shown to be eventually better suited for modeling
    categories. Pro and cons of the two approaches are discussed and a
    hybrid solution is exhibited.",
  paper = "Joll13.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Rainer 14]{Rain14} Joswig, Rainer
  title = {{2014: 30+ Years Common Lisp the Language}},
  link = "\url{http://lispm.de/30ycltl}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Joyn08,
  author = "Joyner, David and Stein, William",
  title = {{Open Source Mathematical Software: A White Paper}},
  year = "2008",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.7499}",
```

```
  abstract =
    "Open source software has had a profound effect on computing during
    the last decade, especially on web servers (Apache), web browsers
    (Firefox), operating systems (Linux and OS X), and programming
    languages (GC C, Java, Python, Perl, etc.). The purpose of this paper
    is to put forward the case that open source development methodologies
    might also have a positive effect on mathematical software,
    especially if the National Science Foundation (NSF) increases their
    support of open source mathematical software de velopment.  We argue
    that careful funding of open source mathematical software may lead to
    a lower total cost of ownership in the research and education
    community, and to more efficient and trustworthy mathematical software.",
  paper = "Joyn08.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Joyn16,
  author = "Joyner, David",
  title = {{Links to some open source mathematical programs}},
  link = "\url{http://www.opensourcemath.org/opensource\_math.html}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Joyn08a,
  author = "Joyner, David",
  title = {{Open source computer algebra systems: Axiom}},
  journal = "ACM Commun. Comput. Algebra",
  volume = "42",
  number = "1-2",
  pages = "39-47",
  year = "2008",
  abstract =
    "This survey will look at Axiom, a free and very powerful computer
    algebra system available. It is a general purpose CAS useful for
    symbolic computation, research, and the development of new
    mathematical algorithms. Axiom is similar in some ways to Maxima,
    covered in the survey, but different in many ways as well. Axiom,
    Maxima, and SAGE, are the largest of the general-purpose open-source
    CASs.",
  keywords = "axiomref, TPDref",
  beebe = "Joyner:2008:OSC"
}
```

———-

— **Joyner:2008:OSC** —

```
@Article{Joyner:2008:OSC,
  author =       "David Joyner",
  title =        {{Open source computer algebra systems: Axiom}},
  journal =      j-ACM-COMM-COMP-ALGEBRA,
  volume =       "42",
  number =       "1--2",
  pages =        "39--47",
  month =        mar # "/" # jun,
  year =         "2008",
  CODEN =        "????",
  DOI =          "http://doi.acm.org/10.1145/1394042.1394046",
  ISSN =         "1932-2232 (print), 1932-2240 (electronic)",
  ISSN-L =       "1932-2232",
  bibdate =      "Tue Aug 12 17:30:40 MDT 2008",
  bibsource =    "http://portal.acm.org/;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "This survey will look at Axiom, a free and very
                 powerful computer algebra system available. It is a
                 general purpose CAS useful for symbolic computation,
                 research, and the development of new mathematical
                 algorithms. Axiom is similar in some ways to Maxima,
                 covered in the survey [J1], but different in many ways
                 as well. Axiom, Maxima, and SAGE [S], are the largest
                 of the general-purpose open-source CAS's. If you want
                 to 'take a test drive,' Axiom can be tested without
                 installation via the web interface [AS] or the SAGE
                 online interface [S].",
  acknowledgement = ack-nhfb,
  fjournal =     "ACM Communications in Computer Algebra",
}
```

———-

— **ignore** —

```
\bibitem[Joswig 03]{JT03} Joswig, Michael; Takayama, Nobuki
  title = {{Algebra, geometry, and software systems}},
Springer-Verlag ISBN 3-540-00256-1 p291
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Joyner 06]{J006} Joyner, David
  title = {{OSCAS - Maxima}},
SIGSAM Communications in Computer Algebra, 157 2006
 link = "\url{http://sage.math.washington.edu/home/wdj/sigsam/oscas-cca1.pdf}",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Joyner 14]{JO14} Joyner, David
  title = {{Links to some open source mathematical programs}},
  link = "\url{http://www.opensourcemath.org/opensource_math.html}",
  keywords = "axiomref"
```

———-

1.40.11 K

— **axiom.bib** —

```
@inproceedings{Kajl92,
  author = "Kajler, Norbert",
  title = {{CAS/PI: a Portable and Extensible Interface for Computer
           Algebra Systems}},
  year = "1992",
  booktitle = "Proc. ISSAC 1992",
  series = "ISSAC 1992",
  pages = "376-386",
  isbn = "0-89791-489-9 (soft cover) 0-89791-490-2 (hard cover)",
  abstract =
    "CAS/$\pi$ is a Computer Algebra System graphic user interface
    designed to be highly portable and extensible. It has been developed
    by composition of pre-existing software tools such as Maple, Sisyphe,
    or Ulysse systems, ZicVis 3-D plotting library, etc, using control
    integration technology and a set of high level graphic toolkits to
    build the formula editor and the dialog manager. The main aim of
    CAS/$\pi$ is to allow a wide range of runtime recon gurations and
    extensions. For instance, it is possible to add new tools to a running
    system, to modify connections between working tools, to extend the set
    of graphic symbols managed by the formula editor, to design new high
    level editing commands based on the syntax or semantics of
    mathematical formulas, to customize and extend the menu-button based
    user interface, etc. More generally, CAS/$\pi$ can be seen equally as
    a powerful system-independent graphic user interface enabling
    inter-systems communications, a toolkit to allow fast development of
    custom-made scientific software environments, or a very convenient
    framework for experimenting with computer algebra systems protocols
    and man-machine interfaces.",
  paper = "Kajl92.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Kajl94,
  author = "Kajler, Norbert and Soiffer, Neil",
  title = {{Some human interaction issues in computer algebra}},
  journal = "SIGSAM Bulletin",
  volume = "28",
  number = "1",
  pages = "18-28",
  year = "1994",
  abstract =
    "This paper addresses some of the current issues concerning the
    improvement of user interfaces for computer algebra systems. Some
    state of the art commercial software as well as research prototypes
    are presented, followed by a description of present research
    directions.",
  paper = "Kajl94.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kand89,
  author = "Kandri-Rody, Abdelilah and Kapur, Deepak and Winkler, Franz",
  title = {{Knuth-Bendix Procedure and Buchberger Algorithm - A Synthesis}},
  booktitle = "ISSAC'89",
  publisher = "ACM Press",
  pages = "55-67",
  year = "1989",
  isbn = "0-89791-325-6",
  abstract =
    "The Knuth-Bendix procedure for the completion of a rewrite rule
    system and the Buchberger algorithm for computing a Groebner basis
    of a polynomial ideal are very similar in two respects: they both
    start wtih an arbitrary specification of an algebraic structure
    (axioms for an equational theory and a basis for a polynomial
    ideal, respectively) which is transformed to a very special
    specification of this algebraic structure (a complete rewrite rule
    system and a Groebner basis of the polynomial ideal, respectively).
    This special specification allows to decide many problems concerning
    the given algebraic structure. Moreover, both algorithms achieve
    their goals by employing the same basic concepts: formation of
    critical pairs and completion.

    Although the two methods are obviously related, the exact nature
    of this relation remains to be clarified. Based on previous work
    we show how the Knuth-Bendix procedure and the Buchberger algorithm
    can be seen as special cases of a more general completion procedure.",
  paper = "Kand89.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Kani16,
  author = "Kanigel, Robert",
  title = {{OldQuotes}},
  link = "\url{http://www.oldquotes.com}",
  year = "2016",
  abstract =
    "Sometimes in studying Ramanujan's work, George Andrews said at
    another time, ``I have wondered how much Ramanujan could have done if
    he had had MACSYMA or SCRATCHPAD or some other symbolic algebra package''"
}
```

———-

— **axiom.bib** —

```
@book{Kapl05,
  author = "Kaplan, Michael",
  title = {{Computer Algebra}},
  publisher = "Springer, Berlin, Germany",
  year = "2005",
  isbn = "3-540-21379-1",
  comment = "German Language",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Kapu81,
  author = "Kapur, D. and Musser, D.R. and Stepanov, A.A.",
  title = {{Operators and Algebraic Structures}},
  link = "\url{http://www.stepanovpapers.com/p59-kapur.pdf}",
  year = "1981",
  abstract =
    "Operators in functional languages such as APL and FFP are a useful
    programming concept.  However, this concept cannot be fully
    exploited in these languages because of certain constraints.  It is
    proposed that an operator should be associated with a structure having
    the algebraic properties on which the operator's behavior depends.
    This is illustrated by introducing a language that provides mechanisms
    for defining structures and operators on them.  Using this language,
    it is possible to describe algorithms abstractly, thus emphasizing
    the algebraic properties on which the algorithms depend.  The role
    that formal representation of mathematical knowledge can play in the
    development of programs is illustrated through an example.  An
    approach for associating complexity measures with a structure and
    operators is also suggested.  This approach is useful in analyzing
    the complexity of algorithms in an abstract setting.",
  paper = "Kapu81.pdf"
```

```
}
```

———-

— axiom.bib —

```
@inproceedings{Kaue08,
  author = "Kauers, Manuel",
  title = {{Integration of Algebraic Functions: A Simple Heuristic for
           Finding the Logarithmic Part}},
  booktitle = "Proc ISSAC 2008",
  series = "ISSAC '08",
  year = "2008",
  pages = "133-140",
  isbn = "978-1-59593-904",
  link =
     "\url{http://www.risc.jku.at/publications/download/risc_3427/Ka01.pdf}",
  abstract =
    "A new method is proposed for finding the logarithmic part of an
    integral over an algebraic function. The method uses Groebner bases
    and is easy to implement. It does not have the feature of finding a
    closed form of an integral whenever there is one. But it very often
    does, as we will show by a comparison with the built-in integrators of
    some computer algebra systems.",
  paper = "Kaue08.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@inproceedings{Kead93,
  author = "Keady, G. and Nolan, G.",
  title = {{Production of Argument SubPrograms in the AXIOM -- NAG link:
           examples involving nonlinear systems}},
  booktitle = "Proc. Workshop on Symbolic and Numeric Computation",
  location = "Helsinki",
  year = "1993",
  pages = "13-32",
  comment = "NAG Technical Report TR1/94",
  link = "\url{school.maths.uwa.edu.au/%7Ekeady/KeadyPapers/93Helsinki.ps}",
  algebra =
   "\newline\refto{domain ASP1 Asp1}
    \newline\refto{domain ASP10 Asp10}
    \newline\refto{domain ASP12 Asp12}
    \newline\refto{domain ASP19 Asp19}
    \newline\refto{domain ASP20 Asp20}
    \newline\refto{domain ASP24 Asp24}
    \newline\refto{domain ASP27 Asp27}
    \newline\refto{domain ASP28 Asp28}
    \newline\refto{domain ASP29 Asp29}
```

```
    \newline\refto{domain ASP30 Asp30}
    \newline\refto{domain ASP31 Asp31}
    \newline\refto{domain ASP33 Asp33}
    \newline\refto{domain ASP34 Asp34}
    \newline\refto{domain ASP35 Asp35}
    \newline\refto{domain ASP4 Asp4}
    \newline\refto{domain ASP41 Asp41}
    \newline\refto{domain ASP42 Asp42}
    \newline\refto{domain ASP49 Asp49}
    \newline\refto{domain ASP50 Asp50}
    \newline\refto{domain ASP55 Asp55}
    \newline\refto{domain ASP6 Asp6}
    \newline\refto{domain ASP7 Asp7}
    \newline\refto{domain ASP73 Asp73}
    \newline\refto{domain ASP74 Asp74}
    \newline\refto{domain ASP77 Asp77}
    \newline\refto{domain ASP78 Asp78}
    \newline\refto{domain ASP8 Asp8}
    \newline\refto{domain ASP80 Asp80}
    \newline\refto{domain ASP9 Asp9}",
  abstract =
    "Dewar's paper [6] earlier in this Proceedings 'sketches out the
    design of the AXIOM-NAG link' and gives a general account of new tools
    for generating Fortran. This paper is a sequel to [6]. Here we present
    'examples' of some of the items discussed in [6]. We have attempted to
    achieve some coherence by selecting our 'examples' from just the one
    application area - solving nonlinear systems.",
  paper = "Kead93.pdf",
  keywords = "axiomref",
  beebe = "Keady:1994:PAS"
}
```

— Keady:1994:PAS —

```
@TechReport{Keady:1994:PAS,
  author =       "G. Keady and G. Nolan",
  title =        {{Production of Argument SubPrograms in the AXIOM
                 --- NAG link: examples involving nonlinear systems}},
  number =       "TR1/94 ATR/7 (NP2680)",
  institution =  inst-NAG,
  address =      inst-NAG:adr,
  pages =        "??",
  year =         "1994",
  bibdate =      "Thu Jan 04 18:40:00 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  link =          "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
  acknowledgement = ack-nhfb,
}
```

— **axiom.bib** —

```
@inproceeding{Kell14,
  author = "Kell, Stephen",
  title = "In Search of Types",
  booktitle = "Int. Symp. on New Ideas, New Pradigms, and Reflections
               on Programming and Software (Onward!)",
  publisher = "ACM",
  year = "2014",
  abstract =
    "The concept of ``type'' has been used without a consistent,
    precise definition in discussions about programming languages for
    60 years. In this essay I explore various concepts lurking behind
    distinct uses of this word, highlighting two traditions in which
    the word came into use largely independently: engineering
    traditions on the one hand, and those of symbolic logic on the
    other. These traditions are founded on differing attitudes to the
    nature and propose of abstraction, but their distinct uses of
    ``type'' have never been explicitly unified. One result is that
    discourse {\sl across} these traditions often finds itself at
    cross purposes, such as {\sl overapplying} one sense of ``type''
    where another is appropriate, and occasionally proceeding to draw
    wrong conslusions. I illustrate this with examples from well-known
    and justly well-regarded literature, and argue that ongoing
    developments in both the theory and practice of programming make
    now a good time to resolve these problems.",
  paper = "Kell14.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Kels99,
  author = "Kelsey, Tom",
  title = {{Formal Methods and Computer Algebra: A Larch Specification of
          AXIOM Categories and Functors}},
  school = "University of St Andrews",
  year = "1999",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Kels00a,
  author = "Kelsey, Tom",
  title = {{Formal specification of computer algebra}},
  abstract =
    "We investigate the use of formal methods languages and tools in the
    design and development of computer algebra systems (henceforth CAS).
```

```
    We demonstrate that errors in CAS design can be identified and
    corrected by the use of (i) abstract specifications of types and
    procedures, (ii) automated proofs of properties of the specifications,
    and (iii) interface specifications which assist the verification of
    pre- and post conditions of implemented code.",
  paper = "Kels00a.pdf",
  keywords = "axiomref, CAS-Proof, printed, DONE"
}
```

———-

— **ignore** —

```
\bibitem[Kelsey 00b]{Kel00b} Kelsey, Tom
  title = {{Formal specification of computer algebra}},
(slides) University of St Andrews, Sept 21, 2000
  link = "\url{http://www.cs.st-andrews.cs.uk/~tom/pub/fscbstalk.ps}",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Kendall 99a]{Ken99a} Kendall, W.S.
  title = {{Itovsn3 in AXIOM: modules, algebras and stochastic differentials}},
  link = "\url{http://www2.warwick.ac.uk/fac/sci/statistics/staff/academic-research/kendall/personal/ppt/328.ps.
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Kend01,
  author = "Kendall, Wilfrid S.",
  title = {{Symbolic It\^o calculus in AXIOM: an ongoing story}},
  journal = "Statistics and Computing",
  volume = "11",
  pages = "25-35",
  year = "2001",
  link = "\url{http://www2.warwick.ac.uk/fac/sci/statistics/staff/academic-research/kendall/personal/ppt/327.ps.
  abstract =
    "Symbolic It\^o calculus refers both to the implementation of the
    It\^o calculus algebra package and to its application. This article
    reports on progress in the implementation of It\^o calculus in the
    powerful and innovative computer algebra package AXIOM, in the context
    of a decade of previous implementations and applications. It is shown
    how the elegant algebraic structure underylying the expressive and
    effective formalism of It\^o calculus can be implemented directly in
    AXIOM using the package's programmable facilities for ``strong
    typing'' of computational objects.  An application is given of the use
    of the implementation to provide calculations for a new proof, based
```

```
    on stochastic differentials, of the Mardia-Dryden distribution from
    statistical shape theory.",
  paper = "Kend01.pdf",
  keywords = "axiomref, CAS-Proof, printed",
  beebe = "Kendall:2001:SIC"
}
```

———-

— Kendall:2001:SIC —

```
@Article{Kendall:2001:SIC,
  author =       "Wilfrid S. Kendall",
  title =        {{Symbolic {It{\^o}} calculus in {AXIOM}: an ongoing
                 story}},
  journal =      j-STAT-COMP,
  volume =       "11",
  number =       "1",
  pages =        "25--35",
  year =         "2001",
  CODEN =        "STACE3",
  DOI =          "http://dx.doi.org/10.1023/A:1026553731272",
  ISSN =         "0960-3174",
  ISSN-L =       "0960-3174",
  MRclass =      "Database Expansion Item",
  MRnumber =     "MR1837142",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  fjournal =     "Statistics and Computing",
  keywords =     "computer algebra; coupling of random processes; Dryden
                 density Mathematica; financial mathematics; Itovsn3;
                 It{\^o} calculus; It{\^o} formula; Macsyma; Maple;
                 Mardia; REDUCE; semimartingale; statistics of shape;
                 stochastic calculus; stochastic integral; symbolic
                 It{\^o} calculus; XIOM",
}
```

———-

— axiom.bib —

```
@article{Kend07,
  author = "Kendall, Wilfrid S.",
  title = {{Coupling all the Levy Stochastic Areas of Multidimensional
           Brownian Motion}},
  journal = "The Annals of Probability",
  volume = "35",
  number = "3",
  pages = "935-953",
  year = "2007",
  comment = "Author used Axiom for computation but says missed citation",
  link = "\url{http://arxiv.org/pdf/math/0512336v2.pdf}",
  abstract =
    "It is shown how to construct a successful co-adapted coupling of two
```

```
    copies of an $n$-dimensional Brownian motion ($B_1,\ldots,B_n$) while
    simultaneously coupling all corresponding copies of the L{\'e}vy
    stochastic areas $\int B_idB_j - \int B_j dB_i$. It is conjectured
    that successful co-adapted couplings still exist when the L{\'e}vy
    stochastic areas are replaced by a finite set of multiply iterated
    path- and time-integrals, subject to algebraic compatibility of the
    initial conditions.",
  paper = "Kend07.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Kenn01,
  author = "Kennedy, A.D.",
  title = {{Semantics of Categories in Aldor}},
  year = "2001",
  abstract =
    "We consider some questions about the semantics of Aldor regarding the
    way that types can be considered on an equal footing with any other
    objects in the language. After a digression into the relationship
    between Aldor categories and mathematical categories, we shall discuss
    the more practical issue of the limitations of the {\tt define}
    keyword and what the compiler should do about them.",
  paper = "Kenn01.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kerb98,
  author = "Kerber, Manfred and Kohlhase, Michael and Volker, Sorge",
  title = {{Integrating computer algebra into proof planning}},
  journal = "J. Autom. Reasoning",
  volume = "21",
  number = "3",
  year = "1998",
  pages = "327-355",
  link =
   "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.40.3914}",
  abstract =
    "Mechanized reasoning systems and computer algebra systems have
    different objectives. Their integration is highly desirable, since
    formal proofs often involve both of the two different tasks proving
    and calculating. Even more important, proof and computation are often
    interwoven and not easily separable.

    In this article, we advocate an integration of computer algebra into
    mechanized reasoning systems at the proof plan level. This approach
```

```
    allows us to view the computer algebra algorithms as methods, that is,
    declarative representations of the problem-solving knowledge specific
    to a certain mathematical domain. Automation can be achieved in many
    cases by searching for a hierarchic proof plan at the method level by
    using suitable domain-specific control knowledge about the
    mathematical algorithms. In other words, the uniform framework of
    proof planning allows us to solve a large class of problems that are
    not automatically solvable by separate systems.

    Our approach also gives an answer to the correctness problems inherent
    in such an integration. We advocate an approach where the computer
    algebra system produces high-level protocol information that can be
    processed by an interface to derive proof plans. Such a proof plan in
    turn can be expanded to proofs at different levels of abstraction, so
    the approach is well suited for producing a high-level verbalized
    explication as well as for a low-level, machine-checkable,
    calculus-level proof. We present an implementation of our ideas and
    exemplify them using an automatically solved example.",
  paper = "Kerb98.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@phdthesis{King70,
  author = "King, James Cornelius",
  title = {{A Program Verifier}},
  school = "Carnegie Mellon University",
  year = "1970"
}
```

———-

— **axiom.bib** —

```
@article{Kocb96,
  author = "Kocbach, Ladislav and Liska, Richard",
  title = {{Generation and Verification of Algorithms for Symbolic\_Numeric
           Processing}},
  journal = "J. Symbolic Computation",
  volume = "11",
  pages = "1-16",
  year = "1996",
  abstract =
    "Some large scale physical computations require algorithms performing
    symbolic computations with a particular class of algebraic formulas in
    a numerical code. Developing and implementing such algorithms in a
    numerical programming language is a tedious and error prone task. The
    algorithms can be developed in a computer algebra system and their
    correctness can be checked by comparison with built-in facilities of
```

```
    the system so that the system is used as an advanced debugging
    tool. After that a numerical code for the algorithms is automatically
    generated from the same source code. The proposed methodolgy is
    explained in detail on a simple example. Real applications to
    calculation of matrix elements of Coulomb interaction and two-centre
    exchange integrals needed in atomic collision codes, are
    described. The method makes the developing and debugging of such
    algorithms easier and faster.",
  paper = "Kocb96.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Koep96,
  author = "Koepf, Wolfram",
  title = {{Closed form Laurent-Puiseux series of algebraic functions}},
  journal = "Appl. Algebra Eng. Commun. Comput.",
  volume = "7",
  number = "1",
  pages = "21-26",
  year = "1996",
  abstract =
    "Let $y$ be an algebraic function defined by a polynomial equation
    $P(x,y)=0$ whose coefficients are polynomials in $x$ over a field $K$
    which may be one of the fields $\mathbb{Q}$, $\mathbb{R}$, or
    $\mathbb{C}$.  D. V. and G. V. Chudnovsky [J. Complexity 2, 271-294
    (1986; Zbl 0629.68038); ibid. 3, 1-25 (1987; Zbl 0656.34003)] describe
    a pair of algorithms to calculate the coefficients in the
    Laurent-Puiseux developments of the branches of $y$: The first
    algorithm returns a linear differential equation
    \[q_n(x)y^{(n)} + y_{n-1}(x)y^{(n-1)}+\cdots+q_1(x)y^{'}+q_0(x)y=0\]
    which is satisfied by all branches of $y$ and whose coefficients are
    polynomials in $x$ over $K$, the other uses this differential equation
    to get a linear recurrence relation for the Puiseux coefficients. The
    author used this algorithms (the second in a simpler version) to
    calculate the recurrence relation; if this relation contains only two
    terms, an algorithm found by the author returns an explicit formula
    for the Puiseux coefficients [J. Symb. Comp. 13, 581-603 (1992; Zbl
    0758.30026)].  In this paper, the author gives examples to illustrate
    his algorithms and to show that for many algebraic functions defined
    by polynomials of low degree a closed form of their Puiseux
    coefficients may be calculated. He points out that on the other side
    the complexity of the resulting recurrence equation may be extremely
    high even for an algebraic function defined by a sparse polynomial of
    low degree.",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@InProceedings{Koep99,
  author = "Koepf, Wolfram",
  title = {{Orthogonal polnomials and computer algebra}},
  booktitle = "Recent developments in complex analysis and computer algebra",
  series = "ISSAC 97",
  year = "1999",
  publisher = "Kluwer Adademic Publishers",
  location = "Newark, DE",
  pages = "205-234",
  abstract =
    "Orthogonal polynomials have a long history, and are still important
    objects of consideration in mathematical research as well as in
    applications in Mathematical Physics, Chemistry, and
    Engineering. Quite a lot is known about them. Particularly well-known
    are differential equations, recurrence equations, Rodrigues formulas,
    generating functions and hypergeometric representations for the
    classical systems of Jacobi, Laguerre and Hermite which can be found
    in mathematical dictionaries. Less well-known are the corresponding
    representations for the classical discrete systems of Hahn,
    Krawtchouk, Meixner and Charlier, as well as addition theorems,
    connection relations between different systems and other identities
    for these and other systems of orthogonal polynomials. The ongoing
    research in this still very active subject of mathematics expands the
    knowledge database about orthogonal polynomials continuously. In the
    last few decades the classical families have been extended to a rather
    large collection of polynomial systems, the so-called Askey-Wilson
    scheme, and they have been generalized in other ways as well.

    Recently new algorithmic approaches have been discovered to compute
    differential, recurrence and similar equations from series or integral
    representations. These methods turn out to be quite useful to prove or
    detect identities for orthogonal polynomial systems. Further
    algorithms to detect connection coefficients or to identify polynomial
    systems from given recurrence equations have been developed. Although
    some algorithmic methods had been known already in the last century,
    their use was rather limited due to the immense amount of
    calculations. Only the existence and distribution of computer algebra
    systems makes their use simple and useful for everybody.

    In this plenary lecture an overview is given of how algorithmic
    methods implemented in computer algebra systems can be used to prove
    identities about and to detect new knowledge for orthogonal
    polynomials and other hypergeometric type special functions.
    Implementations for this type of algorithms exist in Maple,
    Mathematica and REDUCE, and maybe also in other computer algebra
    systems. Online demonstrations will be given using Maple V.5.",
  paper = "Koep99.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Koep99a,
  author = "Koepf, Wolfram",
  title = {{Software for the algorithmic work with orthogonal polynomials
           and special functions}},
  journal = "Electron. Trans. Numer. Anal.",
  volume = "9",
  year = "1999",
  link = "\url{http://arxiv.org/pdf/math/9809125v1.pdf}",
  abstract =
    "An overview of the MAPLE routines that can be used for hypergeometric
    and basic hypergeometric series with some discussion of how and why
    they work.",
  paper = "Koep99a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Koep14,
  author = "Koepf, Wolfram",
  title = {{Methods of Computer Algebra for Orthogonal Polynomials}},
  year = "2014",
  location = "Rutgers, NJ, USA",
  link = "\url{http://www.mathematik.uni-kassel.de/~koepf/Vortrag/2014-Zeilberger-Vortrag.pdf}",
  video1 = "https://vimeo.com/85573338",
  video2 = "https://vimeo.com/85573712",
  website = "http://www.caop.org",
  paper = "Koep14.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Kosl91,
  author = "Koseleff, P.-V.",
  title = {{Word games in free Lie algebras: several bases and formulas}},
  journal = "Theoretical Computer Science",
  volume = "79",
  number = "1",
  pages = "241-256",
  year = "1991",
  abstract =
    "The author compares the efficiency of many methods which allow
    calculations in Lie algebras. Many construction methods exist for the
    base of free Lie algebras developed from finite sets. They use two
    algorithms for calculation of several Campbell-Hausdorf formulas.
    Diverse implementations are realised in LISP on Scratchpad II",
```

```
  keywords = "axiomref",
  beebe = "Koseleff:1991:WGF"
}
```

———-

— **Koseleff:1991:WGF** —

```
@Article{Koseleff:1991:WGF,
  author =       "P.-V. Koseleff",
  title =        {{Word games in free Lie algebras: several bases and
                 formulas}},
  journal =      j-THEOR-COMP-SCI,
  volume =       "79",
  number =       "1",
  pages =        "241--256",
  month =        feb,
  year =         "1991",
  CODEN =        "TCSCDI",
  ISSN =         "0304-3975 (print), 1879-2294 (electronic)",
  ISSN-L =       "0304-3975",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The author compares the efficiency of many methods
                 which allow calculations in Lie algebras. Many
                 construction methods exist for the base of free Lie
                 algebras developed from finite sets. They use two
                 algorithms for calculation of several
                 Campbell--Hausdorf formulas. Diverse implementations
                 are realised in LISP on Scratchpad II.",
  acknowledgement = ack-nhfb,
  affiliation =  "IBM, Paris, France",
  classification = "C6130 (Data handling techniques); C7310
                 (Mathematics)",
  fjournal =     "Theoretical Computer Science",
  link =  "\url{http://www.sciencedirect.com/science/journal/03043975}",
  keywords =     "Bases; Campbell--Hausdorf formulas; Finite sets; Free
                 Lie algebras; LISP; Scratchpad II",
  language =     "English",
  pubcountry =   "Netherlands",
  thesaurus =    "Mathematics computing; Symbol manipulation",
}
```

———-

— **axiom.bib** —

```
@article{Kred90,
  author = "Kredel, Heinz",
  title = {{MAS Modula-2 Algebra System}},
  journal = "LNCS",
  volume = "429",
  pages = "270-271",
```

```
  year = "1990",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Kred08,
  author = "Kredel, Heinz",
  title = {{On a Java computer algebra system,
           its performance and applications}},
  journal = "Sci. Comput. Program.",
  volume = "70",
  number = "2-3",
  pages = "185-207",
  year = "2008",
  link = "\url{http://ac.els-cdn.com/S0167642307001736/1-s2.0-S0167642307001736-main.pdf}",
  abstract =
    "This paper considers Java as an implementation language for a
    starting part of a computer algebra library. It describes a design of
    basic arithmetic and multivariate polynomial interfaces and classes
    which are then employed in advanced parallel and distributed Groebner
    base algorithms and applications. The library is type-safe due to its
    design with Java's generic type parameters and thread-safe using
    Java's concurrent programming facilities. We report on the performance
    of the polynomial arithmetic and on applications built upon the core
    library.",
  paper = "Kred08.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kred07,
  author = "Kredel, Heinz",
  title = {{Evaluation of a Java computer algebra system}},
  booktitle = "Computer Mathematics, 8th Asian symposium",
  series = "ASCM 2007",
  year = "2007",
  isbn = "978-3-540-87826-1",
  location = "Singapore",
  pages = "121-138",
  link = "\url{http://krum.rz.uni-mannheim.de/kredel/oocas-casc2010-slides.pdf}",
  abstract =
    "This paper evaluates the suitability of Java as an implementation
    language for the foundations of a computer algebra library. The design
    of basic arithmetic and multivariate polynomial interfaces and classes
    have been presented. The library is type-safe due to its design with
    Java's generic type parameters and thread-safe using Java's concurrent
```

```
    programming facilities. We evaluate some key points of our library and
    differences to other computer algebra systems.",
  paper = "Kred07.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Kred10,
  author = "Kredel, Heinz and Jolly, Raphael",
  title = {{Generic, type-safe and object oriented computer algebra software}},
  booktitle = "Proc. 12th International Workshop",
  series = "CASC 2010",
  year = "2010",
  isbn = "978-3-642-15273-3",
  location = "Berlin",
  pages = "162-177",
  link = "\url{http://krum.rz.uni-mannheim.de/kredel/oocas-casc2010-slides.pdf}",
  abstract =
    "Advances in computer science, in particular object oriented
    programming, and software engineering have had little practical impact
    on computer algebra systems in the last 30 years. The software design
    of existing systems is still dominated by ad-hoc memory management,
    weakly typed algorithm libraries and proprietary domain specific
    interactive expression interpreters. We discuss a modular approach to
    computer algebra software: usage of state-of-the-art memory management
    and run-time systems (e.g. JVM) usage of strongly typed, generic,
    object oriented programming languages (e.g. Java) and usage of general
    purpose, dynamic interactive expression interpreters (e.g. Python). To
    illuatrate the workability of this approach, we have implemented and
    studied computer algebra systems in Java and Scala. In this paper we
    report on the current state of this work by presenting new examples.",
  paper = "Kred10.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Kred11,
  author = "Kredel, Heinz",
  title = {{Unique factorization domains in the Java computer algebra system}},
  year = "2011",
  publisher = "Springer",
  booktitle = "Automated deduction in geometry (ADG 2008)",
  pages = "86-115",
  isbn = "978-3-642-21045-7",
  abstract =
    "This paper describes the implementation of recursive algorithms in
```

```
    unique factorization domains, namely multivariate polynomial greatest
    common divisors (gcd) and factorization into irreducible parts in the
    Java computer algebra library (JAS). The implementation of gcds,
    resultants and factorization is part of the essential building blocks
    for any computation in algebraic geometry, in particular in automated
    deduction in geometry. There are various implementations of these
    algorithms in procedural programming languages. Our aim is an
    implementation in a modern object oriented language with generic data
    types, as it is provided by Java programming language. We exemplify
    that the type design and implementation of JAS is suitable for the
    implementation of several greatest common divisor algorithms and
    factorization of multivariate polynomials. Due to the design we can
    employ this package in very general settings not commonly seen in other
    computer algebra systems. As for example, in the coefficient
    arithmetic for advanced Groebner basis computations like in polynomial
    rings over rational function fields or (finite, commutative) regular
    rings. The new package provides factory methods for the selection of
    one of the several implementations for non experts. Further we
    introduce a parallel proxy for gcd implementations which runs
    different implementations concurrently.",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@InProceedings{Kred11a,
  author = "Kredel, Heinz and Jolly, Raphael",
  title = {{Algebraic structures as typed objects}},
  booktitle = "Proc. 13th International Workshop",
  series = "CASC 2011",
  year = "2011",
  isbn = "978-3-642-23567-2",
  location = "Berlin",
  pages = "294-308",
  link = "\url{http://krum.rz.uni-mannheim.de/kredel/to-cas-casc2011-slides.pdf}",
  abstract =
    "Following the research direction of strongly typed, generic, object
    oriented computer algebra software, we examine the modeling of
    algebraic structures as typed objects in this paper. We discuss the
    design and implementation of algebraic and transcendental extension
    fields together with the modeling of real algebraic and complex
    algebraic extension fields. We will show that the modeling of the
    relation between algebraic and real algebraic extension fields using
    the delegation design concept has advantages over the modeling as
    sub-types using sub-class implementation. We further present a summary
    of design problems, which we have encountered so far with our
    implementation in Java and present possbile solutions in Scala.",
  paper = "Kred11a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Krei85,
  author = "Kreisel, G.",
  title = {{Proof Theory and the Synthesis of Programs: Potential and
            Limitations}},
  booktitle = "EUROCAL 85",
  publisher = "Springer",
  paper = "Krei85.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Kreu14,
  author = "Kreuzer, Edwin",
  title = {{Computerized Symbolic Manipulation in Mechanics}},
  publisher = "Springer",
  year = "2014",
  abstract =
    "The aim of this book is to present important software tools, basic
    concepts, methods, and highly sophisticated applications of
    computerized symbolic manipulation to mechanics problems. An overview
    about general-purpose symbolic software is followed by general
    guidelines how to develop and implement high-quality computer algebra
    code. The theoretical background including modeling techniques for
    mechanical systems is provided which allows for the computer aided
    generation of the symbolic equation of motion for multibody
    systems. It is shown how the governing equations for different types
    of problems in structural mechanics can be automatically derived and
    how to implement finite element techniques via computer algebra
    software. Perturbation methods as a very powerful approach for
    nonlinear problems are discussed in detail and are demonstrated for a
    number of applications. The applications covered in this book
    represent some of the most advanced topics in the rapidly growing
    field of research on symbolic computation."
}
```

———-

— **axiom.bib** —

```
@misc{Krip94,
  author = "Kripfganz, Jochen and Perlt, Holger",
  title = {{Working with Mathematica. An Introduction with examples}},
  comment = "Arbeiten mit Mathematica. Eine Einfuhrung mit Beispielen",
  book = "Hander",
  year = "1994",
  keywords = "axiomref"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Kuma00,
  author = "Kumar, P. and Pellegrino, S.",
  title = {{Computation of kinematic paths and bifurcation points}},
  journal = "Int. J. Solids Struct.",
  volume = "37",
  number = "46-47",
  pages = "7003-7027",
  year = "2000",
  abstract =
    "This article deals with the kinematic simulation of movable
    structures that go through special configurations of kinematic
    bifurcation, as they move. A series of algorithms are developed for
    structures that can be modelled using pin-jointed bars and that admit
    a single-parameter motion. These algorithms are able to detect and
    locate any bifurcation points that exist along the path of the
    structure and, at each bifurcation point, can determine all possible
    motions of the structure. The theory behind the algorithms is
    explained, and the analysis of a simple example is discussed in
    detail. Then, a simplified version of the particular problem that had
    motivated this work, the simulation of the folding and deployment of a
    thin membrane structure forming a solar sail, is analysed. For the
    particular cases that are considered, it is found that the entire
    process is inextensional, but a detailed study of the simulation
    results shows that in more general cases, it is likely that stretching
    or wrinkling will occur.",
  paper = "Kuma00.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kusc89,
  author = "Kusche, K. and Kutzler, B. and Mayr, H.",
  title = {{Implementation of a geometry theorem proving package
           in SCRATCHPAD II}},
  booktitle = "Proc. of Eurocal '87",
  series = "Lecture Notes in Computer Science 378",
  pages = "246-257",
  isbn = "3-540-51517-8",
  year = "1987",
  abstract =
    "The problem of automatically proving geometric theorems has gained a
    lot of attention in the last two years. Following the general approach
    of translating a given geometric theorem into an algebraic one,
```

```
    various powerful provers based on characteristic sets and Groebner
    bases have been implemented by groups at Academia Sinica Bejing
    (China), U. Texas at Austin (USA), General Electric Schenectady (USA),
    and Research Institute for Symbolic Computation Linz (Austria). So ar,
    fair comparisons of the various provers were not possible, because the
    underlying hardware and the underlying algebra systems differed
    greatly. This paper reports on the first uniform implementation of all
    of these provers in the computer algebra system and language
    SCRATCHPAD II. We summarize the recent achievements in the area of
    automated geometry theorem proving, shortly review the SCRATCHPAD II
    system, describe the implementation of the geometry theorem proving
    package, and finally give a computing time statistics of 24 examples.",
  keywords = "axiomref",
  beebe = "Kusche:1989:IGT"
}
```

———-

— Kusche:1989:IGT —

```
@InProceedings{Kusche:1989:IGT,
  author =       "K. Kusche and B. Kutzler and H. Mayr",
  title =        {{Implementation of a geometry theorem proving package
                 in SCRATCHPAD II}},
  crossref =     "Davenport:1989:EEC",
  pages =        "246--257",
  month =        "",
  year =         "1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The problem of automatically proving geometric
                 theorems has gained a lot of attention in the last two
                 years. Following the general approach of translating a
                 given geometric theorem into an algebraic one, various
                 powerful provers based on characteristic sets and
                 Gr{\"o}bner bases have been implemented by groups at
                 Academia Sinica Beijing (China), U. Texas at Austin
                 (USA), General Electric Schenectady (USA), and Research
                 Institute for Symbolic Computation Linz (Austria). So
                 far, fair comparisons of the various provers were not
                 possible, because the underlying hardware and the
                 underlying algebra systems differed greatly. This paper
                 reports on the first uniform implementation of all
                 these provers in the computer algebra system and
                 language SCRATCHPAD II. The authors summarize the
                 recent achievements in the area of automated geometry
                 theorem proving, shortly review the SCRATCHPAD II
                 system, describe the implementation of the geometry
                 theorem proving package, and finally give computing
                 time statistics of 24 examples.",
  acknowledgement = ack-nhfb,
  affiliation =  "Res. Inst. for Symbolic Comput., RISC-LINZ, Johannes
                 Kepler Univ., Linz, Austria",
```

```
  classification = "C1230 (Artificial intelligence); C7310
                   (Mathematics)",
  keywords =       "Geometry theorem proving package; SCRATCHPAD II;
                   Characteristic sets; Gr{\"o}bner bases; Computer
                   algebra system; Computing time statistics",
  language =       "English",
  thesaurus =      "Algebra; Computational geometry; Mathematics
                   computing; Symbol manipulation; Theorem proving",
}
```

———-

1.40.12 L

— **ignore** —

```
\bibitem[Lahey 08]{Lah08} Lahey, Tim
  title = {{Sage Integration Testing}},
  link = "\url{http://github.com/tjl/sage_int_testing}",
  year = "2008",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Lamb03a,
  author = "Lamban, Laureano and Pascual, Vico and Rubio, Julio",
  title = {{An object-oriented interpretation of the EAT system}},
  journal = "Appl. Algebra Eng. Commun. Comput.",
  volume = "14",
  number = "3",
  year = "2003",
  pages = "187-215",
  abstract =
    "In a previous paper we characterized, in the category theory setting,
    a class of implementations of abstract data types, which has been
    suggested by the way of programming in the EAT system. (EAT, Effective
    Algebraic Topology, is one of Sergeraerts systems for effective
    homology and homotopy computation.) This characterization was
    established using classical tools, in an unrelated way to the current
    mainstream topics in the field of algebraic specifications. Looking
    for a connection with these topics, we have found, rather
    unexpectedly, that our approach is related to some object-oriented
    formalisms, namely hidden specifications and the coalgebraic view. In
    this paper, we explore these relations making explicit the implicit
    object-oriented features of the EAT system and generalizing the data
    structure analysis we had previously done.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Lamb89,
  author = "Lambe, Larry A.",
  title = {{Scratchpad II as a tool for mathematical research}},
  journal = "Notices of the AMS",
  year = "1989",
  pages = "143-147",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Lamb91,
  author = "Lambe, Larry A.",
  title = {{Resolutions via homological perturbation}},
  journal = "Journal of Symbolic Computation",
  volume = "12",
  number = "1",
  pages = "71-87",
  year = "1991",
  abstract =
    "The purpose of this paper is to review an algorithm for computing
    ``small'' resolutions in homological algebra, to provide examples of
    its use as promised in [L1], [LS], and to illustrate the use of
    computer algebra in an area not usually associated with that
    subject. Comparison of the complexes produced by the method discussed
    here with those produced by other methods shows that the algorithm
    generalizes several other approaches, [GL], [GLS1], [GLS2], [BL], [BL2].

    This is an expository note which is intended to help make homological
    perturbation theory more accesible and to encourage wider use of
    Computer Algebra in mathematical research.

    The class of objects presented here -- Finitely generated torsion-free
    nilpotent groups (of arbitrary nilpotency class) -- are given because
    of their simplicity. The examples point to the general phenomena that
    are to be expected when trying to derive complexes smaller than
    ``standard complexes'' in other homological contexts. The complexes
    produced are generally {\sl much} smaller than the bar construction, but
    larger than a {\sl minimal resolution}.",
  paper = "Lamb91.pdf",
  keywords = "axiomref",
  beebe = "Lambe:1991:RHP"
}
```

———-

— **Lambe:1991:RHP** —

```
@Article{Lambe:1991:RHP,
  author =       "L. A. Lambe",
  title =        {{Resolutions via homological perturbation}},
  journal =      j-J-SYMBOLIC-COMP,
  volume =       "12",
  number =       "1",
  pages =        "71--87",
  month =        jul,
  year =         "1991",
  CODEN =        "JSYCEH",
  ISSN =         "0747-7171 (print), 1095-855X (electronic)",
  ISSN-L =       "0747-7171",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "There is a trade-off between the size of the
                 resolutions which arise from the perturbation method
                 and the complexity of the new differential. In order to
                 keep the modules relatively small, there is a
                 considerable increase in the algebraic complexity of
                 the resulting differentials. In order to study such
                 complexes systematically, examples are needed. To
                 facilitate such study, the Scratchpad system was used
                 to set up and perform the necessary calculations.
                 Because of the way Scratchpad is organized, this could
                 be done in a way that minimizes programming effort and
                 provides the natural mathematical environment for such
                 calculations. The author discusses some of the general
                 theory behind homological perturbation theory, gives an
                 idea of what is needed to make calculations within that
                 theory in Scratchpad, and calculates a resolution of
                 the integers over the integral group ring of the 4*4
                 upper triangular matrices with ones along the
                 diagonal.",
  acknowledgement = ack-nhfb,
  affiliation =  "Illinois Univ., Chicago, IL, USA",
  classification = "C4240 (Programming and algorithm theory); C7310
                 (Mathematics)",
  fjournal =     "Journal of Symbolic Computation",
  link =  "\url{http://www.sciencedirect.com/science/journal/07477171}",
  keywords =     "Algebraic complexity; Complexity; Homological
                 perturbation; Integers; Mathematical environment;
                 Resolutions; Scratchpad system",
  language =     "English",
  pubcountry =   "UK",
  thesaurus =    "Computational complexity; Perturbation theory; Symbol
                 manipulation",
}
```

———-

— **axiom.bib** —

```
@article{Lamb92,
  author = "Lambe, Larry",
  title = {{Next Generation Computer Algebra Systems AXIOM and the
           Scratchpad Concept: Applications to Research in Algebra}},
  publisher = "21st Nordic Congress of Mathematicians",
  journal = "unknown",
  year = "1992",
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/axiom-21cong.pdf}",
  algebra = "\newline\refto{category FMCAT FreeModuleCat}",
  abstract =
    "One way in which mathematicians deal with infinite amounts of data is
    symbolic representation. A simple example is the quadratic equation
    \[x = \frac{-b\pm\sqrt{b^2-4ac}}{2a}\]
    a formula which uses symbolic representation to describe the solutions
    to an infinite class of equations. Most computer algebra systems can
    deal with polynomials with symbolic coefficients, but what if symbolic
    exponents are called for (e.g. $1+t^i$)? What if symbolic limits on
    summations are also called for, for example
    \[1+t+\ldots+t^i=\sum_j{t^j}\]

    The ``Scratchpad Concept'' is a theoretical ideal which allows the
    implementation of objects at this level of abstraction and beyond in a
    mathematically consistent way. The Axiom computer algebra system is an
    implementation of a major part of the Scratchpad Concept.  Axiom
    (formerly called Scratchpad) is a language with extensible
    parameterized types and generic operators which is based on the
    notions of domains and categories. By examining some aspects of the
    Axiom system, the Scratchpad Concept will be illustrated. It will be
    shown how some complex problems in homologicial algebra were solved
    through the use of this system.",
  paper = "Lamb92.pdf",
  keywords = "axiomref, printed",
  beebe = "Lambe:1994:NGC"
}
```

———-

— **Lambe:1994:NGC** —

```
@InProceedings{Lambe:1994:NGC,
  author =       "Larry Lambe",
  editor =       "Mats Gyllenberg and Lars Erik Persson",
  booktitle =    "{Analysis, algebra, and computers in mathematical
                 research: proceedings of the Twenty-first Nordic
                 Congress of Mathematicians, Lule{\aa} University of
                 Technology, Sweden, 1992}",
  title =        {{Next generation computer algebra systems AXIOM and
                 the Scratchpad concept: Applications to research in
                 algebra}},
  volume =       "156",
  publisher =    pub-DEKKER,
  address =      pub-DEKKER:adr,
```

```
  pages =        "201--222",
  year =         "1994",
  ISBN =         "0-8247-9217-3",
  ISBN-13 =      "978-0-8247-9217-6",
  LCCN =         "QA299.6 .N67 1992",
  MRclass =      "18-04 (Machine computation, programs (category
                 theory)) 68W30 (Symbolic computation and algebraic
                 computation) 20-04 (Machine computation, programs
                 (group theory)) 18G15 (Ext and Tor, generalizations)
                 18G35 (Chain complexes (homological algebra)) 55U15
                 (Chain complexes) 20J05 (Homological methods in group
                 theory) 16E40 (Homology and cohomology theories for
                 assoc. rings)",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  series =       "Lecture Notes in Pure and Applied Mathematics",
  link =          "\url{http://www.loc.gov/catdir/enhancements/fy0647/94002464-d.html}",
  ZMnumber =     "0832.18001",
  abstract =     "One way in which mathematicians deal with infinite
                 amounts of data is symbolic representation. A simple
                 example is the quadratic equation $ x = {- b \pm \sqrt
                 {b^2 - 4a} c \over 2a} $, a formula which uses symbolic
                 representation to describe the solutions to an infinite
                 class of equations. Most computer algebra systems can
                 deal with polynomials with symbolic coefficients, but
                 what if symbolic exponents are called for (e.g., $ 1 +
                 t^i) $ ? What if symbolic limits on summations are also
                 called for (e.g., $ 1 + t + \cdots + t^i = \sum_j t^j)
                 $ ? The ``Scratchpad concept'' is a theoretical ideal
                 which allows the implementation of objects at this
                 level of abstraction and beyond in a mathematically
                 consistent way. The AXIOM computer algebra system is an
                 implementation of a major part of the Scratchpad
                 concept. AXIOM (formerly called Scratchpad) is a
                 language with extensible parameterized types and
                 generic operators which is based on the notion of
                 domains and categories [{\it R. D. Jenks} and {\it R.
                 S. Sutor}, Axiom. The scientific computation system,
                 Springer, Berlin etc. (1992; Zbl 0758.68010)]. By
                 examining some aspects of the AXIOM system, the
                 Scratchpad concept will be illustrated. It will be
                 shown how some complex problems in homological algebra
                 [cf. the author, Contemp. Math. 134, 183-218 (1992; Zbl
                 0798.16028), J. Pure Appl. Algebra 84, No. 3, 311-329
                 (1993; Zbl 0766.55015)] were solved through the use of
                 this system.",
  bookpages =    "ix + 408",
  keywords =     "AXIOM; bar construction; computer algebra; domains;
                 Ext; generic operators; parameterized types;
                 perturbation lemma; Scratchpad; symbolic exponents;
                 symbolic limits; Tor",
  language =     "English",
}
```

———-

— **ignore** —

```
\bibitem[Lambe 93]{Lam93} Lambe, Larry
  title = {{On Using Axiom to Generate Code}},
(preprint) 1993
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Lamb93a,
  author = "Lambe, Larry and Luczak, Richard",
  title = {{Object-Oriented Mathematical Programming and
           Symbolic/Numeric Interface}},
  journal = "3rd Int. Conf. on Expert Systems in Numerical Computing",
  year = "1993",
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/axiom-fem.pdf}",
  abstract =
    "The Axiom language is based on the notions of ``categories'',
    ``domains'', and ``packages''. These concepts are used to build an
    interface between symbolic and numeric calculations. In particular, an
    interface to the NAG Fortran Library and Axiom's algebra and graphics
    facilities is presented. Some examples of numerical calculations in a
    symbolic computational environment are also included using the finite
    element method. While the examples are elementary, we believe that
    they point to very powerful methods for combining numeric and symbolic
    computational techniques.",
  paper = "Lamb93a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Lamb93b,
  author = "Lambe, Larry A. and Radford, David E.",
  title = {{Algebraic Aspects of the Quantum Yang-Baxter Equation}},
  journal = "Journal of Algebra",
  volume = "154",
  pages = "228-288",
  year = "1993",
  link = "\url{pages.bangor.ac.uk/~mas019/papers/lamrad.pdf}",
  abstract =
    "In this paper we examine a variety of algebraic contexts in which the
    quantum Yang-Baxter equation arises, and derive methods for generating
    new solutions from given ones. The solutions we describe are encoded
    in objects which have a module and a comodule structure over a
    bialgebra. Our work here is based in part on the ideas of [DR1,DR2].",
```

```
  paper = "Lamb93b.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@book{Lamb97,
  author = "Lambe, Larry A. and Radford, David E.",
  title = {{Introduction to the quantum Yang-Baxter equation and quantum
           groups: an algebraic approach}},
  booktitle = "Mathematics and its Applications",
  publisher = "Kluwer Adademic Publishers",
  year = "1997",
  abstract =
    "The quantum Yang-Baxter equation (QYBE) has roots in statistical
    mechanics and the inverse scattering method and leads to a natural
    construction of a bialgebra. It turns out to have important
    connections with knot theory and invariants of 3-manifolds. There are
    now available many reference books to quantum groups and these various
    applications. The book under review develops the algebraic
    underpinning and theory of the QYBE, including the constant form and
    the one and two parameter forms.

    We give a brief description of the chapters. Chapter 1 (together with
    an Appendix) gives the algebraic preliminaries involving coalgebras,
    bialgebras, Hopf algebras, modules and comodules. Chapter 2 introduces
    the various forms of the QYBE, and the basic algebraic structures
    associated to them, including Faddeev-Reshetikhin-Takhtadzhan (FRT)
    construction. Chapter 3 explores various categorical settings for the
    constant form of the QYBE, the most basic being the category of left
    QYB modules over a bialgebra and the notion of algebras, coalgebras,
    etc. in this category. Chapter 4 develops universal mapping properties
    of the FRT construction and its reduced version, and the authors
    investigate when the reduced FRT construction leads to a pointed
    bialgebra or a pointed Hopf algebra. Chapter 5 develops the quantum
    groups associated to $SL(2)$, i.e., the quantum universal enveloping
    algebra, and the quantum function algebra. Chapter 6 introduces
    quasitriangular Hopf algebras, and discusses how the
    finite-dimensional ones give rise to solutions of the QYBE through
    their representation theory. The most important example is the
    Drinfeld double of a finite-dimensional Hopf algebra. The authors note
    (through an exercise!) that every finite-dimensional Hopf algebra is
    the reduced FRT construction of some solution to the QYBE. Chapter 7
    introduces coquasitriangular bialgebras, the most important being the
    FRT and the reduced FRT constructions. There are some generalizations
    here to the one-parameter form of the QYBE. Chapter 8 uses all the
    previously developed techniques to find solutions of the QYBE in
    certain cases, including the one-parameter form. Some of these were
    discovered by computer algebra methods. The final chapter 9 gives a
    brief discussion of certain categorical constructions and the QYBE is
    certain fairly abstract categories, motivated by the fact that the FRT
```

```
    construction is a coend.

    This book fills an important niche in the literature involving the
    QYBE by highlighting the algebraic aspects and applications. Although
    this is basically a reference book, it includes so many important
    parts of the study of Hopf algebras that it could be used as a
    textbook for a certain type of course on Hopf algebras and quantum
    groups, and certainly as supplementary reading material for such a
    course. There are frequent exercises which would be useful for such
    purposes. Besides being a basic source book, the authors include some
    new results and some novel approaches to earlier results. All this
    makes this book a most welcome addition to the quantum group
    literature.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Lamb02,
  author = "Lambe, Larry A. and Seiler, Werner M.",
  title = {{Differential equations, Spencer cohomology, and computing
           resolutions}},
  journal = "Georgian Mathematical Journal",
  volume = "9",
  number = "4",
  pages = "723-774",
  year = "2002",
  link = "\url{https://www.emis.de/journals/GMJ/vol9/v9n4-11.pdf}",
  abstract =
    "Spencer cohomology is one of the classical tools in the study of
    overdetermined systems of partial differential equations. The paper
    proposes a new point of view of the Spencer cohomology based on a dual
    approach via comodules that allows to relate the Spencer cohomology to
    standard constructions in homological algebra, and discuss concrete
    methods for its construction based on homological perturbation
    theory. It also gives a detailed introduction to the subject, and
    proposes a new algorithm of constructing the Spencer resolution by
    using symbolic computation systems.",
  paper = "Lamb02.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Lamb03,
  author = "Lambe, Larry A. and Luczak, Richard and Nehrbass, John W.",
  title = {{A New Finite Difference Method for the Helmholtz Equation
           Using Symbolic Computation}},
```

```
  journal = "Int. J. Comp. Eng. Sci.",
  volume = "4",
  year = "2003",
  link = "\url{http://pages.bangor.ac.uk.~mas019/papers/lln.pdf}",
  abstract =
    "A new finite difference method for the Helmholtz equation is
    presented. The method involves replacing the standard ``weights'' in the
    central difference quotients (Secs. 2.1, 2.2, and 2.3) by weights that
    are optimal in a sense that will be explained in the sections just
    mentioned. The calculation of the optimal weights involves some
    complicated and error-prone manipulations of integral formulas that is
    best done using computer-aided symbolic computation (SC). In addition,
    we discuss the important problem of interpolation involving meshes
    that have been refined in certain subregions. Analytic formulae are
    derived using SC for these interpolation schemes. Our results are
    discussed in Sec. 5. Some hints about the computer methods we used to
    accomplish these results are given in the Appendix. More information
    is available and access to that information is referenced.

    While we do not want to make SC the focus of this work, we also do not
    want to underestimate its value. Armed with robust and efficient SC
    libraries, a researcher can {\sl comfortably} and {\sl conveniently}
    experiment with ideas that he or she might not examine otherwise.",
  paper = "Lamb03.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Laza93,
  author = "Lazard, Daniel",
  title = {{On the representation of rigid-body motions and its application
           to generalized platform manipulators}},
  booktitle = "Proc. Workshop Computational Kinematics",
  year = "1993",
  location = "Dagstuhl Castle, Germany",
  publisher = "Kluwer Academic Publishers",
  pages = "175-181",
  abstract =
    "Different ways for representing rigid body motions (direct isometries)
    by a computer are presented. It turns out that the choice between them
    may have a dramatic effect on the difficulty of a computation or of a
    proof. As an application, a computational proof is given of the fact
    that the direct kinematic problem for the generalized Stewart platform
    has at most 40 complex solutions.",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Lebedev 08]{Leb08} Lebedev, Yuri
  title = {{OpenMath Library for Computing on Riemann Surfaces}},
PhD thesis, Nov 2008 Florida State University
  link = "\url{http://www.math.fsu.edu/~ylebedev/research/HyperbolicGeometry.html}",
  keywords = "axiomref"
```

————-

— **axiom.bib** —

```
@inproceedings{LeBl91,
  author = "LeBlanc, S.E.",
  title = {{The use of MathCAD and Theorist in the ChE classroom}},
  booktitle = "Proc. ASEE Annual Meeting",
  year = "1991",
  pages = "287-299",
  abstract =
    "MathCAD and Theorist are two powerful mathematical packages available
    for instruction in the ChE classroom. MathCAD is advertised as an
    'electronic scratchpad' and it certainly lives up to its billing.  It
    is an extremely user-friendly collection of numerical routines that
    eliminates the drudgery of solving many of the types of problems
    encountered by undergraduate ChE's (and engineers in general). MathCAD
    is available for both the Macintosh and IBM PC compatibles. The PC
    version is available as a full-functioned student version for around
    US\$40 (less than many textbooks). Theorist is a symbolic mathematical
    package for the Macintosh. Many interesting and instructive things can
    be done with it in the ChE curriculum. One of its many attractive
    features includes the ability to generate high quality three
    dimensional plots that can be very instructive in examining the
    behavior of an engineering system. The author discusses the
    application and use of these packages in chemical engineering and give
    example problems and their solutions for a number of courses including
    stoichiometry, unit operations, thermodynamics and design.",
  keywords = "axiomref",
  beebe = "LeBlanc:1991:UMT"
}
```

————-

— **LeBlanc:1991:UMT** —

```
@InProceedings{LeBlanc:1991:UMT,
  author =       "S. E. LeBlanc",
  title =        {{The use of MathCAD and Theorist in the ChE
                 classroom}},
  crossref =     "Anonymous:1991:PAC",
  pages =        "287--299 (vol. 1)",
  year =         "1991",
  bibdate =      "Tue Sep 17 06:37:45 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
```

```
  abstract =     "MathCAD and Theorist are two powerful mathematical
                 packages available for instruction in the ChE
                 classroom. MathCAD is advertised as an 'electronic
                 scratchpad' and it certainly lives up to its billing.
                 It is an extremely user-friendly collection of
                 numerical routines that eliminates the drudgery of
                 solving many of the types of problems encountered by
                 undergraduate ChE's (and engineers in general). MathCAD
                 is available for both the Macintosh and IBM PC
                 compatibles. The PC version is available as a
                 full-functioned student version for around US\$40 (less
                 than many textbooks). Theorist is a symbolic
                 mathematical package for the Macintosh. Many
                 interesting and instructive things can be done with it
                 in the ChE curriculum. One of its many attractive
                 features includes the ability to generate high quality
                 three dimensional plots that can be very instructive in
                 examining the behavior of an engineering system. The
                 author discusses the application and use of these
                 packages in chemical engineering and give example
                 problems and their solutions for a number of courses
                 including stoichiometry, unit operations,
                 thermodynamics and design.",
  acknowledgement = ack-nhfb,
  affiliation =  "Toledo Univ., OH, USA",
  classification = "C7450 (Chemical engineering); C7810C (Computer-aided
                 instruction)",
  keywords =     "Chemical engineering; MathCAD; Mathematical packages;
                 Numerical routines; Stoichiometry; Symbolic
                 mathematical package; Theorist; Thermodynamics; Unit
                 operations",
  language =     "English",
  thesaurus =    "Chemical engineering computing; Computer aided
                 instruction; Spreadsheet programs; Symbol
                 manipulation",
}
```

———-

— axiom.bib —

```
@article{Lece02,
  author = "Lecerf, Gregoire",
  title = {{Quadratic Newton iteration for systems with multiplicity}},
  journal = "Found. Comput. Math.",
  volume = "2",
  number = "3",
  pages = "247-293",
  year = "2002",
  abstract =
    "The author proposes an efficient iterator with quadratic convergence
    that generalizes Newton iterator for multiple roots. It is based on an
    $m$-adic topology where the ideal $m$ can be chosen generic
```

```
    enough. Compared to the Newton iterator the proposed iterator
    introduces a small overhead that grows with the square of the
    multiplicity of the root.",
  paper = "Lece02.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Lece10,
  author = "Lecerf, Gregoire",
  title = {{Mathemagix: Toward Large Scale Programming for Symbolic and
            Certified Numeric Computations}},
  booktitle = "Mathematical software",
  series = "ICMS 2010",
  year = "2010",
  isbn = "978-3-642-15581-9",
  location = "Berlin",
  pages = "329-332",
  abstract =
    "Coordinated by Joris van der Hoeven from the 90's, the Mathemagix
    project aims at the design of a scientific programming language for
    symbolic and certified numeric algorithms. This language can be
    compiled and interpreted, and it features a strong type system with
    classes and categories. Several C++ libraries are also being
    developed, mainly with Bernard Mourrain and Philippe Trebuchet, for
    the elementary operations with polynomials, power series and matrices,
    with a special care towards efficiency and numeric stability.

    In my talk I will give an overview of the language, of the design and
    the contents of the C++ libraries, and I will illustrate possibilities
    offered for certified numeric computations with balls and intervals.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Lece96,
  author = "Lecerf, Gregoire",
  title = {{Dynamic Evaluation and Real Closure Implementation in Axiom}},
  year = "1996",
  link = "\url{http://lecerf.perso.math.cnrs.fr/software/drc/drc.ps}",
  paper = "Lece96.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **ignore** —

```
\bibitem[Lecerf 96a]{Le96a} Lecerf, Gr\'egoire
  title = {{The Dynamic Real Closure implemented in Axiom}},
  link = "\url{http://lecerf.perso.math.cnrs.fr/software/drc/drc.ps}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Leeu94,
  author = "van Leeuwen, Andre M.A.",
  title = {{LiE, A software package for Lie group computations}},
  journal = "Euromath Bulletin",
  volume = "1",
  number = "2",
  year = "1994",
  abstract =
    "A description is given of LiE, a specialized computer algebra package
    for computations concerning Lie groups and algebras, and their
    representations. The functionality of the package is demonstrated by
    sample computations, and the structure of the program and the
    algorithms implemented in it are discussed.",
  paper = "Leeu94.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Leti97,
  author = "Letichevskij, A. Alexander and Marinchenko, V. G.",
  title = {{Objects in algebraic programming system}},
  journal = "Cybern. Syst. Anal.",
  volume = "33",
  number = "2",
  pages = "283-299",
  year = "1997",
  comment = "translated from Russian",
  abstract =
    "The algebraic programming system (APS) developed at the
    V. M. Glushkov Institute of Cybernetics of the Academy of Sciences of
    the Ukrainian SSR integrates the basic programming paradigms,
    including procedural, functional, algebraic, and logic programming.

    Algebraic programming in APS relies on special data structures, the
    so-called graph terms, which permit using diverse data and knowledge
    representations in relevant application domains. In the language
    APLAN, graph terms are described by expressions or systems of
    expressions of a many-sorted algebra of data. They may represent both
```

```
    objects of the application domain and reasoning about these
    objects. The option of setting an arbitrary interpretation of the
    operations in the algebra of data makes it possible to use APS as a
    basis for various extensions.

    Symbolic computation systems such as Scratchpad/AXIOM have acquired
    special importance. They provide various possibilities of manipulating
    typed mathematical objects, including objects of complex hierarchical
    structure. This is a natural requirement when working with algebraic
    objects. In particular, the properties of many algebraic structures
    (such as groups, rings, fields, etc.) are naturally
    hierarchical-modular.

    The Institute of Cybernetics and the Kherson Teachers College have
    developed an instruction-oriented computer algebra system AIST. The
    AIST kernel is a hierarchical structure of mathematical concepts
    described in the APS language. However, construction of new
    applications on the basis of this hierarchical structure has proved
    difficult. The system kernel can be made more flexible by providing
    tools for flexible description of hierarchical structures of
    mathematical concepts.

    In this article, we describe an extension of the language APLAN, which
    provides tools for the object-oriented style of programming. This is
    one of the possible ways of introducing types in APS. The
    object-oriented technology also can be used to develop a hierarchical
    system of mathematical objects.",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Levelt 95]{Lev95} Levelt, A. H. M. (ed)
ISSAC '95: Proceedings of the 1995 International
Symposium on Symbolic and Algebraic Computation: July 10-12, 1995, Montreal,
Canada ISSAC-PROCEEDINGS-1995. ACM Press, New York, NY 10036, USA, 1995
ISBN 0-89791-699-9 LCCN QA76.95 I59 1995 ACM order number 505950
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Lewi99,
  author = "Lewis, Robert H. and Wester, Michael",
  title = {{Comparison of polynomial-orienged computer algebra systems}},
  journal = "SIGSAM Bulletin",
  volume = "33",
  number = "4",
  pages = "5-13",
```

```
  year = "1999",
  link = "\url{https://home.bway.net/lewis/cacomp.ps}",
  abstract =
    "Exact symbolic computation with polynomials and matrices over
    polynomial rings has wide applicability to many fields [Hereman96,
    Lewis99]. By ``exact symbolic'', we mean computation with polynomials
    whose coefficients are integers (of any size), rational numbers, or
    from finite fields, as opposed to coefficients that are ``floats'' of a
    certain precision. Such computation is part of most computer algebra
    (CA) systems. Over the last dozen years, several large CA systems have
    become widely available, such as Axiom, Derive, Macsyma, Maple,
    Mathematica and Reduce. They tend to have great breadth, be produced
    by profit-making companies, and be relatively expensive, at least for
    a full blown non-student version. However, most if not all of these
    systems have difficulty computing with the polynomials and matrices
    that arise in actual research. Real problems tend to produce large
    polynomials and large matrices that the general CA systems cannot
    handle [Lewis99].

    In the last few years, several smaller CA systems focused on
    polynomials have been produced at universities by individual
    researchers or small teams. They run on Macs, PCs and workstations.
    They are freeware or shareware. Several claim to be much more
    efficient than the large systems at exact polynomial computations. The
    list of these systems includes CoCoA, Fermat, MuPAD, Pari-Gp and
    Singular [CoCoA, Fermat, MuPAD, Pari-Gp, Singular].

    In this paper, we compare these small systems to each other and to two
    of the large systems (Magma and Maple) on a set of problems involving
    exact symbolic computation with polynomials and matrices. The problems
    here involve:
    \begin{itemize}
    \item the ground rings $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/p$
    and other finite fields
    \item basic arithmetic of polynomials over the ground ring
    \item basic arithmetic of rational functions over the ground ring
    \item polynomial evaluation (substitution)
    \item matrix normal form
    \item determinants and characteristic polynomial
    \item GCDs of multivariate polynomial
    \item resultants
    \end{itemize}",
  paper = "Lewi99.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Lewi17,
  author = "Lewis, Robert Y.",
  title = {{An Extensible Ad Hoc Interface between Lean and Mathematica}},
```

```
  journal = "EPTCS",
  volume = "262",
  pages = "23-37",
  year = "2017",
  abstract =
    "We implement a user-extensible ad hoc connection between the Lean
    proof assistant and the computer algebra system Mathematica. By
    reflecting the syntax of each system in the other and providing a
    flexible interface for extending translation, our connection allows
    for the exchange of arbitrary information between the two systems. We
    show how to make use of the Lean metaprogramming framework to verify
    certain Mathematica computations, so that the rigor of the proof
    assistant is not compromised.",
  paper = "Lewi17.pdf",
  keywords = "axiomref, printed"
}
```

———-

— axiom.bib —

```
@inproceedings{Liao95,
  author = "Liao, Hsin-Chao and Fateman, Richard J.",
  title = {{Evaluation of the heuristic polynomial GCD}},
  booktitle = "Proc. ISSAC 1995",
  year = "1995",
  pages = "240-247",
  link = "\url{http://www.cs.berkeley.edu/~/fateman/papers/phil8.ps}",
  abstract =
    "The Heuristic Polynomial GCD procedure (GCDHEU) is used by the Maple
    computer algebra system, but not other. Because Maple has an
    especially efficient kernel that provides fast integer arithmetic, but
    a relatively slower interpreter for non-kernel code, the GCDHEU
    routine is especially effective in that it moves much of the
    computation into ``bignum'' arithmetic and hence executes primarily in
    the kernel.

    We speculated that in other computer algebra systems an implementation
    GCDHEU would not be advantageous. In particular, if {\sl all} the
    system code is compiled to run at ``full speed'' in a (presumably more
    bulky) kernel that is entirely written in C or compiled Lisp, then
    there would seem to be no point in recasting the polynomial gcd
    problem into a bignum GCD problem. Manipulating polynomials that are
    vectors of coefficients would seem to be equivalent computationally to
    manipulating vectors of big digits.

    Yet our evidence suggests that one can take advantage of the GCDHEU in
    a Lisp system as well. Given a good implementation of bignums, for
    most small problems and many large ones, a substantial speedup can be
    obtained by the appropriate choice of GCD algorithm, including often
    enought, the GCDHEU approach. Another major winner seem to be the
    subresultant polynomial remainder sequence algorithm. Because more
    sophisticated sparse algorithms are relatively slow on small problems
```

```
    and only occasionally emerge as superior (on larger problems) it seems
    the choice of a fast GCD algorithm is tricky.",
  paper = "Liao95.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Lixx05,
  author = "Li, Xin",
  title = {{Efficient Management of Symbolic Computation with Polynomials}},
  school = "University of Western Ontario",
  year = "2005",
  link = "\url{http://www.csd.uwo.ca/~moreno//Publications/XinLi-MSThesis-2005.pdf.gz}",
  abstract =
    "Symbolic polynomial computation is widely used to solve many applied
    or abstract mathematical problems. Some of them, such as solving
    systems of polynomial equations, have exponential complexity. Their
    implementation is, therefore, a challenging task.

    By using adapted data structures, asymptotically fast algorithms and
    effective code optimization techniques, we show how to reduce the
    practical and theoretical complexity of these computations. Our effort
    is divided into three categories: integrting the best known techniques
    into our implementation, investigating new directions, and measuring
    the interactions between numerous techniques.

    We chose AXIOM and ALDOR as our implementation and experimentation
    environment, since they are both strongly typed and highly efficient
    Computer Algebra Systems (CAS). Our implementation results show that
    our methods have great potential to improve the efficiency of exact
    polynomial computations with the selected CASs. The performance of our
    implementation is comparable to that of (often outperforming) the best
    available packages for polynomial computations.",
  paper = "Lixx05.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Lixx06,
  author = "Li, Xin and Moreno Maza, Marc",
  title = {{Efficient implementation of polynomial arithmetic in a
           multiple-level programming environment}},
  booktitle = "Mathematical Software",
  series = "ICMS 2006",
  year = "2006",
  isbn = "978-3-540-38084-9",
  location = "Spain",
```

```
  pages = "12-23",
  link = "\url{http://www.math.kobe-u.ac.jp/icms2006/icms2006-video/slides/046.pdf}",
  abstract =
    "The purpose of this study is to investigate implementation techniques
    for polynomial arithmetic in a multiple-level programming
    environment. Indeed, certain polynomial data types and algorithms can
    further take advantage of the features of lower level languages, such
    as their specialized data structures or direct access to machine
    arithmetic. Whereas, other polynomial operations, like Groebner basis
    over an arbitrary field, are suitable for generic programming in a
    high-level language.

    We are interested in the integration of polynomial data type
    implementations realized at different language levels, such as Lisp, C
    and Assembly. In particular, we consider situations for which code
    from different levels can be combined together within the same
    application in order to achieve high-performance.  We have developed
    implementation techniques in the multiple-level programming
    environment provided by the computer algebra system AXIOM. For a given
    algorithm realizing a polynomial operation, available at the user
    level, we combine the strengths of each language level and the
    features of a specific machine architecture. Our experimentations show
    that this allows us to improve performances of this operation in a
    significant manner.",
  paper = "Lixx06.pdf",
  keywords = "axiomref",
  beebe = "Li:2006:EIP"
}
```

———-

— Li:2006:EIP —

```
@Article{Li:2006:EIP,
  author =       "Xin Li and Moreno Maza",
  title =        {{Efficient Implementation of Polynomial Arithmetic in a
                 Multiple-Level Programming Environment}},
  journal =      j-LECT-NOTES-COMP-SCI,
  volume =       "4151",
  pages =        "12--23",
  year =         "2006",
  CODEN =        "LNCSD9",
  ISBN =         "3-540-38084-1",
  ISBN-13 =      "978-3-540-38084-9",
  ISSN =         "0302-9743 (print), 1611-3349 (electronic)",
  ISSN-L =       "0302-9743",
  bibdate =      "Mon Apr 19 08:40:16 2010",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "Proceedings of the International Congress of
                 Mathematical Software (ICMS 2006).",
  link =          "\url{http://www.csd.uwo.ca/~moreno//Publications/Li-MorenoMaza-ICMS-06.pdf}",
  acknowledgement = ack-nhfb,
  fjournal =     "Lecture Notes in Computer Science",
```

```
  keywords =     "Axiom symbolic-algebra system",
}
```

———-

— **axiom.bib** —

```
@inproceedings{Lixx07,
  author = "Li, Xin and Maza, Marc Moreno and Schost, Eric",
  title = {{On the Virtues of Generic Programming for Symbolic Computation}},
  booktitle = "Computational Science -- ICCS 2007",
  series = "Lecture Notes in Computer Science",
  isbn = "3-540-72585-7",
  pages = "251-258",
  year = "2007",
  abstract =
    "The purpose of this study is to measure the impact of C level code
    polynomial arithmetic on the performances of Axiom high-level algorithms,
    such as polynomial factorization. More precisely, given a high-level
    Axiom package P parameterized by a univariate polynomial domain U, we
    have compared the performances of P when applied to different U's,
    including an Axiom wrapper for our C level code.

    Our experiments show that when P relies on U for its univariate
    polynomial computations, our specialized C level code can provide a
    significant speed-up. For instance, the improved implementation of
    square-free factorization in Axiom is 7 times faster than the one in
    Maple and very close to the one in Magma. On the contrary, when P does
    not rely much on the operations of U and implements its private univariate
    polynomial operation, the P cannot benefit from our highly optimized C
    level code. Consequently, code which is poorly generic reduces the
    speed-up opportunities when applied to highly efficient and specialized.",
  keywords = "axiomref",
  paper = "Lixx07.pdf",
  beebe = "Li:2007:VGP"
}
```

———-

— **Li:2007:VGP** —

```
@InProceedings{Li:2007:VGP,
  author =       "Xin Li and Marc Moreno Maza and {\'E}ric Schost",
  title =        {{On the Virtues of Generic Programming for Symbolic
                 Computation}},
  crossref =     "Shi:2007:CSIb",
  pages =        "251--258",
  year =         "2007",
  DOI =          "http://dx.doi.org/10.1007/978-3-540-72586-2_35",
  bibdate =      "Tue Aug 12 10:36:21 2014",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib;
                 http://www.math.utah.edu/pub/tex/bib/magma.bib;
                 http://www.math.utah.edu/pub/tex/bib/maple-extract.bib",
```

```
  acknowledgement = ack-nhfb,
  keywords =      "Axiom; Magma; Maple",
}
```

———-

— **axiom.bib** —

```
@phdthesis{Lixx09a,
  author = "Li, Xin",
  title = {{Toward High-Performance Polynomial System Solvers Based On
           Triangluar Decompositions}},
  school = "University of Western Ontario",
  year = "2009",
  link = "\url{http://www.csd.uwo.ca/~moreno/Publications/XinLiPhDThesis-2008.pdf}",
  abstract =
    "This thesis is devoted to the design and implementation of polynomial
    system solvers based on symbolic computation. Solving systems of
    non-linear, algebraic or differential equations, is a fundamental
    problem in mathematical sciences. It has been studied for centuries
    and still stimulates many research developments, in particular on the
    front of high-performance computing.

    Triangular decompositions are a highly promising technique with the
    potential to produce high-performance polynomial system solvers. This
    thesis makes several contributions to this effort.

    We propose asymptotically fast algorithms for the core operati ons on
    which triangular decompositions rely. Complexity results and comparati
    ve implementation show that these new algorithms provide substantial
    performance improvements.

    We present a fundamental software library for polynomial arithmetic in
    order to support the implementation of high-performance solvers based
    on triangular decompositions. We investigate strategies for the
    integration of this library in high-level programming environments
    where triangular decompositions are usually implemented.

    We obtain a high performance library combining highly optimized C
    routines and solving procedures written in the Maple computer algebra
    system. The experimental result shows that our approaches are very
    effective, since our code often outperforms pre-existing solvers in a
    significant manner.",
  paper = "Lixx09a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Lixx09,
```

```
  author = "Li, Xin and Moreno Maza, Marc and Schost, Eric",
  title = {{Fast arithmetic for triangular sets: from theory to practice}},
  booktitle = "Proc 32nd ISSAC",
  series = "ISSAC 2007",
  year = "2007",
  isbn = "978-1-59593-743-8",
  location = "Canada",
  pages = "269-276",
  link = "\url{http://www.csd.uwo.ca/~moreno/Publications/LiMorenoSchost-ISSAC-2007.pdf}",
  abstract =
    "We study arithmetic operations for triangular families of
    polynomials, concentrating on multiplication in dimension zero. By a
    suitable extension of fast univariate Euclidean division, we obtain
    theoretical and practical improvements over a direct recursive
    approach; for a family of special cases, we reach quasi-linear
    complexity. The main outcome we have in mind is the acceleration of
    higher-level algorithms, by interfacing our low-level implemention
    with languages such as AXIOM or Maple. We show the potential for hugh
    speed-ups, by comparing two AXIOM implementations of vanHoeij and
    Monagan's modular GCD algorithm.",
  paper = "Lixx09.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Li 10]{YL10} Li, Yue; Dos Reis, Gabriel
  title = {{A Quantitative Study of Reductions in Algebraic Libraries}},
PASCO 2010
  link = "\url{http://www.axiomatics.org/~gdr/concurrency/quant-pasco10.pdf}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Lixx11,
  author = "Li, Yue and Dos Reis, Gabriel",
  title = {{An Automatic Parallelization Framework for Algebraic
           Computation Systems}},
  booktitle = "Proc. ISSAC 2011",
  pages = "233-240",
  isbn = "978-1-4503-0675-1",
  year = "2011",
  link = "\url{http://www.axiomatics.org/~gdr/concurrency/oa-conc-issac11.pdf}",
  abstract =
    "This paper proposes a non-intrusive automatic parallelization
    framework for typeful and property-aware computer algebra systems.
    Automatic parallelization remains a promising computer program
    transformation for exploiting ubiquitous concurrency facilities
```

```
    available in modern computers. The framework uses semantics-based
    static analysis to extract reductions in library components based on
    algebraic properties. An early implementation shows up to 5 times
    speed-up for library functions and homotopy-based polynomial system
    solver. The general framework is applicable to algebraic computation
    systems and programming languages with advanced type systems that
    support user-defined axioms or annotation systems.",
  paper = "Lixx11.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Ligatsikas 96]{Liga96} Ligatsikas, Zenon; Rioboo, Renaud;
Roy, Marie Francoise
  title = {{Generic computation of the real closure of an ordered field}},
Math. and Computers in Simulation 42 pp 541-549 (1996)
  abstract = "
    This paper describes a generalization of the real closure computation
    of an ordered field (Rioboo, 1991) enabling to use different technques
    to code a single real algebraic number.",
  paper = "Liga96.pdf",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Linton 93]{Lin93} Linton, Steve
  title = {{Vector Enumeration Programs, version 3.04}},
  link = "\url{http://www.cs.st-andrews.ac.uk/~sal/nme/nme_toc.html#SEC1}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Lint98,
  author = "Linton, Stephen",
  title = {{The GAP 4 Type System Organising Algebraic Algorithms}},
  link = "\url{http://www.gap-system.org/Doc/Talks/kobe.ps}",
  paper = "Lint98.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Linu15,
  author = "Linux Links",
  title = {{Axiom}},
  link = "\url{http://www.linuxlinks.com/article/20080810064748970/Axiom.html}",
  year = "2015",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Lisk99,
  author = "Liska, Richard and Drska, Ladislav and Limpouch, Jiri and
            Sinor, Milan and Wester, Michael and Winkler, Franz",
  title = {{Computer Algebra - Algorithms, Systems and Applications}},
  year = "1999",
  publisher = "web",
  link =
    "\url{https://people.eecs.berkeley.edu/~fateman/282/readings/liska.pdf}",
  paper = "Lisk99.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Lomo02,
  author = "Lomonaco, Samual J. and Kauffman, Louis H.",
  title = {{Quantum hidden subgroup algorithms: a mathematical perspective}},
  booktitle = "Quantum computation and information",
  series = "AMS special session",
  year = "2002",
  isbn = "0-8218-2140-7",
  location = "Washington",
  pages = "139-202",
  link = "\url{https://arxiv.org/pdf/quant-ph/0201095v3.pdf}",
  abstract =
    "The ultimate objective of this paper is to create a stepping stone to
    the development of new quantum algorithms. The strategy chosen is to
    begin by focusing on the class of abelian quantum hidden subgroup
    algorithms, i.e., the class of abelian algorithms of the Shor/Simon
    genre. Our strategy is to make this class of algorithms as
    mathematically transparent as possible. By the phrase ``mathematically
    transparent'' we mean to expose, to bring to the surface, and to make
    explicit the concealed mathematical structures that are inherently and
    fundamentally a part of such algorithms. In so doing, we create
    symbolic abelian quantum hidden subgroup algorithms that are analogous
    to the those symbolic algorithms found within such software packages
    as Axiom, Cayley, Maple, Mathematica, and Magma.
```

```
    As a spin-off of this effort, we create three different
    generalizations of Shors quantum factoring algorithm to free abelian
    groups of finite rank. We refer to these algorithms as wandering (or
    vintage $\mathbb{Z}_Q$) Shor algorithms. They are essentially quantum
    algorithms on free abelian groups of finite rank $n$ which, with each
    iteration, first select a random cyclic direct summand $\mathbb{Z}$ of
    the group and then apply one iteration of the standard Shor algorithm
    to produce a random character of the approximating finite group
    $\widetilde{A} = \mathbb{Z}_Q$, called the group probe. These
    characters are then in turn used to find either the order $P$ of a
    maximal cyclic subgroup $\mathbb{Z}_P$ of the hidden quotient group
    $H_{\varphi}$, or the entire hidden quotient group $H_{\varphi}$. An
    integral part of these wandering quantum algorithms is the selection
    of a very special random transversal $t_{\mu}:\widetilde{A}\rightarrow A$,
    which we refer to as a Shor transversal. The algorithmic time
    complexity of the first of these wandering Shor algorithms is found to
    be $O(n^2({\rm lg\ } Q)^3 ({\rm lg\ }{\rm lg\ } Q)^{n+1})$.",
  paper = "Lomo02.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@inproceedings{Luck86,
  author = "Lucks, Michael",
  title = {{A fast implementation of polynomial factorization}},
  booktitle = "Proc. 1986 Symposium on Symbolic and Algebraic Computation",
  series = "SYMSAC '86",
  year = "1986",
  location = "Waterloo, Ontario",
  pages = "228-232",
  publisher = "ACM Press",
  isbn = "0-89791-199-7",
  doi = "http://dx.doi.org/10.1145/32439.32485",
  abstract =
    "A new package for factoring polynomials with integer coefficients is
    described which yields significant improvements over previous
    implementations in both time and space requirements. For multivariate
    problems, the package features an inexpensive method for early
    detection and correction of spurious factors. This essentially solves
    the multivariate extraneous factor problem and eliminates the need to
    factor more than one univariate image, except in rare cases. Also
    included is an improved technique for coefficient prediction which is
    successful more frequently than prior versions at short-circuiting the
    expensive multivariate Hensel lifting stage. In addition some new
    approaches are discussed for the univariate case as well as for the
    problem of finding good integer substitution values. The package has
    been implemented both in Scratchpad II and in an experimental version
    of muMATH.",
  keywords = "axiomref",
  beebe = "Lucks:1986:FIP"
```

```
}
```

———-

— **Lucks:1986:FIP** —

```
@InProceedings{Lucks:1986:FIP,
  author =       "Michael Lucks",
  editor =       "Bruce W. Char",
  booktitle =    "Proceedings of the 1986 Symposium on Symbolic and
                 Algebraic Computation: Symsac '86, July 21--23, 1986,
                 Waterloo, Ontario",
  title =        {{A fast implementation of polynomial factorization}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "228--232",
  year =         "1986",
  DOI =          "http://dx.doi.org/10.1145.32485",
  ISBN =         "0-89791-199-7",
  ISBN-13 =      "978-0-89791-199-3",
  LCCN =         "QA155.7.E4 A281 1986",
  bibdate =      "Thu Jul 26 09:06:12 2001",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "ACM order number 505860.",
  acknowledgement = ack-nhfb,
  keywords =     "Scratchpad",
}
```

———-

— **axiom.bib** —

```
@mastersthesis{Luek77,
  author = "Lueken, E.",
  title = {{Ueberlegungen zur Implementierung eines
           Formelmanipulationssystems}},
  school = {Technischen Universit{\"{a}}t Carolo-Wilhelmina zu Braunschweig},
  address = "Braunschweig, Germany",
  year = "1977",
  keywords = "axiomref",
  beebe = "Lueken:1977:UIF"
}
```

———-

— **Lueken:1977:UIF** —

```
@MastersThesis{Lueken:1977:UIF,
  author =       "E. Lueken",
  title =        {{Ueberlegungen zur Implementierung eines
                 Formelmanipulationssystemes}},
  school =       "Technischen Universit{\"{a}}t Carolo-Wilhelmina zu
```

```
                Braunschweig",
  address =     "Braunschweig, Germany",
  pages =       "??",
  year =        "1977",
  bibsource =   "/usr/local/src/bib/bibliography/Misc/TUBScsd.bib;
                http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  descriptor =  "Alpak, Altran, Formac, Funktion, G.g.t., Kanonische
                Darstellung von Polynomen, Macsyma, Mathlab, Polynom,
                Rationale Funktion, Reduce, Sac-1, Scratchpad",
}
```

———-

— axiom.bib —

```
@article{Lync91,
  author = "Lynch, R. and Mavromatis, H. A.",
  title = {{New quantum mechanical perturbation technique using an
           'electronic scratchpad' on an inexpensive computer}},
  journal = "American Journal of Pyhsics",
  volume = "59",
  number = "3",
  pages = "270-273",
  year = "1991",
  abstract =
    "The authors have developed a new method for doing numerical quantum
    mechanical perturbation theory. It has the flavor of
    RayleighSchrdinger perturbation theory (division of the Hamiltonian
    into an unperturbed Hamiltonian and a perturbing term, use of the
    basis formed by the eigenfunctions of the unperturbed Hamiltonian)
    while turning out to be a variational technique. Furthermore, it is
    easily implemented by means of the widely used electronic
    scratchpad, MathCAD 2.0, using an inexpensive computer. As an
    example of the method, the problem of a harmonic oscillator with a
    quartic perturbing term is examined.",
  keywords = "axiomref",
  beebe = "Lynch:1991:NQM"
}
```

———-

— Lynch:1991:NQM —

```
@Article{Lynch:1991:NQM,
  author =      "R. Lynch and H. A. Mavromatis",
  title =       {{New quantum mechanical perturbation technique using an
                'electronic scratchpad' on an inexpensive computer}},
  journal =     j-AMER-J-PHYSICS,
  volume =      "59",
  number =      "3",
  pages =       "270--273",
  month =       mar,
  year =        "1991",
```

```
  CODEN =        "AJPIAS",
  ISSN =         "0002-9505 (print), 1943-2909 (electronic)",
  ISSN-L =       "0002-9505",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The authors have developed a new method for doing
                 numerical quantum mechanical perturbation theory. It
                 has the flavor of Rayleigh--Schr{\"o}dinger
                 perturbation theory (division of the Hamiltonian into
                 an unperturbed Hamiltonian and a perturbing term, use
                 of the basis formed by the eigenfunctions of the
                 unperturbed Hamiltonian) while turning out to be a
                 variational technique. Furthermore, it is easily
                 implemented by means of the widely used 'electronic
                 scratchpad,' MathCAD 2.0, using an inexpensive
                 computer. As an example of the method, the problem of a
                 harmonic oscillator with a quartic perturbing term is
                 examined.",
  acknowledgement = ack-nhfb,
  affiliation =  "Dept. of Phys., King Fahd Univ. of Pet. and Miner.,
                 Dhahran, Saudi Arabia",
  classification = "A0150H (Instructional computer use); A0210 (Algebra,
                 set theory, and graph theory); A0230 (Function theory,
                 analysis); A0365D (Functional analytical methods);
                 A0365F (Algebraic methods); A0365G (Solutions of wave
                 equations: bound state); C7810C (Computer-aided
                 instruction)",
  fjournal =     "American Journal of Physics",
  keywords =     "Electronic scratchpad; Eigenvalues; Eigenfunctions;
                 Quantum mechanical perturbation technique;
                 Rayleigh--Schr{\"o}dinger perturbation theory;
                 Hamiltonian; Variational technique; MathCAD 2.0;
                 Harmonic oscillator",
  language =     "English",
  pubcountry =   "USA",
  thesaurus =    "Computer aided instruction; Eigenvalues and
                 eigenfunctions; Harmonic oscillators; Perturbation
                 theory; Quantum theory; Variational techniques",
}
```

———-

1.40.13 M

— **axiom.bib** —

```
@inproceedings{Mahb05,
  author = "Mahboubi, Assia",
  title = {{Programming and certifying the CAD algorithm inside the
           coq system}},
  year = "2005",
  booktitle = "Mathematics, Algorithms, Proofs, volume 05021 of Dagstuhl
```

```
               Seminar Proceedings, Schloss Dagstuhl (2005)",
  abstract =
    "A. Tarski has shown in 1975 that one can perform quantifier
    elimination in the theory of real closed fields. The introduction of
    the Cylindrical Algebraic Decomposition (CAD) method has later allowed
    to design rather feasible algorithms. Our aim is to program a
    reflectional decision procedure for the Coq system, using the CAD, to
    decide whether a (possibly multivariate) system of polynomial
    inequalities with rational coefficients has a solution or not. We have
    therefore implemented various computer algebra tools like gcd
    computations, subresultant polynomial or Bernstein polynomials.",
  paper = "Mahb05.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Math89,
  author = "Mathews, J.",
  title = {{Symbolic computational algebra applied to Picard iteration}},
  journal = "Mathematics and computer education",
  volume = "23",
  number = "2",
  pages = "117-122",
  year = "1989",
  link = "\url{http://mathfaculty.fullerton.edu/mathews/articles/1989PicardIteration.pdf}",
  abstract =
    "The term ‘‘Picard iteration’’ occurs two places in undergraduate
    mathematics. In numerical analysis it is used when discussing fixed
    point iteration for finding a numerical approximation to the equation
    $s=g(x)$. In differential equations, Picard iteration is a
    constructive procedure for establishing the existence of a solution to
    a differential equation $y^{\prime} = f(x,y)$.

    The first type of Picard iteration uses computations to generate a
    sequence of numbers which converges to a solution. We will not present
    this application, but mention that it involves the traditional role of
    the computer as a ‘‘number cruncher.’’

    The second application of Picard iteration illustrates how to use a
    computer to generate a sequence of functions which converges to a
    solution. The purpose of this article is to show the step by step
    process in translating mathematical theory into the symbolic
    manipulation setting. Systems such as MACSYM, ALTRAN, REDUCE, SMP,
    MAPLE, SCRATCHPAD and muMATH are being introduced in undergraduate
    mathematics courses to assist in keeping trace of equations during
    complicated manipulations.",
  paper = "Math89.pdf",
  keywords = "axiomref",
  beebe = "Mathews:1989:SCA"
}
```

———-

— **Mathews:1989:SCA** —

```
@Article{Mathews:1989:SCA,
  author =       "J. Mathews",
  title =        {{Symbolic computational algebra applied to {Picard}
                 iteration}},
  journal =      j-MATH-COMP-EDU,
  volume =       "23",
  number =       "2",
  pages =        "117--122",
  month =        "Spring",
  year =         "1989",
  CODEN =        "MCEDDA",
  ISSN =         "0730-8639",
  bibdate =      "Tue Sep 17 06:48:10 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Picard iteration occurs in differential equations as a
                 constructive procedure for establishing the existence
                 of a solution to a differential equation. This
                 application of Picard iteration illustrates how to use
                 a computer to generate a sequence of functions which
                 converges to a solution. The article shows the step by
                 step process in translating mathematical theory into
                 the symbolic manipulation setting. Systems such as
                 MACSYMA, ALTRAN, REDUCE, SMP, MAPLE, SCRATCHPAD, and
                 muMATH are being introduced in undergraduate
                 mathematics courses to assist in keeping track of
                 equations during complicated manipulations. The product
                 muMATH is illustrated because of its availability. It
                 runs on all 16-bit computers which are IBM compatible.
                 The way has been opened to see how computers can be
                 used as a symbol cruncher.",
  acknowledgement = ack-nhfb,
  affiliation =  "California State Univ., Fullerton, CA, USA",
  classification = "C4130 (Interpolation and function approximation);
                 C4170 (Differential equations); C6130 (Data handling
                 techniques); C7310 (Mathematics)",
  fjournal =     "Mathematics and computer education",
  keywords =     "Differential equations; IBM compatible; Mathematical
                 theory; Mathematics computing; MuMATH; Picard
                 iteration; Symbol cruncher; Symbolic manipulation;
                 Undergraduate mathematics",
  language =     "English",
  pubcountry =   "USA",
  thesaurus =    "Differential equations; Iterative methods; Mathematics
                 computing; Microcomputer applications; Symbol
                 manipulation",
}
```

———-

— **axiom.bib** —

```
@misc{Maxi16,
  author = "Maxima",
  title = {{Other Free Computer Algebra Systems}},
  link = "\url{http://maxima.sourceforge.net/compalg.html}",
  year = "2016",
  abstract =
    "Axiom is a general purpose Computer Algebra system. It is useful for
    doing mathematics by computer and for research and development of
    mathematical algorithms. It defines a strongly typed, mathematically
    correct type hierarchy. It has a programming language and a built-in
    compiler.

    There is also an interesting Rosetta Stone which offers translations
    of many basic operations for several computer algebra systems,
    including Maxima.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Maza00,
  author = "Maza, Marc Moreno",
  title = {{On Triangular Decompositions of Algebraic Varieties}},
  institution = "Numerical Algorithms Group",
  year = "2000",
  month = "June",
  type = "technical report",
  number = "TR 4/99",
  link = "\url{http://www.csd.uwo.ca/~moreno//Publications}",
  algebra =
   "\newline\refto{category TSETCAT TriangularSetCategory}
    \newline\refto{category RSETCAT RegularTriangularSetCategory}
    \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
    \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
    \newline\refto{package RSDCMPK RegularSetDecompositionPackage}",
  abstract =
    "Different kinds of triangular decompositions of algebraic varieties
    are presented. The main result is an efficient method for obtaining
    them. Our strategy is based on a lifting theorem for polynomial
    computations module regular chains.",
  paper = "Maza00.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Maza06,
  author = "Maza, Marc Moreno",
  title = {{Axiom: Generic, open and powerful}},
  link = "\url{http://www.csd.uwo.ca/~moreno//Publications/ICMS-06-1.pdf}",
  year = "2006",
  paper = "Maza06.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Mcge10,
  author = "McGettrick, Michael",
  title = {{One dimensional quantum walks with memory}},
  journal = "Quantum Inf. Comput.",
  volume = "10",
  number = "5-6",
  pages = "509-524",
  year = "2010",
  link = "\url{https://arxiv.org/pdf/0911.1653v2.pdf}",
  abstract =
    "We investigate the quantum versions of a one-dimensional random walk,
    whose corresponding Markov chain is of order 2. This corresponds to
    the walk having a memory of one previous step. We derive the
    amplitudes and probabilities for these walks, and point out how they
    differ from bot classical random walks, and quantum walks without
    memory.",
  paper = "Mcge10.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[McJones 11]{McJ11} McJones, Paul
  title = {{Software Presentation Group -- Common Lisp family}},
  link = "\url{http://www.softwarepreservation.org/projects/LISP/common_lisp_family}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Mela90,
  author = "Melachrinoudis, E.; Rumpf, D. L.",
  title = {{Teaching advantages of transparent computer software -- MathCAD}},
  journal = "CoED",
  volume = "10",
```

```
  number = "1",
  pages = "71-76",
  year = "1990",
  abstract =
    "The case is presented for using mathematical scratchpad software,
    such as MathCAD, in undergraduate and graduate engineering
    courses. The pedagogical benefits, especially relative to the usual
    black box engineering software, are described. Several examples of
    student written projects are presented. The projects solve problems in
    operations research, control theory and statistical regression
    analysis.",
  keywords = "axiomref",
  beebe = "Melachrinoudis:1990:TAT"
}
```

———-

— Melachrinoudis:1990:TAT —

```
@Article{Melachrinoudis:1990:TAT,
  author =       "E. Melachrinoudis and D. L. Rumpf",
  title =        {{Teaching advantages of transparent computer software
                 --- MathCAD}},
  journal =      j-COED,
  volume =       "10",
  number =       "1",
  pages =        "71--76",
  month =        jan # "-" # mar,
  year =         "1990",
  CODEN =        "CWLJDP",
  ISSN =         "0736-8607",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The case is presented for using mathematical
                 scratchpad software, such as MathCAD, in undergraduate
                 and graduate engineering courses. The pedagogical
                 benefits, especially relative to the usual black box
                 engineering software, are described. Several examples
                 of student written projects are presented. The projects
                 solve problems in operations research, control theory
                 and statistical regression analysis.",
  acknowledgement = ack-nhfb,
  affiliation =  "Dept. of Ind. Eng., Northeastern Univ., Boston, MA,
                 USA",
  classification = "C7110 (Education); C7310 (Mathematics); C7400
                 (Engineering); C7810C (Computer-aided instruction)",
  fjournal =     "CoED",
  keywords =     "Black box engineering software; Control theory;
                 Graduate engineering courses; MathCAD; Mathematical
                 scratchpad software; Operations research; Pedagogical
                 benefits; Statistical regression analysis; Student
                 written projects; Transparent computer software;
                 Undergraduate",
```

```
  language =     "English",
  pubcountry =   "USA",
  thesaurus =    "CAD; Educational computing; Engineering computing;
                 Mathematics computing; Teaching",
}
```

———-

— **axiom.bib** —

```
@misc{Mele88,
  author = "Melenk, H. and Moller, H. M. and Neun, W.",
  title = {{On Groebner Bases Computation on a Supercomputer Using REDUCE}},
  link = "\url{https://opus4.kobv.de/opus4-zib/files/10/SC-88-02.pdf}",
  abstract =
    "Groebner bases are the main tool for solving systems of algebraic
    equations and some other problems in connection with polynomial ideals
    using Computer Algebra Systems. The procedure for the computation of
    Groebner bases in REDUCE 3.3 has been modified in order to solve more
    complicated algebraic systems of equations by some general
    improvements and by some tools based on the specific resources of the
    CRAY X-MP. We present this modification and illustrate it by examples.",
  paper = "Mele88.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Mich01,
  author = "Michler, Gerhard O.",
  title = {{The character values of multiplicity-free irreducible constituents
           of a transitive permutation representation}},
  journal = "Kyushu J. Math.",
  volume = "55",
  number = "1",
  pages = "75-106",
  year = "2001",
  link = "\url{https://www.jstage.jst.go.jp/article/kyushujm/55/1/55\_1\_75/\_pdf}",
  abstract =
    "Let $G$ be a finite group and $M$ a subgroup of $G$. Let the
    permutation character of $G$ on the set of right cosets of $M$ be
    denoted by $(1_M)^G$. Any ordinary irreducible constituent of
    $(1_M)^G$ with multiplicity one is called a multiplicity-free
    constituent of $(1_M)^G$. In the paper under review the author gives
    an efficient algorithm to compute the values of any multiplicity-free
    constituent of $(1_M)^G$. The character value is in terms of the
    double coset decomposition of $M$ and related concepts concerning
    intersection matrices. The author uses this algorithm to determine the
    values of the multiplicity-free constituents of $(1_C)^{J_1}$, where
    $J_1$ is the smallest Janko group of order 175560 and $C$ is the
```

```
    centralizer of an involution in $J_1$. Using these, he then is able to
    compute the character table of $J_1$ which is already known.",
  paper = "Mich01.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Mill95,
  author = "Miller, Bruce R.",
  title = {{An expression formatter for MACSYMA}},
  year = "1995",
  abstract =
    "A package for formatting algebraic expressions in MACSYA is described.
    It provides facilities for user-directed hierarchical structuring of
    expressions, as well as for directing simplifications to selected
    subexpressions. It emphasizes a semantic rther than syntactic description
    of the desired form. The package also provides utilities for obtaining
    efficiently the coefficients of polynomials, trigonometric sums and
    power series. Similar capabilities would be useful in other computer
    algebra systems.",
  paper = "Mill95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Mino07,
  author = "Minoiu, N. and Netto, M and Mammar, S",
  title = {{Assistance control based on a composite Lyapunov function for
           lane departure avoidance}},
  booktitle = "Proc. 15 Med. Conf. on Control \& Automation",
  year = "2007",
  abstract =
    "This paper presents a vehicle steering assistance designed to avoid
    lane departure during driver inattention periods. Activated for a
    driver loss of concentration during a lane keeping maneuver the
    steering assistance drives the vehicle back to the center of the
    lane. In order to ensure a vehicle trajectory as close as possible to
    the centerline, the control law has been developed based on invariant
    sets theory and on composite Lyapunov functions. The computation has
    been performed using LMI methods, which allow in addition imposing a
    maximum bound for the control steering angle.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Mioa90,
  author = "Miola, A.",
  title = {{Design and Implementation of Symbolic Computation Systems}},
  publisher = "Springer-Verlag",
  year = "1990",
  isbn = "0-387-52531-9",
  journal = "LNCS",
  volume = "429",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Miola 93]{Mio93} Miola, A. (ed)
  title = {{Design and Implementation of Symbolic Computation Systems}},
International Symposium DISCO '93 Gmunden, Austria, September 15-17, 1993:
Proceedings.
Springer-Verlag, Berlin, Germany / Heildelberg, Germany / London, UK / etc.,
1993 ISBN 3-540-57235-X LCCN QA76.9.S88I576 1993
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Mish87,
  author = "Mishra, Bhubaneswar and Yap, Chee",
  title = {{Notes on Groebner Basis}},
  year = "1987",
  link = "\url{https://people.eecs.berkeley.edu/~fateman/282/readings/mishra87note.pdf}",
  abstract =
    "We present a self-contained exposition of the theory of Groebner
     basis and its applications",
  paper = "Mish87.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Mish93,
  author = "Mishra, Bhubaneswar",
  title = {{Algorithmic Algebra}},
  publisher = "Springer-Verlag",
  series = "Texts and Monographs in Computer Sciences",
  year = "1993",
  abstract =
```

```
    "This book is based on a graduate course in computer science taught in
    1987. The following topics are covered: computational ideal theory,
    solving systems of polynomial equations, elimination theory, real
    algebra, as well as an introduction chapter and two chapters with the
    needed algebraic background. The book is self-contained and the proofs
    are given with many details.

    It is clear that this book is only an introduction to the topic and
    does not cover the many improvements that appeared in the last 7 years
    about for example the computation of Groebner basis, polynomial
    solving, multivariate resultants and algorithms in real
    algebra. Choices had to be made to keep the content of a reasonable
    size and the complexity issues are not considered.

    The choice of topics is excellent, there are many exercises and
    examples. It is a very useful book.",
  paer = "Mish93.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Miss05,
  author = "Missura, Stephan A.",
  title = {{Theories = Signatures + Propositions Used as Types}},
  booktitle = "Integrating Symbolic Mathematics and Artificial Intelligence",
  volume = "958",
  pages = "144-155",
  year = "2005",
  abstract =
    "Languages that distinguish between types and structures use explicit
    components for the carrier type(s) in structures. Examples are the
    function language Standard ML and most algebraic specification
    systems.  Hence, they have to use general sum types or signatures to
    give types to structures and be able to build, for instance, the
    algebraic hierarchy.",
  paper = "Miss05",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Miss94,
  author = "Missura, Stephan A. and Weber, Andreas",
  title = {{Using Commutativity Properties for Controlling Coercions}},
  link = "\url{http://cg.cs.uni-bonn.de/personal-pages/weber/publications/pdf/WeberA/MissuraWeber94a.pdf}",
  abstract =
    "This paper investigates some soundness conditions which have to be
```

```
    fulfilled in systems with coercions and generic operators. A result of
    Reynolds on unrestricted generic operators is extended to generic
    operators which obey certain constraints. We get natural conditions
    for such operators, which are expressed within the theoretic framework
    of category theory. However, in the context of computer algebra, there
    arise examples of coercions and generic operators which do not fulfil
    these conditions. We describe a framework -- relaxing the above
    conditions -- that allows distinguishing between cases of ambiguities
    which can be resolved in a quite natural sense and those which
    cannot. An algorithm is presented that detects such unresolvable
    ambiguities in expressions.",
  paper = "Miss94.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **ignore** —

```
\bibitem[Monagan 87]{Mon87} Monagan, Michael B.
  title = {{Support for Data Structures in Scratchpad II}},
in [Wit87], pp17-18
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Mona93,
  author = "Monagan, Michael B.",
  title = {{Gauss: a parameterized domain of computation system with
           support for signature functions}},
  booktitle = "Design and Implementation of Symbolic Computation Systems",
  series = "Lecture Notes in Computer Science 722",
  pages = "81-94",
  isbn = "3-540-57235-X",
  year = "1993",
  link = "\url{http://www.cecm.sfu.ca/~monaganm/papers/DISCO93.pdf}",
  abstract =
    "The fastest known algorithms in classical algebra make use of signature
    functions. That is, reducing computation with formulae to computing with
    the integers module $p$, by substituting random numbers for variables,
    and mapping constants module $p$. This isea is exploited in specific
    algorithms in computer algebra systems, e.g. algorithms for polynomial
    greatest common divisors. It is also used as a heuristic to speed up
    other calculations. But none exploit it in a systematic manner. The
    goal of this work was twofold. First, to design an AXIOM like system
    in which these signature functions can be constructed automatically,
    hence better exploited, and secondly, to exploit them in new ways. In
    this paper we report on the design of such a system, Gauss.",
  paper = "Mona93.pdf",
  keywords = "axiomref",
```

```
  beebe = "Monagan:1993:GPD"
}
```

———-

— **Monagan:1993:GPD** —

```
@InProceedings{Monagan:1993:GPD,
  author =       "M. B. Monagan",
  title =        {{Gauss: a parameterized domain of computation system
                 with support for signature functions}},
  crossref =     "Miola:1993:DIS",
  pages =        "81--94",
  month =        "",
  year =         "1993",
  bibdate =      "Fri Dec 29 12:46:02 MST 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The fastest known algorithms in classical algebra make
                 use of signature functions. That is, reducing
                 computation with formulae to computing with the
                 integers modulo p, by substituting random numbers for
                 variables, and mapping constants modulo p. This idea is
                 exploited in specific algorithms in computer algebra
                 systems, e.g. algorithms for polynomial greatest common
                 divisors. It is also used as a heuristic to speed up
                 other calculations. But none exploit it in a systematic
                 manner. The author designs an AXIOM like system in
                 which these signature functions can be constructed
                 automatically, hence better exploited. He exploits them
                 in new ways. He reports on the design of such a system,
                 Gauss.",
  acknowledgement = ack-nhfb,
  affiliation =  "Inst. fur Wissenschaftliches Rechnen, ETH, Zurich,
                 Switzerland",
  classification = "C4130 (Interpolation and function approximation);
                 C7310 (Mathematics)",
  keywords =     "AXIOM like system; Classical algebra; Computation
                 system; Computer algebra systems; Gauss; Heuristic;
                 Integers modulo; Mapping constants modulo;
                 Parameterized domain; Polynomial greatest common
                 divisors; Random numbers; Signature functions",
  language =     "English",
  thesaurus =    "Polynomials; Symbol manipulation",
}
```

———-

— **axiom.bib** —

```
@misc{Mona94,
  author = "Monagan, Michael B. and Gonnet, Gaston H.",
  title = {{Signature Functions for Algebraic Numbers}},
  link = "\url{http://lib.org/by/\_djvu\_Papers/Computer\_algebra/Algebraic\%20numbers}",
```

```
  abstract =
    "In 1980 Schwartz gave a fast {\sl probabilistic} method which tests
    if a matrix of polynomials of $\mathbb{Z}$ is singular or not. The
    method is based on the idea of {\sl signature functions} which are
    mappings of mathematical expressions into finite rings. In Schwartz's
    paper, they were polynomials over $\mathbb{Z}$ into GF($p$). Because
    computation in GF($p$) is very fast compared with computing with
    polynomials, Schwartz's method yields an enormous speedup both in
    theory and in practice. Therefore it is desirable to extend the class
    of expressions for which we can find effective signature functions. In
    the mid 80's Gonnet extended the class of expressions, for which
    signature functions can be found, to include a restricted class of
    elementary functions and integer roots. In this paper we present and
    compare methods for constructing signature functions for expressions
    containing {\sl algebraic numbers}. Some experimental results are
    given.",
  paper = "Mona94.djvu",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@InProceedings{Mona07,
  author = "Monagan, Michael and Pearce, Roman",
  title = {{Polynomial division using dynamic arrays, heaps, and packed
           exponent vectors}},
  booktitle = "Computer algebra in scientific computing",
  series = "CASC 2007",
  year = "2007",
  isbn = "978-3-540-75186-1",
  location = "Bonn",
  pages = "295-315",
  link = "\url{http://www.cecm.sfu.ca/~rpearcea/sdmp/sdmp\_paper.pdf}",
  abstract =
    "A common way of implementing multivariate polynomial multiplication
    and division is to represent polynomials as linked lists of terms
    sorted in a term ordering and to use repeated merging. This results in
    poor performance on large sparse polynomials.

    In this paper we use an auxiliary heap of pointers to reduce the
    number of monomial comparisons in the worst case while keeping the
    overall storage linear. We give two variations. In the first, the size
    of the heap is bounded by the number of terms in the quotient(s). In
    the second, which is new, the size is bounded by the number of terms
    in the divisor(s).

    We use dynamic arrays of terms rather than linked lists to reduce
    storage allocations and indirect memory references. We pack monomials
    in the array to reduce storage and to speed up monomial
    comparisons. We give a new packing for the graded reverse
    lexicographical ordering.
```

```
    We have implemented the heap algorithms in C with an interface to
    Maple. For comparison we have also implemented Yans ``geobuckets'' data
    structure. Our timings demonstrate that heaps of pointers are
    comparable in speed with geobuckets but use significantly less
    storage.",
  paper = "Mona07.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Mona00,
  author = "Monagan, Michael and Wittkopf, Allan D.",
  title = {{On the Design and Implementation of Brown's Algorithm over the
           Integers and Number Fields}},
  booktitle = "ISSAC 2000",
  pages = "225-233",
  year = "2000",
  isbn = "1-58113-218-2",
  abstract =
    "We study the design and implementation of the dense modular GCD
    algorithm of Brown applied to bivariate polynomial GCDs over the
    integers and number fields. We present an improved design of Brown's
    algorithm and compare it asymptotically with Brown's original
    algorithm, with GCDHEU, the heuristic GCD algorithm, and with the
    EEZGCD algorithm. We also make an empirical comparison based on Maple
    implementations of the algorithms. Our findings show that a careful
    implementation of our improved version of Brown's algorithm is much
    better than the other algorithms in theory and in practice.",
  paper = "Mona00.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Mont07,
  author = "Montes, Antonio",
  title = {{On the canonical discussion of polynomial systems with
            parameters}},
  year = "2007",
  link = "\url{http://arxiv.org/pdf/math/0601674.pdf}",
  abstract =
    "Given a parametric polynomial ideal $I$, the algorithm DISPGB,
    introduced by the author in 2002, builds up a binary tree describing a
    dichotomic discussion of the different reduced Groebner bases
    depending on the values of the parameters, whose set of terminal
    vertices form a Comprehensive Groebner System (CGS). It is relevant
    to obtain CGSs having further properties in order to make them more
```

```
    useful for the applications. In this paper the interest is focused on
    obtaining a canonical CGS. We define the objective, show the
    difficulties and formulate a natural conjecture. If the conjecture is
    true then such a canonical CGS will exist and can be computed. We also
    give an algorithm to transform our original CGS in this direction and
    show its utility in applications.",
  paper = "Mont07.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Mora 89]{Mor89}  Mora, T. (ed)
Applied Algebra, Algebraic Algorithms and Error-Correcting
Codes, 6th International Conference, AAECC-6, Rome, Italy, July 4-8, 1998,
Proceedings, volume 357 of Lecture Notes in Computer Science
Springer-Verlag, Berlin, Germany / Heildelberg, Germany / London, UK / etc.,
1989 ISBN 3-540-51083-4, LCCN QA268.A35 1988 Conference held jointly with
ISSAC '88
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Moses 71]{Mos71} Moses, Joel
  title = {{Algebraic Simplification: A Guide for the Perplexed}},
CACM August 1971 Vol 14 No. 8 pp527-537
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Moses 08]{Mos08} Moses, Joel
  title = {{Macsyma: A Personal History}},
Invited Presentation in Milestones in Computer Algebra, May 2008, Tobago
  link = "\url{http://esd.mit.edu/Faculty_Pages/moses/Macsyma.pdf}",
  abstract = "
    The Macsyma system arose out of research on mathematical software in
    the AI group at MIT in the 1960's. Algorithm development in symbolic
    integration and simplification arose out of the interest of people,
    such as the author, who were also mathematics students. The later
    development of algorithms for the GCD of sparse polynomials, for
    example, arose out of the needs of our user community. During various
    times in the 1970's the computer on which Macsyma ran was one of the
    most popular notes on the ARPANET. We discuss the attempts in the late
    70's and the 80's to develop Macsyma systems that ran on popular
    computer architectures.  Finally, we discuss the impact of the
```

```
    fundamental ideas in Macsyma on current research on large scale
    engineering systems.",
  paper = "Mos08.pdf",
  keywords = "axiomref"
```

———-

”

— **axiom.bib** —

```
@misc{Muld95,
  author = "Mulders, Thom",
  title = {{Primitives: Orepoly and Lodo}},
  year = "1995",
  link = "\url{ftp://ftp.inf.ethz.ch/org/cathode/workshops/jan95/abstracts/mulders.ps}",
  algebra =
  "\newline\refto{category OREPCAT UnivariateSkewPolynomialCategory}
   \newline\refto{category LODOCAT LinearOrdinaryDifferentialOperatorCategory}
   \newline\refto{domain AUTOMOR Automorphism}
   \newline\refto{domain ORESUP SparseUnivariateSkewPolynomial}
   \newline\refto{domain OREUP UnivariateSkewPolynomial}
   \newline\refto{domain LODO LinearOrdinaryDifferentialOperator}
   \newline\refto{domain LODO1 LinearOrdinaryDifferentialOperator1}
   \newline\refto{domain LODO2 LinearOrdinaryDifferentialOperator2}
   \newline\refto{package APPLYORE ApplyUnivariateSkewPolynomial}
   \newline\refto{package OREPCTO UnivariateSkewPolynomialCategoryOps}
   \newline\refto{package LODOF LinearOrdinaryDifferentialOperatorFactorizer}
   \newline\refto{package LODOOPS LinearOrdinaryDifferentialOperatorsOps}",
  paper = "Muld95.pdf",
  keywords = "axiomref"

}
```

———-

— **axiom.bib** —

```
@article{Muss82,
  author = "Musser, David R. and Kapur, Deepak",
  title = {{Rewrite Rule Theory and Abstract Data Type Analysis}},
  journal = "Lecture Notes in Computer Science",
  volume = "144",
  pages = "77-90",
  year = "1982",
  paper = "Muss82.pdf",
  keywords = "axiomref"
}
```

———-

1.40.14 N

— **axiom.bib** —

```
@InProceedings{Nayl06,
  author = "Naylor, William and Padget, Julian",
  title = {{From Untyped to Polymorphically Typed Objects in Mathematical
           Web Services}},
  booktitle = "Lecture Notes in Computer Science",
  volume = "4108",
  pages = "222-236",
  year = "2006",
  abstract =
    "OpenMath is a widely recognized approach to the semantic markup of
    mathematics that is often used for communication between OpenMath
    compliant systems. The Aldor language has a sophisticated
    category-based type system that was specifically developed for the
    purpose of modelling mathematical structures, while the system itself
    supports the creation of small-footprint applications suitable for
    deployment as web services. In this paper we present our first results
    of how one may perform translations from generic OpenMath objects into
    values in specific Aldor domains, describing how the Aldor interfae
    domain ExpresstionTree is used to achieve this. We outline our Aldor
    implementation of an OpenMath translator, and describe an efficient
    extention of this to the Parser category. In addition, the Aldor
    service creation and invocation mechanism are explained. Thus we are
    in a position to develop and deploy mathematical web services whose
    descriptions may be directly derived from Aldor's rich type language.",
  paper = "NPxx.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Naylor 95]{N95} Naylor, Bill
  title = {{Symbolic Interface for an advanced hyperbolic PDE solver}},
  link = "\url{http://www.sci.csd.uwo.ca/~bill/Papers/symbInterface2.ps}",
  abstract = "
    An Axiom front end is described, which is used to generate
    mathematical objects needed by one of the latest NAG routines, to be
    included in the Mark 17 version of the NAG Numerical library. This
    routine uses powerful techniques to find the solution to Hyperbolic
    Partial Differential Equations in conservation form and in one spatial
    dimension. These mathematical objects are non-trivial, requiring much
    mathematical knowledge on the part of the user, which is otherwise
    irrelvant to the physical problem which is to be solved. We discuss
    the individual mathematical objects, considering the mathematical
    theory which is relevant, and some of the problems which have been
    encountered and solved during the FORTRAN generation necessary to
    realise the object.  Finally we display some of our results.",
  paper = "N95.pdf",
```

```
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Naylor 00b]{ND00} Naylor, W.A.; Davenport, J.H.
  title = {{A Monte-Carlo Extension to a Category-Based Type System}},
  link = "\url{http://www.sci.csd.uwo.ca/~bill/Papers/monteCarCat3.ps}",
  abstract = "
    The normal claim for mathematics is that all calculations are 100\%
    accurate and therefore one calculation can rely completely on the
    results of sub-calculations, hoever there exist {\sl Monte-Carlo}
    algorithms which are often much faster than the equivalent
    deterministic ones where the results will have a prescribed
    probability (presumably small) of being incorrect. However there has
    been little discussion of how such algorithms can be used as building
    blocks in Computer Algebra.  In this paper we describe how the
    computational category theory which is the basis of the type structure
    used in the Axiom computer algebra system may be extended to cover
    probabilistic algorithms, which use Monte-Carlo techniques. We follow
    this with a specific example which uses Straight Line Program
    representation.",
  paper = "ND00.pdf",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Nehr85,
  author = "Nehrkorn, Klaus",
  title = {{Symbolic Integration of Exponential Polynomials}},
  booktitle = "European COnference on Computer Algebra",
  publisher = "Springer",
  pages = "599-600",
  year = "1985",
  comment = "LNCS 204",
  paper = "Nehr85.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Ngxx80,
  author = "Ng, Edward W.",
  title = {{Symbolic-Numeric Interface: A Review}},
  type = "technical report",
  year = "1980",
  number = "NASA-CR-162690 HC A02/MF A01",
```

```
  institution = "NASA Jet Propulsion Lab",
  link = "\url{http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19800008508.pdf}",
  abstract =
    "This is a survey of recent activities that either used or encouraged the
    potential use of a combination of symbolic and numerical calculations.
    Symbolic calculations here primarily refer to the computer processing of
    procedures from classical algebra, analysis and calculus. Numerical
    calculations refer to both numerical mathematics research and scientific
    computation. This survey is inteded to point out a large number of problem
    areas where a co-operation of symbolic and numeric methods is likely to
    bear many fruits. These areas include such classical operations as
    differentiation and integration, such diverse activities as function
    approximations andqualitative analysis, and such contemporary topics as
    finite element calculations and computational complexity. It is contended
    that other less obvious topics such as the fast Fourier transform, linear
    algebra, nonlinear analysis and error analysis would also benefti from a
    synergistic approach advocated here.",
  paper = "Ngxx80.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Norm75,
  author = "Norman, Arthur C.",
  title = {{Computing with Formal Power Series}},
  journal = "ACM Transactions on Mathematical Software",
  volume = "1",
  number = "4",
  pages = "346-356",
  year = "1975",
  doi = "10.1145/355656.355660",
  keywords = "axiomref",
  beebe = "Norman:1975:CFP"
}
```

———-

— Norman:1975:CFP —

```
@Article{Norman:1975:CFP,
  author =       "A. C. Norman",
  title =        {{Computing with Formal Power Series}},
  journal =      j-TOMS,
  volume =       "1",
  number =       "4",
  pages =        "346--356",
  month =        dec,
  year =         "1975",
  CODEN =        "ACMSCU",
  ISSN =         "0098-3500 (print), 1557-7295 (electronic)",
```

```
  ISSN-L =       "0098-3500",
  bibdate =      "Sat Aug 27 00:22:26 1994",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  fjournal =     "ACM Transactions on Mathematical Software",
  link =  "\url{http://portal.acm.org/toc.cfm?idx=J782}",
  keywords =     "Scratchpad",
}
```

———-

— **axiom.bib** —

```
@misc{Normxx,
  author = "Norman, Arthur C.",
  title = {{Notes 13: How to Compute a Groebner Basis}},
  link = "\url{http://people.math.umass.edu/~norman/462\_11/notes/m462notes13.pdf}",
  algebra =
  "\newline\refto{package AFALGGRO AffineAlgebraicSetComputeWithGroebnerBasis}
   \newline\refto{package GBEUCLID EuclideanGroebnerBasisPackage}
   \newline\refto{package GBF GroebnerFactorizationPackage}
   \newline\refto{package GBINTERN GroebnerInternalPackage}
   \newline\refto{package GB GroebnerPackage}
   \newline\refto{package GROEBSOL GroebnerSolve}
   \newline\refto{package INTERGB InterfaceGroebnerPackage}
   \newline\refto{package LGROBP LinGroebnerPackage}
   \newline\refto{package PGROEB PolyGroebner}",
  paper = "Normxx.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Norm75a,
  author = "Norman, Arthur C.",
  title = {{The SCRATCHPAD Power Series Package}},
  journal = "SIGSAM",
  volume = "9",
  number = "1",
  pages = "12-20",
  year = "1975",
  comment = "IBM T.J. Watson Research RC4998",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Norm94,
```

```
  author = "Norman, Arthur C.",
  title = {{Algebraic Manipulation}},
  paper = "Norm94.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Norm96,
  author = "Norman, Arthur C.",
  title = {{Memory tracing of algebraic calculations}},
  booktitle = "Proc. 1996 ISSAC",
  series = "ISSAC 1996",
  year = "1996",
  publisher = "ACM Press",
  location = "New York, NY",
  pages = "113-119",
  link = "\url{http://opus.bath.ac.uk/16452/1/NormanFitch96a.ps}",
  abstract =
    "We present a software tool which allows us to visualize details of the
    use of memory during the execution of an algebra system. We apply this
    to gain a better understanding of the behaviour of REDUCE, and hence
    to make proposals for ways in which the execution can be improved. The
    same tool will soon be used in the performance engineering of a
    version of axiom.",
  paper = "Norm96.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Nguy09,
  author = "Nguyen, Minh Van",
  title = {{Exploring Cryptography Using the Sage Computer Algebra System}},
  school = "Victoria University",
  year = "2009",
  abstract =
    "Cryptography has become indispensable in areas such as e-commerce,
    the legal safe-guarding of medical records, and secure electronic
    communication. Hence, it is incumbent upon software engineers to
    understand the concepts and techniques underlying the cryptosystems
    that they implement. An educator needs to consider which topics to
    cover in a course on cryptography as well as how to present the
    concepts and techniques to be covered in the course. This thesis
    contributes to the field of cryptography pedagogy by discussing and
    implementing small-scale cryptosystems whose encryption and
    decryption processes can be stepped through by hand. Our
    implementation has been accepted and integrated into the code base of
```

```
    the computer algebra system Sage. As Sage is free and open source,
    students and educators of cryptology need not worry about paying
    license fees in order to use Sage, but can instead concentrate on
    exploring cryptography using Sages built-in support for cryptography.",
  paper = "Nguy09.pdf",
  keywords = "axiomref"
}
```

———-

1.40.15 O

— **axiom.bib** —

```
@InProceedings{Oanc05,
  author = "Oancea, Cosmin E. and Watt, Stephen M.",
  title = {{Domains and expressions: an interface between two approaches to
           computer algebra}},
  booktitle = "Proc. 2005 ISSAC",
  series = "ISSAC 2005",
  year = "2005",
  isbn = "1-59593-095-7",
  location = "Beijing, China",
  pages = "261-268",
  link = "\url{http://www.csd.uwo.ca/~watt/pub/reprints/2005-issac-alma.pdf}",
  abstract =
    "This paper describes a method to use compiled, strongly typed Aldor
    domains in the interpreted, expression-oriented Maple
    environment. This represents a non-traditional approach to structuring
    computer algebra software: using an efficient, compiled language,
    designed for writing large complex mathematical libraries, together
    with a top-level system based on user-interface priorities and ease of
    scripting.

    We examine what is required to use Aldor libraries to extend Maple in
    an effective and natural way. Since the computational models of Maple
    and Aldor differ significantly, new run-time code must implement a
    non-trivial semantic correspondence. Our solution allows Aldor
    functions to run tightly coupled to the Maple environment, able to
    directly and efficiently manipulate Maple data objects. We call the
    overall system Alma.",
  paper = "Oanc05.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{ORMS,
  author = "Unknown",
```

```
  title = {{Oberwolfach References on Mathematical Software}},
  link = "\url{http://orms.mfo.de/project?id=234}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{OKee96,
  author = "O'Keefe, Christine M. and Storme, Leo",
  title = {{Arcs in PG(n,q) fixed by {A\_5} and {A\_6}}},
  journal = "J. Geom.",
  volume = "55",
  number = "1-2",
  pages = "123-138",
  year = "1996",
  abstract =
    "A $k$-arc in a projective space is a set of points, no three of which
    are collinear. The author determines the $k$ arcs in $PG(n,q)$ which
    are fixed by primitive groups isomorphic to $A_5$ or $A_6$. The best
    known examples are $q+1$ arcs: in general these are normal rational
    curves and are conics in $PG(2,q)$ and twisted cubics in $PG(3,q)$.
    The other cases turn out to be 6-arcs or 10-arcs.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Olli89,
  author = "Ollivier, F.",
  title = {{Inversibility of rational mappings and structural
           identifiablility in automatics}},
  booktitle = "Proc. SIGSAM 1989",
  series = "SIGSAM '89",
  pages = "43-54",
  isbn = "0-89791-325-6",
  year = "1989",
  abstract =
    "We investigate different methods for testing whether a rational
    mapping $f$ from $k^n$ to $k^m$ admits a rational inverse, or whether
    a polynomial mapping admits a polynomial one. We give a new solution,
    which seems much more efficient in practice than previously known ones
    using ``tag'' variables and standard basis, and a majoration for the
    degree of the standard basis calculations which is valid for both
    methods in the case of a polynomial map which is birational. We
    further show that a better bound can be given for our method, under
    some assumptions on the form of $f$. Our method can also extend to
    check whether a given polynomial belong to the subfield generated by a
    finite set of fractions.
```

```
    We then illustrate our algorithm, with a application to structural
    identifiability. The implantation has been done in the IBM computer
    algebra system Scratchpad II.",
  paper = "Olli89.pdf",
  keywords = "axiomref",
  beebe = "Ollivier:1989:IRM"
}
```

———-

— Ollivier:1989:IRM —

```
@InProceedings{Ollivier:1989:IRM,
  author =       "F. Ollivier",
  title =        {{Inversibility of rational mappings and structural
                 identifiability in automatics}},
  crossref =     "ACM:1989:PAI",
  pages =        "43--54",
  month =        "",
  year =         "1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The author investigates different methods for testing
                 whether a rational mapping f from k/sup n/ to k/sup m/
                 admits a rational inverse, or whether a polynomial
                 mapping admits a polynomial one. He gives a new
                 solution, which seems much more efficient in practice
                 than previously known ones using 'tag' variables and
                 standard basis, and a majoration for the degree of the
                 standard basis calculations which is valid for both
                 methods in the case of a polynomial map which is
                 birational. He shows that a better bound can be given
                 for the method, under some assumption on the form of f.
                 The method can also extend to check whether a given
                 polynomial belongs to the subfield generated by a
                 finite set of fractions. The author illustrates the
                 algorithm with an application to structural
                 identifiability. The implementation has been done in
                 the IBM computer algebra system Scratchpad II.",
  acknowledgement = ack-nhfb,
  affiliation =  "Lab. d'Inf. de l'X, Ecole Polytech., Palaiseau,
                 France",
  classification = "C1110 (Algebra); C1120 (Analysis); C7310
                 (Mathematics)",
  keywords =     "Computer algebra system; Fractions; IBM;
                 Inversibility; Polynomial inverse; Polynomial mapping;
                 Rational inverse; Rational mappings; Scratchpad II;
                 Structural identifiability",
  language =     "English",
  thesaurus =    "Inverse problems; Mathematics computing; Polynomials;
                 Set theory; Symbol manipulation",
}
```

———-

— **ignore** —

```
\bibitem[Online 72]{Onl72}.
Online 72: conference proceedings ... international conference on online
interactive computing, Brunel University, Uxbridge, England, 4-7 September
1972 ISBN 0-903796-02-3 LCCN QA76.55.O54 1972 Two volumes.
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[OpenMath]{OpenMa}.
  title = {{OpenMath Technical Overview}},
  link = "\url{http://www.openmath.org/overview/technical.html}",
  keywords = "axiomref"
```

———-

1.40.16 P

— **axiom.bib** —

```
@InProceedings{Page07,
  author = "Page, William S.",
  title = {{Axiom - Open Source Computer Algebra System}},
  booktitle = "Poster ISSAC 2007 Proceedings",
  series = "ISSAC 2007",
  year = "2007",
  volume = "41",
  pages = "114",
  abstract =
    "Axiom has been in development since 1971. Originally called
    Scratchpad II, it was developed by IBM under the direction of Richard
    Jenks[1]. The project evolved over a period of 20 years as a research
    platform for developing new ideas in computational mathematics.
    ScratchPad also attracted the interest and contributions of a large
    number of mathematicians and computer scientists outside of IBM. In
    the 1990s, the Scratchpad project was renamed to Axiom, and sold to
    the Numerical Algorithms Group (NAG) in England who marketed it as a
    commercial system. NAG withdrew Axiom from the market in October 2001
    and agreed to release Axiom as free software, under an open source
    license.

    Tim Daly (a former ScratchPad developer at IBM) setup a pubic open
    source Axiom project[2] in October 2002 with a primary goal to improve
    the documentation of Axiom through the extensive use of literate
```

```
    programming[3].  The first free open source version of Axiom was
    released in 2003. Since that time the project has attracted a small
    but very active group of developers and a growing number of users.

    This exhibit includes a laptop computer running a recent version
    of Axiom, Internet access (if available) to the Axiom Wiki website[4],
    and CDs containing Axiom software for free distribution[5].",
  keywords = "axiomref, TPDref",
  beebe = "Page:2007:AOS"
}
```

———-

— Page:2007:AOS —

```
@Article{Page:2007:AOS,
  author =       "William S. Page",
  title =        {{Axiom: open source computer algebra system}},
  journal =      j-ACM-COMM-COMP-ALGEBRA,
  volume =       "41",
  number =       "3",
  pages =        "114--114",
  month =        sep,
  year =         "2007",
  CODEN =        "????",
  DOI =          "http://doi.acm.org/10.1145/1358190.1358206",
  ISSN =         "1932-2232 (print), 1932-2240 (electronic)",
  ISSN-L =       "1932-2232",
  bibdate =      "Wed Jun 18 09:23:01 MDT 2008",
  bibsource =    "http://portal.acm.org/;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Axiom has been in development since 1971. Originally
                 called Scratchpad II, it was developed by IBM under the
                 direction of Richard Jenks[1]. The project evolved over
                 a period of 20 years as a research platform for
                 developing new ideas in computational mathematics.
                 ScratchPad also attracted the interest and
                 contributions of a large number of mathematicians and
                 computer scientists outside of IBM. In the 1990s, the
                 Scratchpad project was renamed to Axiom, and sold to
                 the Numerical Algorithms Group (NAG) in England who
                 marketed it as a commercial system. NAG withdrew Axiom
                 from the market in October 2001 and agreed to release
                 Axiom as free software, under an open source
                 license.\par

                 Tim Daly (a former ScratchPad developer at IBM) setup a
                 pubic open source Axiom project[2] in October 2002 with
                 a primary goal to improve the documentation of Axiom
                 through the extensive use of literate programming[3].
                 The first free open source version of Axiom was
                 released in 2003. Since that time the project has
                 attracted a small but very active group of developers
```

```
                and a growing number of users.\par

                This exhibit includes a laptop computer running a
                recent version of Axiom, Internet access (if available)
                to the Axiom Wiki website[4], and CDs containing Axiom
                software for free distribution[5].",
  acknowledgement = ack-nhfb,
  fjournal =     "ACM Communications in Computer Algebra",
  issue =        "161",
}
```

———-

— **axiom.bib** —

```
@misc{Page08,
  author = "Page, Bill",
  title = {{Algebraist Network}},
  link = "\url{http://lambda-the-ultimate.org/node/2737}",
  year = "2008",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Paulxx,
  author = "Paule, Peter and Kartashova, Lena and Kauers, Manuel and
            Schneider, Carsten and Winkler, Franz",
  title = {{Hot Topics in Symbolic Computation}},
  publisher = "Springer",
  link = "\url{http://www.risc.jku.at/publications/download/risc_3845/chapter1.pdf}",
  paper = "Paulxx.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Peti99,
  author = "Petitjean, S.",
  title = {{Algebraic Geometry and Computer Vision: Polynomial Systems, Real
           and Complex Roots}},
  journal = "J. of Mathematical Imaging and Vision",
  volume = "10",
  number = "1",
  year = "1999",
  link = "\url{http://www.loria.fr/~petitjea/papers/jmiv99.pdf}",
  abstract =
    "We review the different techniques known for doing exact computations
```

```
    on polynomial systems. Some are based on the use of Groebner bases and
    linear algebra, others on the more classical resultants and its modern
    counterparts. Many theoretical examples of the use of these techniques
    are given. Furthermore, a full set of examples of applications in the
    domain of artificial vision, where many constraints boil down to
    polynomial systems, are presented. Emphasis is also put on very recent
    methods for determining the number of (isolated) real and complex
    roots of such systems.",
  paper = "Peti99.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Petitot 90]{Pet90} Petitot, Michel
  title = {{Types r\'ecursifs en scratchpad, application aux polyn\^omes
            non commutatifs}},
LIFL, 1990
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Peti93,
  author = "Petitot, Michel",
  title = {{Experience with Axiom}},
  booktitle = "Proc. Int. IMACS Symposium on Symbolic Computation",
  pages = "240",
  year = "1993",
  keywords = "axiomref",
  beebe = "Petitot:1993:EA"
}
```

———-

— **Petitot:1993:EA** —

```
@InProceedings{Petitot:1993:EA,
  author =       "M. Petitot",
  title =        {{Experience with Axiom}}",
  crossref =     "Jacob:1993:PSI",
  pages =        "240",
  month =        "",
  year =         "1993",
  bibdate =      "Tue Sep 17 06:35:39 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The computer algebra system Axiom (formerly
                 Scratchpad) allows a strict typing of the manipulated
                 data. Using examples from noncommutative algebra
```

```
                (polynomials in noncommutative variables, Lie
                polynomials, Poincar{\'e}--Birkoff--Witt basis) the
                authors show the interest and the limits of some
                essential notions in Axiom: the genericity, the
                inheritance, the distinction domain/category and the
                type inference.",
  acknowledgement = ack-nhfb,
  affiliation =  "LIFL, Lille I Univ., Villeneuve d'Ascq, France",
  classification = "C1230 (Artificial intelligence); C4130
                (Interpolation and function approximation); C7310
                (Mathematics)",
  keywords =     "Category; Computer algebra system Axiom; Distinction
                domain; Genericity; Inheritance; Lie polynomials;
                Manipulated data; Noncommutative algebra;
                Poincar{\'e}--Birkoff--Witt basis; Polynomials;
                Scratchpad; Type inference",
  language =     "English",
  thesaurus =    "Inference mechanisms; Lie algebras; Mathematics
                computing; Polynomials",
}
```

———-

— **ignore** —

```
\bibitem[Petric 71]{Pet71} Petric, S. R. (ed)
Proceedings of the second symposium on Symbolic and
Algebraic Manipulation, March 23-25, 1971, Los Angeles, California, ACM Press,
New York, NY 10036, USA, 1971. LCCN QA76.5.S94 1971
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Pinch 93]{Pin93} Pinch, R.G.E.
  title = {{Some Primality Testing Algorithms}},
Devlin, Keith (ed.)
Computers and Mathematics November 1993, Vol 40, Number 9 pp1203-1210
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Pollxx,
  author = "Poll, Erik",
  title = {{The type system of Axiom}},
  link = "\url{https://www.cs.ru.nl/E.Poll/talks/axiom.pdf}",
  paper = "Polxx.pdf",
  keywords = "axiomref"
```

```
}
```

———-

— **axiom.bib** —

```
@InProceedings{Prit06,
  author = "Pritchard, F. Leon and Sit, William Y.",
  title = {{On Initial Value Problems for Ordinary Differential-Algebraic
           Equations}},
  booktitle = "Radon Series on Computational and Applied Mathematics",
  year = "2006",
  pages = "283-340",
  isbn = "978-3-11-019323-7",
  abstract =
    "This paper addresses polynomial implicit ODEs in an autonomous
    context. These ODEs are defined by a system of the form
    \[f_i(z_1,\cdots,z_n,\dot{z_1},\cdots,\dot{z_n})=0,\quad i=1,\cdots,m\]
    where $f_i$ is (for $i=1,\ldots,m$) a polynomial in $2n$ variables
    $(X,P)=(X_1,\ldots,X_n,P_1,\ldots,P_n)$. This covers in particular
    quasilinear systems, often encountered in applications and defined
    by polynomials $f_i$ in which the total degree in the variables $P$
    is at most one. The approach is close to the geometrical framework
    of Rabier and Rheinboldt, profitting from the polynomial form of
    the system.

    The authors develop an algorithm for the symbolic computation of
    the set of consistent initial values via ideal-theoretic results;
    this is based on a stationary algebraic process of ‘‘prolongation’’,
    together with the notions of the completion of a given ideal and the
    algebraic index of the system, defined as the number of steps taken
    by the process to stabilize. Over- and under-determined systems are
    also accommodated in their framework.",
  keywords = "axiomref, TPDref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Purt86,
  author = "Purtilo, J.",
  title = {{Applications of a software interconnection system in
           mathematical problem solving environments}},
  booktitle = "Proc.1986 Symposium on Symbolic and Algebraic Computation",
  series = "SYMSAC '86",
  pages = "16-23",
  year = "1986",
  publisher = "ACM Press",
  isbn = "0-89791-199-7",
  doi = "http://dx.doi.org/10.1145/32439.32443",
  keywords = "axiomref",
```

```
  beebe = "Purtilo:1986:ASI"
}
```

———-

— **Purtilo:1986:ASI** —

```
@InProceedings{Purtilo:1986:ASI,
  author =       "J. Purtilo",
  editor =       "Bruce W. Char",
  booktitle =    "Proceedings of the 1986 Symposium on Symbolic and
                 Algebraic Computation: Symsac '86, July 21--23, 1986,
                 Waterloo, Ontario",
  title =        {{Applications of a software interconnection system in
                 mathematical problem solving environments}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "16--23",
  year =         "1986",
  DOI =          "http://dx.doi.org/10.1145.32443",
  ISBN =         "0-89791-199-7",
  ISBN-13 =      "978-0-89791-199-3",
  LCCN =         "QA155.7.E4 A281 1986",
  bibdate =      "Thu Jul 26 09:26:18 2001",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "ACM order number 505860.",
  acknowledgement = ack-nhfb,
  keywords =     "Scratchpad",
}
```

———-

1.40.17 Q

1.40.18 R

— **axiom.bib** —

```
@book{Rals78,
  author = "Ralston, Anthony and Rabinowitz, Philip",
  title = {{A First Course in Numerical Analysis}},
  year = "1978",
  publisher = "McGraw-Hill",
  isbn = "0-07-051158-6",
}
```

———-

— **axiom.bib** —

```
@article{Riga99,
  author = "Rigal, Alain",
  title = {{High-order compact schemes: Application to bidimensional unsteady
           diffusion-convection problems. I}},
  journal = "C. R. Acad. Sci.",
  volume = "328",
  number = "6",
  pages = "535-538",
  year = "1999",
  abstract =
    "For unsteady 2D diffusion-convection problems, we present two classes
    of compact difference schemes of order 2 in time and 4 in space. These
    finite difference schemes are essentially derived from 1D schemes,
    extensively analyzed in our previous paper [J. Comput. Phys. 114,
    No. 1, 59-76 (1994; Zbl 0807.65056)]. We propose two approaches:
    construction of 2D schemes as product of 1D schemes and global
    formulation of 2D schemes. Part II by M. Fourni [C. R. Acad. Sci.,
    Paris, Sr. I, Math. 328, No. 6, 539-542 (1999; reviewed below)]
    focuses on the development and analysis of global schemes with the
    assistance of symbolic computation software (AXIOM).",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Riob09,
  author = "Rioboo, Renaud",
  title = {{Invariants for the FoCaL language}},
  journal = "Ann. Math. Artif. Intell.",
  volume = "56",
  number = "3-4",
  pages = "273-296",
  year = "2009",
  abstract =
    "We present a FoCaL formalization for quotient structures which are
    common in mathematics. We first present a framework for stating
    invariant properties of the data manipulated by running programs.  A
    notion of equivalence relation is then encoded for the FoCaL library.
    It is implemented through projections functions, this enables us to
    provide canonical representations which are commonly used in Computer
    Algebra but seldom formally described. We further provide a FoCaL
    formalization for the code used inside the library for modular
    arithmetic through the certification of quotient groups and quotient
    rings which are involved in the model. We finally instantiate our
    framework to provide a trusted replacement of the existing FoCaL
    library.",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Rioboo 03a]{Riob03a} Rioboo, Renaud
  title = {{Quelques aspects du calcul exact avec des nombres r\'eels}},
Ph.D. Thesis, Laboratoire d'Informatique Th\'eorique et Programmationg
  paper = "Riob03a.ps",
  paper = "Riob03a.pdf",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Rioboo 03]{Riob03} Rioboo, Renaud
  title = {{Towards Faster Real Algebraic Numbers}},
J. of Symbolic Computation 36 pp 513-533 (2003)
  abstract = "
    This paper presents a new encoding scheme for real algebraic number
    manipulations which enhances current Axiom's real closure. Algebraic
    manipulations are performed using different instantiations of
    sub-resultant-like algorithms instead of Euclidean-like algorithms.
    We use these algorithms to compute polynomial gcds and Bezout
    relations, to compute the roots and the signs of algebraic
    numbers. This allows us to work in the ring of real algebraic integers
    instead of the field of read algebraic numbers avoiding many
    denominators.",
  paper = "Riob03.pdf",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Roan03,
  author = "Roanes-Lozano, Eugenio and Roanes-Macias, Eugenio and
            Villar-Mena, M.",
  title = {{A bridge between dynamic geometry and computer algebra}},
  journal = "Math. Comput. Modelling",
  volume = "37",
  number = "9-10",
  pages = "1005-1028",
  year = "2003",
  link = "\url{http://ac.els-cdn.com/S0895717703001158/1-s2.0-S0895717703001158-main.pdf}",
  abstract =
    "Both Computer Algebra Systems (CASs) and dynamic geometry systems
    (DGSs) have reached a high level of development. Some CASs (like Maple
    or Derive) include specific and powerful packages devoted to Euclidean
    geometry, but CASs have incorporated neither mouse drawing
    capabilities nor dynamic capabilities. Meanwhile, the well-known DGSs
    do not provide algebraic facilities.

    Maples and Derives paramGeo packages and the DGS-CAS translator (all
```

```
    freely available from the authors) make it possible to draw a
    geometric configuration with the mouse (using The Geometers Sketchpad
    3 or 4) and to obtain the coordinates, equations, etc., of the drawn
    configuration in Maples or Derives syntax. To obtain complicated
    formulae, coordinates of points or equations of loci, to perform
    automatic theorem proving and to perform automatic discovery directly
    from sketches are examples of straightforward applications. Moreover,
    this strategy could be adapted to other CASs and DGSs.

    This work clearly has a didactic application in geometric problems
    exploration. Nevertheless, its main interest is to provide a
    convenient time-saving way to introduce data when dealing with rule
    and compass geometry, which has a wider scope than only educational
    purposes.",
  paper = "Roan03.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Roan10,
  author = "Roanes-Lozano, Eugenio and val Labeke, Nicolas and
            Roanes-Macias, Eugenio",
  title = {{Connecting the 3D DGS Calques3D with the CAS Maple}},
  journal = "Math. Comput. Simul.",
  volume = "80",
  number = "6",
  pages = "1153-1176",
  year = "2010",
  link = "\url{http://nvl.calques3d.org/publications/2010.MatCom.Connecting.pdf}",
  abstract =
    "Many (2D) Dynamic Geometry Systems (DGSs) are able to export numeric
    coordinates and equations with numeric coefficients to Computer
    Algebra Systems (CASs). Moreoever, different approaches and systems
    that linke (2D) DGSs with CASs, so that symbolic coordinates and
    equations with symbolic coefficients can be exported from the DGS to
    the CAS, already exist. Although the 2D DGS Calques3D can export
    numeric coordinates and equations with numeric coefficients to Maple
    and Mathematica, it cannot export symbolic coordinates and equations
    with symbolic coefficients. A connetion between the 3D DGS Calques3D
    and the CAS Maple, that can handle symbolic coordinates and equations
    with symbolic coefficients, is presented here. Its main interest is to
    provide a convenient time-saving way to explore problems and directly
    obtain both algebraic and numeric data when dealing with a 3D
    extension of ``ruler and compass geometry''. This link has not only
    educational purposes but mathematical ones, like mechanical theorem
    proving in geometry, geometry discovery (hypothesis completion),
    geometric loci finding -- As far as we know, there is no comparable
    ``symbolic'' link in the 3D case, except the prototype 3D-LD
    (restricted to determining algebraic surfaces as geometric loci).",
  paper = "Roan10.pdf",
```

```
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Roes99,
  author = "Roesner, K. G.",
  title = {{Supersonic flow around accelerated and decelerated bodies,
           analysed by analytical methods}},
  journal = "Z. Angew. Math. Mech.",
  volume = "79",
  number = "3",
  pages = "815-816",
  year = "1999",
  abstract =
    "By an extensive use of the computer algebra system AXIOM, a power
    series expansion with respect to the radial variable $r$ is used to
    describe the accelerated or decelerated supersonic flow field around
    the tip of slender conical bodies. The set of coupled nonlinear
    differential equations for the coefficient functions, depending on
    $\theta$ and $t$, is derived in closed form, and the first and second
    approximation of the coefficient functions are determined numerically.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
\article{Roja13,
  author = "Rojas-Bruna, Carlos",
  title = {{Trace forms and ideals on commutative algebras satisfying an
           identity of degree four}},
  journal = "Rocky Mt. J. Math.",
  year = "2013",
  volume = "43",
  number = "4",
  pages = "1325-1336",
  abstract =
    "This paper deals with the variety of commutative algebras satsifying
    the identity
    \[((xy)z)t - ((xy)t)z + ((yt)x)z - ((yt)z)x + ((yz)t)x - ((yz)x)t = 0\]
    These algebras appeared in the classification of the degree four
    identities in Carini et al. We prove the existence of a trace
    form. Moreover, if we assume the existence of degenerate trace form,
    the $A$ satisfies the identity $((yx)x)x=y((xx)x)$, a generalization
    of right-alternativity.  Finally we prove that $Ass[A]$ and $N(A)$ are
    ideals in these algebras.",
  paper = "Roja13.pdf",
  keywords = "axiomref"
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Robi93,
  author = "Robidoux, Nicolas",
  title = {{Does Axiom Solve Systems of O.D.E's Like Mathematica?}},
  year = "1993",
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/Robidoux.pdf}",
  abstract = "
    If I were demonstrating Axiom and were asked this question, my reply
    would be ``No, but I am not sure that this is a bad thing''. And I
    would illustrate this with the following example.

    Consider the following system of O.D.E.'s
    \[
    \begin{array}{rcl}
    \frac{dx_1}{dt} & = & \left(1+\frac{cos t}{2+sin t}\right)x_1\\
    \frac{dx_2}{dt} & = & x_1 - x_2
    \end{array}
    \]
    This is a very simple system: $x_1$ is actually uncoupled from $x_2$",
  paper = "Robi93.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Roes95,
  author = "Roesner, K. G.",
  title = {{Verified solutions for parameters of an exact solution for
           non-Newtonian liquids using computer algebra}},
  journal = "Zeitschrift fur Angewandte Mathematik und Physik",
  volume = "75",
  number = "suppl. 2",
  pages = "S435-S438",
  year = "1995",
  abstract =
    "An exact solution of the time independent velocity field for the
    Taylor--Couette flow of a polymer solution is derived solving the
    resulting first order ordinary differential equation of fifth degree
    analytically. Intensive use is made of computer algebra systems AXIOM
    and MACSYMA to find the exact solution.  The coaxial cylinders in the
    Taylor--Couette flow problem are assumed to rotate at different
    angular velocities. The geometrical and kinematic parameters can be
    chosen arbitrarily. The model equation for the material law of the
    viscoelastic liquid is based on the thermodynamic model for dilute
    solutions due to Lhuillier and Ouibrahim (1980) which is an analogy to
```

```
    the earlier paper of Frankel and Acrivos (1970). In the present
    investigation the influence of the parameters of the viscoelastic
    model on the velocity profile in the cylindrical gap is studied and
    the range of validity of the constitutive equation is investigated.",
  keywords = "axiomref",
  beebe = "Roesner:1995:VSP"
}
```

———-

— Roesner:1995:VSP —

```
@Article{Roesner:1995:VSP,
  author =       "K. G. Roesner",
  title =        {{Verified solutions for parameters of an exact solution
                 for non-Newtonian liquids using computer algebra}},
  journal =      j-ZEIT-ANGE-MATH-PHYS,
  volume =       "75",
  number =       "suppl. 2",
  pages =        "S435--S438",
  month =        "",
  year =         "1995",
  ISSN =         "0044-2267",
  bibdate =      "Fri Dec 29 12:46:02 MST 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "An exact solution of the time independent velocity
                 field for the Taylor--Couette flow of a polymer
                 solution is derived solving the resulting first order
                 ordinary differential equation of fifth degree
                 analytically. Intensive use is made of computer algebra
                 systems AXIOM and MACSYMA to find the exact solution.
                 The coaxial cylinders in the Taylor--Couette flow
                 problem are assumed to rotate at different angular
                 velocities. The geometrical and kinematic parameters
                 can be chosen arbitrarily. The model equation for the
                 material law of the viscoelastic liquid is based on the
                 thermodynamic model for dilute solutions due to
                 Lhuillier and Ouibrahim (1980) which is an analogy to
                 the earlier paper of Frankel and Acrivos (1970). In the
                 present investigation the influence of the parameters
                 of the viscoelastic model on the velocity profile in
                 the cylindrical gap is studied and the range of
                 validity of the constitutive equation is
                 investigated.",
  acknowledgement = ack-nhfb,
  affiliation =  "Inst. fur Mech., Tech. Hochschule Darmstadt, Germany",
  classification = "A0210 (Algebra, set theory, and graph theory); A0230
                 (Function theory, analysis); A0270 (Computational
                 techniques); A4710 (General fluid dynamics theory,
                 simulation and other computational methods); A4715
                 (Laminar flows); A4730 (Rotational flow, vortices,
                 buoyancy and other flows involving body forces); A4750
                 (Non-Newtonian dynamics)",
```

```
  keywords =     "AXIOM; Coaxial cylinders; Computer algebra;
                 Constitutive equation; Cylindrical gap; Dilute
                 solutions; Exact solution; First order ordinary
                 differential equation; Geometrical parameters;
                 Kinematic parameters; MACSYMA; Material law;
                 NonNewtonian liquids; Polymer solution; Taylor--Couette
                 flow; Thermodynamic model; Time independent velocity
                 field; Velocity profile; Viscoelastic liquid;
                 Viscoelastic model",
  language =     "English",
  thesaurus =    "Algebra; Couette flow; Differential equations; Flow
                 simulation; Non-Newtonian fluids; Physics computing;
                 Polymer solutions; Rotational flow; Thermodynamics",
}
```

———-

1.40.19 S

— axiom.bib —

```
@article{Safo00,
  author = "Safouhi, Hassan",
  title = {{The $HD$ and $H\overline{D}$ methods for accelerating the
          convergence of three-center nuclear attraction and four-center
          two-electron Coulomb integrals over $B$ functions and their
          convergence properties.}},
  journal = "J. Comput. Phys.",
  volume = "165",
  number = "2",
  pages = "473-495",
  year = "2000",
  abstract =
    "Three-center nuclear attraction and four-center two-electron
    Coulomb integrals over Slater-type orbitals are required for $ab$
    initio and density functional theory molecular structure
    calculations. They occur in many millions of terms, even for small
    molecules and require rapid and accurate evaluation. The $B$
    functions are used as a basis set of atomic orbitals. These
    functions are well adapted to the Fourier transform method that
    allowed analytical expressions for the integrals of interest to be
    developed. Rapid and accurate evaluation of these analytical
    expressions is now made possible by applying the $HD$ and
    $H\overline{D}$ methods for accelerating the convergence of the
    semi-infinite oscillatory integrals. The convergence properties of
    the new methods are analyzed. The numerical results section shows
    the high predetermined accuracy and the substantial gain in the
    calculation times obtained using the new methods.",
  paper = "Safo00.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Safo01,
  author = "Safouhi, Hassan",
  title = {{Numerical evaluation of three-center two-electron Coulomb and
           hybrid integrals over $B$ functions using the $HD$ and
           $H\overline{D}$ methods and convergence properties}},
  journal = "J. Math. Chem.",
  volume = "29",
  number = "3",
  pages = "213-232",
  year = "2001",
  abstract =
    "The $B$ function is a product of a spherical $K$ Bessel function and
    a spherical harmonic. The integrals to be evaluated contain
    complicated expressions of finite hypergeometric functions and
    spherical $J$ Bessel functions. Sequence transformation techniques are
    used for the numerical evaluation of the integrals. Numerical examples
    illustrate the accuracy and the efficiency of the method.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{SALSA,
  author = "SALSA",
  title = {{Solvers for Algebraic Systems and Applications}},
  link = "\url{http://www.ens-lyon.fr/LIP/Arenaire/SYMB/teams/salsa/proposal-salsa.pdf}",
  algebra =
    "\newline\refto{category TSETCAT TriangularSetCategory}
     \newline\refto{category RSETCAT RegularTriangularSetCategory}
     \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
     \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
     \newline\refto{package RSDCMPK RegularSetDecompositionPackage}",
  paper = "SALSA.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Salv89,
  author = "Salvy, Bruno",
  title = {{Examples of automatic asymptotic expansions}},
  institution = "Inst. Nat. Recherche Inf. Autom.",
  type = "technical report",
  number = "114",
  year = "1989",
```

```
  comment = "SIGSAM Bulletin Vol 25 No 2 1991 pp4-17",
  abstract =
    "We describe the current state of a Maple library, gdev, designed to
    perform asymptotic expansions for a large class of expressions. Many
    examples are provided, along with a short sketch of the underlying
    principles. At the time when this report is written, a striking
    feature of these examples is that none of them can be computed
    directly with any of today's most widespread symbolic computation
    systems (Macsyma, Mathematica, Maple or Scratchpad II).",
  paper = "Salv89.pdf",
  keywords = "axiomref",
  beebe = "Salvy:1989:EAA"
}
```

———-

— Salvy:1989:EAA —

```
@TechReport{Salvy:1989:EAA,
  author =       "B. Salvy",
  title =        {{Examples of automatic asymptotic expansions}},
  number =       "114",
  institution =  "Inst. Nat. Recherche Inf. Autom.",
  address =      "Le Chesnay, France",
  pages =        "18",
  month =        dec,
  year =         "1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Describes the current state of a Maple library, gdev,
                 designed to perform asymptotic expansions for a large
                 class of expressions. Many examples are provided, along
                 with a short sketch of the underlying principles. A
                 striking feature of these examples is that none of them
                 can be computed directly with any of the most
                 widespread symbolic computation systems (Macsyma,
                 Mathematica, Maple or Scratchpad II).",
  acknowledgement = ack-nhfb,
  classification = "C1120 (Analysis); C6130 (Data handling techniques);
                 C7310 (Mathematics)",
  keywords =     "Asymptotic expansions; Gdev; Maple library; Symbolic
                 computation systems",
  language =     "English",
  pubcountry =   "France",
  thesaurus =    "Mathematical analysis; Mathematics computing;
                 Subroutines; Symbol manipulation",
}
```

———-

— axiom.bib —

```
@article{Salv91,
```

```
  author = "Salvy, Bruno",
  title = {{Examples of automatic asymptotic expansions}},
  journal = "SIGSAM Bulletin",
  volume = "25",
  number = "2",
  pages = "4-17",
  year = "1991",
  abstract =
    "We describe the current state of a Maple library, gdev, designed to
    perform asymptotic expansions for a large class of expressions. Many
    examples are provided, along with a short sketch of the underlying
    principles. At the time when this report is written, a striking
    feature of these examples is that none of them can be computed
    directly with any of today's most widespread symbolic computation
    systems (Macsyma, Mathematica, Maple or Scratchpad II).",
  keywords = "axiomref",
  beebe = "Salvy:1991:EAA"
}
```

———-

— Salvy:1991:EAA —

```
@Article{Salvy:1991:EAA,
  author =       "B. Salvy",
  title =        {{Examples of automatic asymptotic expansions}},
  journal =      j-SIGSAM,
  volume =       "25",
  number =       "2",
  pages =        "4--17",
  month =        apr,
  year =         "1991",
  CODEN =        "SIGSBZ",
  ISSN =         "0163-5824 (print), 1557-9492 (electronic)",
  ISSN-L =       "0163-5824",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Describes the current state of a Maple library, gdev,
                 designed to perform asymptotic expansions for a large
                 class of expressions. Many examples are provided, along
                 with a short sketch of the underlying principles. At
                 the time when this report is written, a striking
                 feature of these examples is that none of them can be
                 computed directly with any of today's most widespread
                 symbolic computation systems (Macsyma, Mathematica,
                 Maple or Scratchpad II).",
  acknowledgement = ack-nhfb,
  affiliation =  "LIX, Ecole Polytech., Palaiseau, France",
  classification = "C6130 (Data handling techniques); C7310
                 (Mathematics)",
  fjournal =     "SIGSAM Bulletin",
  keywords =     "Automatic asymptotic expansions; Expressions; Gdev;
                 Maple library; Symbolic computation systems",
```

```
  language =     "English",
  pubcountry =   "USA",
  thesaurus =    "Symbol manipulation",
}
```

———-

— **axiom.bib** —

```
@article{Sant93,
  author = "Santas, Philip S.",
  title = {{A Type System for Computer Algebra}},
  booktitle = "DISCO 1993",
  year = "1993",
  pages = "177-191",
  publisher = "Springer",
  abstract =
    "We examine type systems for support of subtypes and categories in
    computer algebra systems. By modeling representation of instances
    in terms of existential types instead of recursive types, we
    obtain not only a simplified model, but we build a basis for
    defining subtyping among algebraic domains. The introduction of
    metaclasses, facilitates the task, by allowing the inference of
    type classes. By means of type classes and existential types we
    construct subtype relations without involving coercions.",
  paper = "Sant93.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Sant95,
  author = "Santas, Philip S.",
  title = {{A Type System for Computer Algebra}},
  journal = "J. Symbolic Computation",
  volume = "19",
  number = "1-3",
  pages = "79-109",
  year = "1995",
  abstract =
    "This paper presents a type system for support of subtypes,
    parameterized types with sharing and categories in a computer algebra
    environment. By modeling representation of instances in terms of
    existential types, we obtain a simplified model, and build a basis for
    defining subtyping among algebraic domains. The inheritance at
    category level has been formalized; this allows the automatic
    inference of type classes. By means of type classes and existential
    types we construct subtype relations without involving coercions. A
    type sharing mechanism works in parallel and allows the consistent
    extension and combination of domains. The expressiveness of the system
```

```
    is further increased by viewing domain types as special case of
    package types, forming weak and strong sums respectively. The
    introduced system, although awkward at first sight, is simpler than
    other proposed systems for computer algebra without including some of
    their problems. The system can be further extended in other to support
    more constructs and increase its flexibility.",
  paper = "Sant95.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Sant96,
  author = "Santas, Philip S.",
  title = {{Conditional Categories and Domains}},
  booktitle = "Proc. DISCO 1996",
  year = "1996",
  pages = "112-125",
  isbn = "3-540-61697-7",
  abstract =
    "We extend the Type System defined in [San95] with Axiom-like
    Conditional Categories with the additional property of Static Typing
    and Checking.  Categories and Domains may contain conditionals in
    their bodies, which are elaborated by our compiler by techniques used
    in standard typing.  We define an appropriate calozlus and discuss
    its properties.  Examples inspired by the Axiom library illustrate the
    power of our approach and its application in constructing algebraic
    concepts.  The full calculus has been implemented and tested with our
    LA compiler which generates executable files.",
  paper = "Sant96.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@book{Sant05,
  author = "Santas, Philip S.",
  title = {{Conditional Categories and Domains}},
  booktitle = "Design and Implementation of Symbolic Computation Systems",
  year = "2005",
  series = "Lecture Notes in Computer Science",
  volume = "1128",
  publisher = "Springer",
  abstract =
    "We extend the Type system defined in [Sant95] with Axiom-like
    Conditional Categories with the additional property of Static Typing
    and Checking. Categories and Domains may contain conditionals in their
    bodies, which are elaborated by our compiler by techniques used in
```

```
    standard typing. We define an appropriate calculus and discuss its
    properties. Examples inspired by the Axiom library illustrate the
    power of our apprach and its application in constructing algebraic
    concepts. The full calculus has been implemented and tested with our
    LA compiler which generated executable files.",
  paper = "Sant05.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Saun80,
 author = "Saunders, B. David",
 title = {{A Survey of Available Systems}},
 journal = "SIGSAM Bull.",
 issue_date = "November 1980",
 volume = "14",
 number = "4",
 month = "November",
 year = "1980",
 issn = "0163-5824",
 pages = "12--28",
 numpages = "17",
 link = "\url{http://doi.acm.org/10.1145/1089235.1089237}",
 doi = "10.1145/1089235.1089237",
 acmid = "1089237",
 publisher = "ACM",
 address = "New York, NY, USA",
 paper = "Saun80.pdf",
 keywords = "axiomref,survey"
}
```

———-

— **ignore** —

```
\bibitem[Schu 92]{Sch92} Sch\"u, J.
  title = {{Implementing des Cartan-Kuranishi-Theorems in AXIOM}},
Master's diploma thesis (in german), Institut f\"ur Algorithmen und
Kognitive Systeme, Universit\"t Karlsruhe 1992
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@book{Schw95,
  author = "Schwardmann, Ulrich",
  title = {{Computer algebra systems}},
```

```
  comment = "Computeralgebra-Systeme, German",
  publisher = "Addison-Wesley",
  year = "1995",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Schw87,
  author = "Schwarz, Fritz",
  title = {{Programming with abstract data types: the symmetry package
          SPDE in Scratchpad'}},
  booktitle = "Trends in Computer Algebra",
  series = "Lecture Notes in Computer Science 296",
  year = "1987",
  pages = "167-176",
  isbn = "3-540-18928-9",
  abstract =
    "The main problem which occurs in developing Computer Algebra packages
    for special areas in mathematics is the complexity. The unique concept
    which is advocated to cope with that problem is the introduction of
    suitable abstract data types. The corresponding decomposition into
    modules makes it much easier to develop, maintain and change the
    program. After introducing the relevant concepts from software
    engineering they are elaborated by means of the symmetry analysis of
    differential equations and the Scratchpad package SPDE which
    abbreviates Symmetries of Partial Differential Equations.",
  paper = "Schw87.pdf",
  keywords = "axiomref, printed",
  beebe = "Schwarz:1988:PAD"
}
```

———-

— **Schwarz:1988:PAD** —

```
@InCollection{Schwarz:1988:PAD,
  author =       "F. Schwarz",
  title =        {{Programming with abstract data types: the symmetry
                 package SPDE in Scratchpad}},
  crossref =     "Janssen:1988:TCA",
  pages =        "167--176",
  year =         "1988",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/cathode.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— **axiom.bib** —

```
@inproceedings{Schw89,
  author = "Schwarz, Fritz",
  title = {{A factorization algorithm for linear ordinary
           differential equations}},
  booktitle = "Proc. SYMSAC 1989",
  series = "SYMSAC '89",
  isbn = "0-89791-325-6",
  year = "1989",
  pages = "17-25",
  abstract =
    "The reducibility and factorization of linear homogeneous differential
    equations are of great theoretical and practical importance in
    mathematics.  Although it has been known for a long time that
    factorization is in principle a decision procedure, its use in an
    automatic differential equation solver requires a more detailed
    analysis of the various steps involved. Especially important are
    certain auxiliary equations, the so-called associated equations. An
    upper bound for the degree of its coefficients is derived.  Another
    important ingredient is the computation of optimal estimates for the
    size of polynomial and rational solutions of certain differential
    equations with rotational coefficients. Applying these results, the
    design of the factorization algorithm LODEF and its implementation in
    the Scratchpad II Computer Algebra System is described.",
  paper = "Schw89.pdf",
  keywords = "axiomref",
  beebe = "Schwarz:1989:FAL"
}
```

———-

— **Schwarz:1989:FAL** —

```
@InProceedings{Schwarz:1989:FAL,
  author =       "F. Schwarz",
  title =        {{A factorization algorithm for linear ordinary
                 differential equations}},
  crossref =     "ACM:1989:PAI",
  pages =        "17--25",
  month =        "",
  year =         "1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The reducibility and factorization of linear
                 homogeneous differential equations are of great
                 theoretical and practical importance in mathematics.
                 Although it has been known for a long time that
                 factorization is in principle a decision procedure, its
                 use in an automatic differential equation solver
                 requires a more detailed analysis of the various steps
                 involved. Especially important are certain auxiliary
                 equations, the so-called associated equations. An upper
                 bound for the degree of its coefficients is derived.
```

```
                Another important ingredient is the computation of
                optimal estimates for the size of polynomial and
                rational solutions of certain differential equations
                with rotational coefficients. Applying these results,
                the design of the factorization algorithm LODEF and its
                implementation in the Scratchpad II Computer Algebra
                System is described.",
  acknowledgement = ack-nhfb,
  affiliation =  "GMD, Inst. F1, St. Augustin, West Germany",
  classification = "C1120 (Analysis); C4170 (Differential equations);
                C7310 (Mathematics)",
  keywords =     "Associated equations; Automatic differential equation
                solver; Factorization algorithm; Linear ordinary
                differential equations; LODEF; Optimal estimates;
                Polynomial solutions; Rational solutions; Rotational
                coefficients; Scratchpad II Computer Algebra System;
                Upper bound",
  language =     "English",
  thesaurus =    "Linear differential equations; Mathematics computing;
                Polynomials; Symbol manipulation",
}
```

———-

— axiom.bib —

```
@article{Schw91,
  author = "Schwarz, Fritz",
  title = {{Monomial orderings and Groebner bases}},
  journal = "SIGSAM Bulletin",
  volume = "25",
  number = "1",
  year = "1991",
  pages = "10-23",
  abstract =
    "Let there be given a set of monomials in n variables and some order
    relations between them. The following {\sl fundamental problem of
    monomial ordering} is considered. Is it possible to decide whether
    these ordering relations are consistent and if so to extend them to an
    {\sl admissible} ordering for all monomials? The answer is given in
    terms of the algorithm {\sl MACOT} which constructs a matrix of so
    called {\sl cotes} which establishes the desired ordering
    relations. The main area of application of this algorithm, i.e. the
    construction of Groebner bases for different orderings and of
    universal Groebner bases is treated in the last section.",
  keywords = "axiomref",
  beebe = "Schwarz:1991:MOG"
}
```

———-

— Schwarz:1991:MOG —

```
@Article{Schwarz:1991:MOG,
  author =       "F. Schwarz",
  title =        {{Monomial orderings and Gr\"obner bases}},
  journal =      j-SIGSAM,
  volume =       "25",
  number =       "1",
  pages =        "10--23",
  month =        jan,
  year =         "1991",
  CODEN =        "SIGSBZ",
  ISSN =         "0163-5824 (print), 1557-9492 (electronic)",
  ISSN-L =       "0163-5824",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Let there be given a set of monomials in n variables
                 and some order relations between them. The following
                 fundamental problem of monomial ordering is considered.
                 Is it possible to decide whether these ordering
                 relations are consistent and if so to extend them to an
                 admissible ordering for all monomials? The answer is
                 given in terms of the algorithm MACOT which constructs
                 a matrix of so-called cotes which establishes the
                 desired ordering relations. The main area of
                 application of this algorithm, i.e. the construction of
                 Gr{\"o}bner bases for different orderings and of
                 universal Gr{\"o}bner bases, is presented. An
                 implementation in Scratchpad is also briefly
                 described.",
  acknowledgement = ack-nhfb,
  affiliation =  "GMD Inst., St. Augustin, Germany",
  classification = "C1110 (Algebra); C4140 (Linear algebra); C7310
                 (Mathematics)",
  fjournal =     "SIGSAM Bulletin",
  keywords =     "Computer algebra; Thomas theorem; Multivariate
                 polynomial; Gr{\"o}bner bases; Monomial ordering;
                 Ordering relations; Admissible ordering; MACOT; Matrix;
                 Cotes; Scratchpad",
  language =     "English",
  pubcountry =   "USA",
  thesaurus =    "Algebra; Matrix algebra; Polynomials; Symbol
                 manipulation",
}
```

———-

— axiom.bib —

```
@InProceedings{Schw02,
  author = "Schwarz, Fritz",
  title = {{ALLTYPES: An algebraic language and type system}},
  booktitle = "1st Int. Congress on Mathematical Software",
  year = "2002",
  isbn = "981-238-048-5",
```

```
  location = "Beijing China",
  pages = "486-500",
  link = "\url{http://www.scai.fraunhofer.de/content/dam/scai/de/documents/Mitarbeiterinnen-und-Mitarbeiter/ICMS
  abstract =
    "The software system ALLTYPES provides an environment that is
    particularly designed for developing software in differential
    algebra. Its most important features may be described as follows: A
    set of about thirty parametrized algebraic types is defined. Data
    objects represented by these types may be manipulated by more than one
    hundred polymorphic functions. Reusability of code is achieved by
    genericity and multiple inheritance. The user may extend the system by
    defining new types and polymorphic functions. A language comprising
    seven basic language constructs is defined for implementing
    mathematical algorithms. The easy manipulation of types is
    particularly supported due to a special portion of the language
    dedicated to manipulating typed objects, i.e. for performing
    user-defined or automatic type coercions. Type inquiries are also
    included in the language.",
  paper = "Schw02.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Schw94,
  author = "Schwarz, Fritz",
  title = {{Computer algebra software for scientific applications}},
  booktitle = "Computerized symbolic manipulation in mechanics",
  year = "1994",
  publisher = "Springer-Verlag",
  pages = "67-117",
  series = "CISM Courses Lecture 343",
  abstract =
    "The central subject of this article are two basic questions: How to
    make the process of developing computer algebra software on a large
    scale ($10^4$ to $10^5$) lines of code or more) more efficient and
    how to improve the quality of the result. Taking procedures from well
    established engineering sciences as a guide, two fundamental
    principles turned out to be of overwhelming importance: Modularization
    and limitation of growth through reuse. Important means for achieving
    these goals turned out to be concept of an abstract data type and the
    principles of object-oriented design. It is advocated to install an
    additional abstraction level between the mathematics and the machine
    in order to render it possible to develop (computer algebra) system
    independent mathematical software. Basic constituents of this level
    are a type system and a high-level language.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Seil94,
  author = "Seiler, Werner Markus",
  title = {{Analysis and Application of the Formal Theory of
           Partial Differential Equations}},
  school = "Universitat Karlsruhe",
  year = "1994",
  link =
  "\url{http://www.mathematik.uni-kassel.de/~seiler/Papers/Diss/diss.ps.gz}",
  abstract = "
    An introduction to the formal theory of partial differential equations
    is given emphasizing the properties of involutive symbols and
    equations.  An algorithm to complete any differential equation to an
    involutive one is presented. For an involutive equation possible
    values for the number of arbitrary functions in its general solution
    are determined. The existence and uniqueness of solutions for analytic
    equations is proven.  Applications of these results include an
    analysis of symmetry and reduction methods and a study of gauge
    systems. It is show that the Dirac algorithm for systems with
    constraints is closely related to the completion of the equation of
    motion to an involutive equation. Specific examples treated comprise
    the Yang-Mills Equations, Einstein Equations, complete and Jacobian
    systems, and some special models in two and three dimensions. To
    facilitate the involved tedious computations an environment for
    geometric approaches to differential equations has been developed in
    the computer algebra system Axiom. The appendices contain among others
    brief introductions into Carten-K{\"a}hler Theory and Janet-Riquier
    Theory.",
  paper = "Seil94.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Seil94a,
  author = "Seiler, Werner Markus",
  title = {{Completion to involution in AXIOM}},
  booktitle = "Proc. Rhine Workshop on Computer Algebra",
  year = "1994",
  pages = "103-104",
  abstract =
    "We have implemented an algorithm to complete a given system of
    partial differential equations to an involutive one in the computer
    algebra system AXIOM. An earlier version of this program has been
    described in Schu, Seiler, and Calmet (1992). The new version is much
    more efficient due to better simplification and many new features. It
    also provides procedures for the analysis of the arbitrariness of the
    general solution.  The goal of the implementation was not to transform
    simply an algorithm into a program but to start with the construction
    of an environment for symbolic computations within the geometric
```

```
    theory of differential equations. The modular structure allows an easy
    extension e.g. by a package for the symmetry analysis.",
  keywords = "axiomref",
  beebe = "Seiler:1994:CIA"
}
```

———-

— Seiler:1994:CIA —

```
@InProceedings{Seiler:1994:CIA,
  author =       "W. M. Seiler",
  title =        {{Completion to involution in AXIOM}},
  crossref =     "Calmet:1994:RWC",
  pages =        "103--104",
  month =        "",
  year =         "1994",
  bibdate =      "Tue Sep 17 06:32:41 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "We have implemented an algorithm to complete a given
                 system of partial differential equations to an
                 involutive one in the computer algebra system AXIOM. An
                 earlier version of this program has been described in
                 Schu, Seiler, and Calmet (1992). The new version is
                 much more efficient due to better simplification and
                 many new features. It also provides procedures for the
                 analysis of the arbitrariness of the general solution.
                 The goal of the implementation was not to transform
                 simply an algorithm into a program but to start with
                 the construction of an environment for symbolic
                 computations within the geometric theory of
                 differential equations. The modular structure allows an
                 easy extension e.g. by a package for the symmetry
                 analysis.",
  acknowledgement = ack-nhfb,
  affiliation =  "Inst. fur Algorithmen und Kognitive Syst., Karlsruhe
                 Univ., Germany",
  classification = "C4170 (Differential equations); C7310 (Mathematics
                 computing)",
  keywords =     "AXIOM; Computer algebra system; Involution; Partial
                 differential equations; Symbolic computations; Symmetry
                 analysis",
  language =     "English",
  thesaurus =    "Mathematics computing; Partial differential equations;
                 Symbol manipulation",
}
```

———-

— axiom.bib —

```
@article{Seil94b,
  author = "Seiler, Werner Markus",
```

```
  title = {{Pseudo Differential Operators and Integrable Systems in AXIOM}},
  journal = "Computer Physics Communications",
  volume = "79",
  number = "2",
  pages = "329-340",
  year = "1994",
  abstract =
    "An implementation of the algebra of pseudo differential operators in
    the computer algebra system Axiom is described. In several exmaples
    the application of the package to typical computations in the theory
    of integrable systems is demonstrated.",
  paper = "Seil94b.pdf",
  keywords = "axiomref",
  beebe = "Seiler:1994:PDO"
}
```

———-

— Seiler:1994:PDO —

```
@Article{Seiler:1994:PDO,
  author =       "Werner M. Seiler",
  title =        {{Pseudo differential operators and integrable systems
                 in AXIOM}},
  journal =      j-COMP-PHYS-COMM,
  volume =       "79",
  number =       "2",
  pages =        "329--340",
  month =        apr,
  year =         "1994",
  CODEN =        "CPHCBZ",
  DOI =          "http://dx.doi.org/10.1016/0010-4655(94)90076-0",
  ISSN =         "0010-4655 (print), 1879-2944 (electronic)",
  ISSN-L =       "0010-4655",
  bibdate =      "Mon Feb 13 21:29:43 MST 2012",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib;
                 http://www.math.utah.edu/pub/tex/bib/compphyscomm1990.bib",
  link =          "\url{http://www.sciencedirect.com/science/article/pii/0010465594900760}",
  abstract =     "An implementation of the algebra of pseudo
                 differential operators in the computer algebra system
                 AXIOM is described. In several examples the application
                 of the package to typical computations in the theory of
                 integrable systems is demonstrated.",
  acknowledgement = ack-nhfb,
  affiliation =  "Inst. fur Algorithmen und Kognitive Syst., Karlsruhe
                 Univ., Germany",
  classification = "C4170 (Differential equations); C7310
                 (Mathematics)",
  fjournal =     "Computer Physics Communications",
  link =  "\url{http://www.sciencedirect.com/science/journal/00104655}",
  keywords =     "AXIOM; Computer algebra; PDO; Pseudo differential
                 operators",
  language =     "English",
```

```
  pubcountry =   "Netherlands",
  thesaurus =    "Mathematics computing; Partial differential equations;
                 Symbol manipulation",
}
```

———-

— **axiom.bib** —

```
@misc{Seil95,
  author = "Seiler, Werner Markus",
  title = {{Applying AXIOM to partial differential equations}},
  institution = {Universit\"at Karlsruhe, Fakult\"at f\"ur Informatik},
  year = "1995",
  type = "Internal Report",
  number = "95-17",
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/Axiom-pdf.pdf}",
  abstract =
    "We present an Axiom environment called JET for geometric computations
    with partial differential equations within the framework of the jet
    bundle formalism. This comprises expecially the completion of a given
    differential equation to an involutive one according to the
    Cartan-Kuranishi Theorem and the setting up of the determining system
    for the generators of classical and non-classical Lie
    symmetries. Details of the implementations are described and
    applications are given. An appendix contains tables of all exported
    functions.",
  paper = "Seil95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Seil95a,
  author = "Seiler, Werner Markus and Calmet, J.",
  title = {{JET -- An Axiom Environment for Geometric Computations with
           Differential Equations}},
  link = "\url{http://axiom-wiki.newsynthesis.org/public/refs/axiom-jet95.pdf}",
  abstract =
    "JET is an environment within the computer algebra system Axiom to
    perform such computations. The current implementation emphasises the
    two key concepts involution and symmetry. It provides some packages
    for the completion of a given system of differential equations to an
    equivalent involutive one based on the Cartan-Kuranishi theorem and
    for setting up the determining equations for classical and
    non-classical point symmetries.",
  paper = "Seil95a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Seil97,
  author = "Seiler, Werner M.",
  title = {{Computer Algebra and Differential Equations: An Overview}},
  journal = "mathPAD7",
  volume = "7",
  pages = "34-49",
  year = "1997",
  link = "\url{http://www.mathematik.uni-kassel.di/~seiler/Papers/Postscript/CADERep.ps.gz}",
  abstract =
    "We present an informal overview of a number of approaches to
    differential equations which are popular in computer algebra. This
    includes symmetry and completion theory, local analysis, differential
    ideal and Galois theory, dynamical systems and numerical analysis.  A
    large bibliography is provided.",
  paper = "Seil97.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Seil99,
  author = "Seiler, Werner Markus",
  title = {{DETools: A Library for Differential Equations}},
  year = "1999",
  abstract =
    "This article tries to give at least a brief introduction. The MuPAD
    library is extended on two levels. The first one consists of a new
    library detools containing a number of routines for treating
    differential equations. This includes support for the graphical
    presentation of the output of the numerical routines in MuPAD, some
    methods for analysing or generating differential equations and also
    routines for solving some classes of partial differential
    equations. The use of this new library will be described in this
    article. The second level is somewhat more advanced and requires a
    certain familiarity with the object-oriented domains.",
  paper = "Seil99.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Seil01,
  author = "Seiler, Werner Markus",
  title = {{Involution - The Formal Theory of Differential Equations and
```

```
          its Applications in Computer Algebra and Numerical Analysis}},
  year = "2001",
  link =
  "\url{http://www.mathematik.uni-kassel.de/~seiler/Papers/Habil/habil.ps.gz}",
  paper = "Seil01.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Sene87,
  author = "Senechaud, P. and Siebert, F. and Villard, Gilles",
  title = {{Scratchpad II: Pr{\'e}sentation d'un nouveau langage de
          calcul formel}},
  type = "Technical Report",
  number = "640-M",
  institution = "TIM 3 (IMAG)",
  address = "Grenoble, France",
  year = "1987",
  keywords = "axiomref",
  beebe = "Senechaud:1987:SIP"
}
```

———-

— **Senechaud:1987:SIP** —

```
@TechReport{Senechaud:1987:SIP,
  author =       "P. Senechaud and F. Siebert and G. Villard",
  title =        {{Scratchpad II: Pr\'esentation d'un nouveau langage
                 de calcul formel}},
  number =       "640-M",
  institution =  "TIM 3 (IMAG)",
  address =      "Grenoble, France",
  pages =        "??",
  month =        feb,
  year =         "1987",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.1.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— **axiom.bib** —

```
@techreport{Sene87a,
  author = "Senechaud, Pascale and Siebert, F.",
  title = {{Etude dl l'algorithme de Kovacic et son implantation sur
          Scratchpad II}},
  type = "Technical Report",
```

```
  number = "639",
  institution = "Institut IMAG, Informatique et Mathematiques Appliquees
                 de Grenoble",
  address = "Grenoble, France",
  year = "1987",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Shan88,
  author = "Shannon, D. and Sweedler, M.",
  title = {{Using Gr\"obner bases to determine algebra membership,
           split surjective algebra homomorphisms determine birational
           equivalence}},
  journal = "Journal of Symbolic Computation",
  volume = "6",
  number = "2-3",
  pages = "267-273",
  year = "1988",
  abstract =
    "This paper presents a simple algorithm, based on Groebner bases, to
    test if a given polynomial $g$ of $k(X_1,\ldots,X_n)$ lies in
    $k(f_1,\ldots,f_m)$ where $k$ is a field, $X_1,\ldots,X_n$ are
    indeterminates over $k$ and $f_1,\ldots,f_m$ in $k(X_1,\ldots,X_n)$.
    If so, the algorithm produces a polynomial $P$ of $m$ variables
    where $g=P(f_1,\ldots,f_m)$. Say $\Omega(B)$ to $k(X_1,\ldots,X_n)$ is a
    homomorphism where $\Omega(b_i)=f_i$, for algebra generators ($b_i$)
    contained in / implied by $B$. If $\Omega$ is onto, the algorithm
    gives a homomorphism $\lambda k(X_1,\ldots,X_n)$ to $B$, where the
    composite $\Omega \lambda$ is the identity map. In particular, the
    algorithm computes the inverse of algebra automorphisms of the
    polynomial ring. A variation of the test if
    $k(f_1,\ldots,f_m)=k(X_1,\ldots,X_n)$ tells if
    $k(f_1,\ldots,f_m)=k(X_1,\ldots,X_n)$. Existing computer algebra
    systems, such as IBM's SCRATCHPAD II, have Groebner basis packages
    which allow the user to specify a term ordering sufficient to carry
    out the algorithm.",
  keywords = "axiomref",
  beebe = "Shannon:1988:UGB"
}
```

———-

— Shannon:1988:UGB —

```
@Article{Shannon:1988:UGB,
  author =       "D. Shannon and M. Sweedler",
  title =        {{Using Gr\"obner bases to determine algebra
                 membership, split surjective algebra homomorphisms
                 determine birational equivalence}},
```

```
  journal =      j-J-SYMBOLIC-COMP,
  volume =       "6",
  number =       "2-3",
  pages =        "267--273",
  month =        oct # "-" # dec,
  year =         "1988",
  CODEN =        "JSYCEH",
  ISSN =         "0747-7171 (print), 1095-855X (electronic)",
  ISSN-L =       "0747-7171",
  bibdate =      "Tue Sep 17 06:48:10 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "This paper presents a simple algorithm, based on
                 Gr{\"o}bner bases, to test if a given polynomial g of
                 k(X/sub 1/,\ldots{}, X/sub n/) lies in k(f/sub
                 1/,\ldots{}, f/sub m/) where k is a field, X/sub
                 i/,\ldots{}, X/sub n/ are indeterminates over k and
                 f/sub 1/,\ldots{}, f/sub m/ in k(X/sub 1/,\ldots{},
                 X/sub n/). If so, the algorithm produces a polynomial P
                 of m variables where g=P(f/sub 1/,\ldots{}, f/sub m/).
                 Say omega:B to k(X/sub 1/,\ldots{}, X/sub n/) is a
                 homomorphism where omega (b/sub i/)=f/sub i/, for
                 algebra generators (b/sub i/) contained in/implied by
                 B. If omega is onto, the algorithm gives a homomorphism
                 lambda:k(X/sub 1/,\ldots{}, X/sub n/) to B, where the
                 composite omega lambda is the identity map. In
                 particular, the algorithm computes the inverse of
                 algebra automorphisms of the polynomial ring. A
                 variation of the test if k(f/sub 1/,\ldots{}, f/sub
                 m/)=k(X/sub 1/,\ldots{}, X/sub n/), tells if k(f/sub
                 1/,\ldots{}, f/sub m/)=k(X/sub 1/,\ldots{}, X/sub n/).
                 Existing computer algebra systems, such as IBM'S
                 SCRATCHPAD II, have Gr{\"o}bner basis packages which
                 allow the user to specify a term ordering sufficient to
                 carry out the algorithm.",
  acknowledgement = ack-nhfb,
  affiliation =  "Dept. of Math., Transylvania Univ., Lexington, KY,
                 USA",
  classification = "C4130 (Interpolation and function approximation);
                 C6130 (Data handling techniques); C7310 (Mathematics)",
  fjournal =     "Journal of Symbolic Computation",
  link =  "\url{http://www.sciencedirect.com/science/journal/07477171}",
  keywords =     "IBM; Gr{\"o}bner bases; Algebra membership; Split
                 surjective algebra homomorphisms; Birational
                 equivalence; Polynomial; Homomorphism; Algebra
                 generators; Identity map; Algebra automorphisms;
                 Computer algebra systems; SCRATCHPAD II",
  language =     "English",
  pubcountry =   "UK",
  thesaurus =    "Polynomials; Symbol manipulation",
}
```

———-

— **axiom.bib** —

```
@misc{SIGS16,
  author = "SIGSAM, ACM",
  title = {{Axiom}},
  link = "\url{http://www.sigsam.org/software/axiom.html}",
  year = "2016",
  contact = "Infodir\_SIGSAM\@acm.org",
  abstract =
    "Axiom is a free, open source, general-purpose computer algebra
    system. It features a strongly typed language. The system has an
    interactive interpreter and a compiler. It includes over 1100
    supported categories, domains, and packages covering large areas of
    Mathematics."
}
```

———-

— **axiom.bib** —

```
@article{Sing93,
  author = "Singer, Michael F. and Ulmer, Felix",
  title = {{Galois groups of second and third order linear differential
           equations}},
  journal = "J. Symb. Comput.",
  volume = "16",
  number = "1",
  pages = "9-36",
  year = "1993",
  abstract =
    "The authors discuss the first problem of Galois theory of differential
    equations. Let $F$ be an ordinary (for simplicity) differential field
    and $L(y)=0$ be an ordinary linear differential equation over $F$. How
    can one calculate the Galois group of $L$ over $F$? The authors
    suppose a new approach to the problem. They reduce it to the problem
    of finding solutions of linear differential equations in $F$ and to
    the factorization problem of such equations over $F$. These allow them
    to give simple necessary and sufficient conditions for a second order
    linear differential equation to have Liouvillian solutions and for a
    third order linear differential equation to have Liouvillian solutions
    or to be solvable in terms of second order equations.",
  paper = "Sing93.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Sing93a,
  author = "Singer, Michael F. and Ulmer, Felix",
  title = {{Liouvillian and algebraic solutions of second and third order
           linear differential equations}},
```

```
  journal = "J. Symb. Comput.",
  volume = "16",
  number = "1",
  pages = "37-73",
  year = "1993",
  abstract =
    "Let $F$ be an ordinary differential field of characteristic 0 and
    $L \in F <y>$ be a linear homogeneous polynomial. How can one find the
    Liouvillian solutions of $L(y)=0$? In the paper this problem is
    reduced to the problems of (1) factorization and (2) finding $u$
    solutions such that $\frac{u^{\prime}}{y} \in F$ of $L$ and some
    polynomials associated with it (symmetric powers of $L$).

    Now there are the algorithms for the solution of the last problems for
    $F=\mathbb{Q}(x)$ [see D. Yu. Grigorev, J. Symb. Comput. 10, 7-37
    (1990; Zbl 0728.68067) and M. F. Singer, Am. J. Math. 103, 661-682
    (1981; Zbl 0477.12026)].

    For polynomials $L$ of the second and third order the authors provide
    full investigation of the most difficult case when the solution $u$ of
    $L(y)$ is algebraic. They show that one can compute the minimal
    polynomial $P(y) \in F[y]$ of $u$. We note that the authors
    essentially used the tools of representation theory, invariant theory
    and computer algebra.",
  paper = "Sing93a.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@inproceedings{Sitx89,
  author = "Sit, William Y.",
  title = {{On Goldman's algorithm for solving first-order multinomial
           autonomous systems}},
  booktitle = "Proc. Algebraic Algorithms and Error-Correcting Codes, AAECC-6",
  series = "Lecture Notes in Computer Science 357",
  location = "Rome, Italy",
  year = "1988",
  isbn = "3-540-51083-4",
  pages = "386-395",
  abstract =
    "In this article, a brief exposition of a method for finding first
    integrals for first order multinomial autonomous systems (FOMAS) of
    ordinary differential equations with constant coefficients will be
    given. The method is a simplified as well as a redesigned version
    based on a paper of Goldman (1987). We shall see how it can be applied
    to FOMAS with parametric coefficients. The algorithm is currently
    being implemented by the author, using the SCRATCHPAD II computer
    algebra language and system at the IBM T.J. Watson Research Center.

    FOMAS occur and are of interest in many disciplines and their first
```

```
integrals (or trajectories of motion) are generally difficult to
find. Examples of FOMAS are too numerous to list, some well-known ones
are the Riccati equation, the Lotka-Volterra equations for competing
populations, Selkov's model for chemical reactions, the Lorenz system
of the Rayleigh-Bernard problem, and Hamiltonian systems (where the
Hamiltonian is a sum of monomial terms with constant coefficients).

Let $Y=(y_1,\ldots,y_n)$ be $n$ functions depending on the variable
$\tau$. A monomial in $Y$ is a product of the form $y_1^{k_1}\cdots
y_n^{k_n}$, where $k_1,\ldots,k_n$ are constants. If
$K=(k_1,\ldots,k_n)$, we shall denote the monomial in $Y$ by $Y^K$,
and $K$ is called the exponent vector for the monomial. By convention,
exponent vectors are column vectors, but whenever convenient, we shall
write exponent vectors as row vectors. We say that $Y$ satisfies a
first-order multinomial autonomous system (FOMAS) if for each $i$, $1
\le i \le n$, $y_i$ satisfies a first order differential equation of
the form:
\[y_i^{\prime} = f_i(Y)\quad\quad\quad(1)\]
where $f_i$ is a linear combination of monomials in $Y$ with coefficients
which may be either constants or parametric constants. For example, the
Lotka-Volterra equations for three competing species considered by
Schwarz and Steeb (1984), form a FOMAS:
\[x_1^{\prime}=x_1(1+ax_2+bx_3)\]
\[x_2^{\prime}=x_2(1-ax_1+bx_3)\]
\[x_3^{\prime}=x_3(1-bx_1-cx_2)\]
When the exponent vectors occuring in $f_i$ are all non-negative integers,
as in the example above, a FOMAS reduces to a polynomial autonomous
system (FOPAS).

A computer program was developed by Schwarz (1986) to compute the
first integrals of FOPAS's which are themselves polynomials in
$y_1,\ldots,y_n$. Schwarz's algorithm literally takes a general
polynomial of a fixed degree $d$ in $n$ variables and substitutes it
into (1). This method does not work well on a FOMAS, because in a
FOMAS, the exponent vectors need not have integral components. Also,
it wll not find integrals with exponent vectors that involve
fractional or irrational numbers.

Goldman (1987) proved a theorem which gives necessary and sufficient
conditions for the existence of a multinomial first integral for
FOMAS. The proof also contained the outline of an algorithm for
finding such integrals. In Goldman's paper, he introduced the notion
of an integral array, which is a certain matrix satisfying some 10
conditions. He gave a few hints and several examples but did not
elaborate on how such an integral array can be found in general
(except in the case $q=2$). Assuming such an array is found, he can
compute the integral, in most cases, by solving systems of linear
equations, or at worse in certain cases, by solving a system of
algebraic equations. It was not clear when algebraic conditions are
necessary.

In this brief exposition, Goldman's method will be expanded to a
complete algorithm with a new simplified notation. The integral arrays
are replaced by addition schemes (which is equivalent to integral
```

```
    arrays with some conditions removed). The generation of addition
    schemes is a combinatorial problem unrelated, in a sense, to
    FOMAS. When the first integral is a polynomial, the additioin scheme
    is trivial to compute. We shall now begin by explaining some details
    of this theory.",
  keywords = "axiomref",
  beebe = "Sit:1989:GAS"
}
```

———-

— Sit:1989:GAS —

```
@InProceedings{Sit:1989:GAS,
  author =       "W. Y. Sit",
  title =        {{On Goldman's algorithm for solving first-order
                 multinomial autonomous systems}},
  crossref =     "Mora:1989:AAA",
  pages =        "386--395",
  month =        "",
  year =         "1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "A brief exposition of a method for finding first
                 integrals for first order multinomial autonomous
                 systems (FOMAS) of ordinary differential equations with
                 constant coefficients is given. The method is a
                 simplified as well as a redesigned version based on a
                 paper of Goldman (1987). The author shows how it can be
                 applied to FOMAS with parametric coefficients. The
                 algorithm is currently being implemented using the
                 SCRATCHPAD II computer algebra language and system at
                 the IBM TJ Watson Research Center.",
  acknowledgement = ack-nhfb,
  affiliation =  "Dept. of Math., City Coll. of New York, NY, USA",
  classification = "B0290P (Differential equations); B0290R (Integral
                 equations); C4170 (Differential equations); C4180
                 (Integral equations); C7310 (Mathematics)",
  keywords =     "Computer algebra language; Constant coefficients;
                 First integrals; First order multinomial autonomous
                 systems; FOMAS; Goldman algorithm; IBM; Ordinary
                 differential equations; SCRATCHPAD II",
  language =     "English",
  thesaurus =    "Differential equations; Integral equations;
                 Mathematics computing",
}
```

———-

— axiom.bib —

```
@article{Sitx92,
  author = "Sit, William Y.",
```

```
  title = {{An algorithm for solving parametric linear systems}},
  journal = "Journal of Symbolic Computations",
  volume = "13",
  number = "4",
  pages = "353-394",
  year = "1992",
  abstract =
    "We present a theoretical foundation for studying parametric systesm of
    linear equations and prove an efficient algorithm for identifying all
    parametric values (including degnerate cases) for which the system is
    consistent. The algorithm gives a small set of regimes where for each
    regime, the solutions of the specialized systems may be given
    uniformly. For homogeneous linear systems, or for systems were the
    right hand side is arbitrary, this small set is irredunant. We discuss
    in detail practical issues concerning implementations, with particular
    emphasis on simplification of results. Examples are given based on a
    close implementation of the algorithm in SCRATCHPAD II. We also give a
    complexity analysis of the Gaussian elimination method and compare
    that with our algorithm.",
  paper = "Sitx92.pdf",
  keywords = "axiomref",
  beebe = "Sit:1992:ASP"
}
```

———-

— Sit:1992:ASP —

```
@Article{Sit:1992:ASP,
  author =       "W. Y. Sit",
  title =        {{An algorithm for solving parametric linear systems}},
  journal =      j-J-SYMBOLIC-COMP,
  volume =       "13",
  number =       "4",
  pages =        "353--394",
  month =        apr,
  year =         "1992",
  CODEN =        "JSYCEH",
  ISSN =         "0747-7171 (print), 1095-855X (electronic)",
  ISSN-L =       "0747-7171",
  bibdate =      "Tue Sep 17 06:41:20 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The author presents a theoretical foundation for
                 studying parametric systems of linear equations and
                 proves an efficient algorithm for identifying all
                 parametric values (including degenerate cases) for
                 which the system is consistent. The algorithm gives a
                 small set of regimes where for each regime, the
                 solutions of the specialized systems may be given
                 uniformly. For homogeneous linear systems, or for
                 systems where the right hand side is arbitrary, this
                 small set is irredundant. He discusses in detail
                 practical issues concerning implementations, with
```

```
                particular emphasis on simplification of results.
                Examples are given based on a close implementation of
                the algorithm in SCRATCHPAD II. He also gives a
                complexity analysis of the Gaussian elimination method
                and compares that with the algorithm.",
  acknowledgement = ack-nhfb,
  affiliation =  "Dept. of Math., City Coll. of New York, NY, USA",
  classification = "B0290H (Linear algebra); C4140 (Linear algebra);
                C4240 (Programming and algorithm theory); C7310
                (Mathematics)",
  fjournal =     "Journal of Symbolic Computation",
  link =  "\url{http://www.sciencedirect.com/science/journal/07477171}",
  keywords =     "Complexity analysis; Efficient algorithm; Gaussian
                elimination method; Linear equations; Parametric linear
                systems; SCRATCHPAD II",
  language =     "English",
  pubcountry =   "UK",
  thesaurus =    "Computational complexity; Matrix algebra; Symbol
                manipulation",
}
```

———-

— **ignore** —

```
\bibitem[Sit 06]{Sit06} Sit, Emil
  title = {{Tools for Repeatable Research}},
  link = "\url{http://www.emilsit.net/blog/archives/tools-for-repeatable-research}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Smed92,
  author = "Smedley, Trevor J.",
  title = {{Using pictorial and object oriented programming for computer
           algebra}},
  booktitle = "Applied computing --- technological challenges of the
              1990's: proceedings of the 1992 ACM\slash SIGAPP
              Symposium on Applied Computing, Kansas City Convention
              Center, March 1--3, 1992",
  publisher = "ACM Press",
  pages = "1243-1247",
  year = "1992",
  isbn = "0-89791-502-X",
  keywords = "axiomref",
  beebe = "Smedley:1992:UPO"
}
```

———-

— **Smedley:1992:UPO** —

```
@InProceedings{Smedley:1992:UPO,
  author =       "Trevor J. Smedley",
  editor =       "Hal Berghel and others",
  booktitle =    "Applied computing --- technological challenges of the
                 1990's: proceedings of the 1992 ACM\slash SIGAPP
                 Symposium on Applied Computing, Kansas City Convention
                 Center, March 1--3, 1992",
  title =        {{Using pictorial and object oriented programming for
                 computer algebra}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "1243--1247",
  year =         "1992",
  DOI =          "http://dx.doi.org/10.1145.130154",
  ISBN =         "0-89791-502-X",
  ISBN-13 =      "978-0-89791-502-1",
  LCCN =         "QA76.76.A65 S95 1992",
  bibdate =      "Thu Jul 26 09:02:03 2001",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  bookpages =    "xiii + 1257 (2 volumes)",
  keywords =     "Scratchpad",
}
```

———-

— **axiom.bib** —

```
@article{Smit93,
  author = "Smith, Geoff C.",
  title = {{Group theory results with machine generated proofs}},
  journal = "An. Univ. Timis., Ser. Mat.-Inform.",
  volume = "31",
  number = "2",
  pages = "273-280",
  year = "1993",
  abstract =
    "There are a variety of theorems in group theory which admit of proofs
    by machine. This talk illustrates these techniques in action. Examples
    are given of this phenomenon, drawn from the theory of group
    presentations, and from the theory of $p$-groups. The systems involved
    include AXIOM, CAYLEY and QUOTPIC",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Smit07,
```

```
  author = "Smith, Jacob and Dos Reis, Gabriel and Jarvi, Jaakko",
  title = {{Algorithmic differentiation in Axiom}},
  booktitle = "ACM SIGSAM Proceedings",
  series = "ISSAC 2007",
  year = "2007",
  pages = "347-354",
  isbn = "978-1-59593-743-8",
  abstract = "
    This paper describes the design and implementation of an algorithmic
    differentiation framework in the Axiom computer algebra system. Our
    implementation works by transformations on Spad programs at the level
    of the typed abstract syntax tree -- Spad is the language for extending
    Axiom with libraries. The framework illustrates an algebraic theory
    of algorithmic differentiation, here only for Spad programs, but
    we suggest that the theory is general. In particular, if it is
    possible to define a compositional semantics for programs, we define
    the exact requirements for when a program can be algorithmically
    differentiated. This leads to a general algorithmic differentiation
    system, and is not confined to functions which compute with basic
    data types, such as floating point numbers.",
  paper = "Smit07.pdf",
  keywords = "axiomref, printed",
  beebe = "Smith:2007:ADA"
}
```

———-

— Smith:2007:ADA —

```
@InProceedings{Smith:2007:ADA,
  author =       "Jacob Smith and Gabriel {Dos Reis} and Jaakko
                 J{\"a}rvi",
  title =        {{Algorithmic differentiation in Axiom}},
  crossref =     "Brown:2007:PIS",
  pages =        "347--354",
  year =         "2007",
  DOI =          "http://doi.acm.org/10.1145/1277548.1277595",
  bibdate =      "Fri Jun 20 08:46:50 MDT 2008",
  bibsource =    "http://portal.acm.org/;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "This paper describes the design and implementation of
                 an algorithmic differentiation framework in the Axiom
                 computer algebra system. Our implementation works by
                 transformations on Spad programs at the level of the
                 typed abstract syntax tree -- Spad is the language for
                 extending Axiom with libraries. The framework
                 illustrates an algebraic theory of algorithmic
                 differentiation, here only for Spad programs, but we
                 suggest that the theory is general. In particular, if
                 it is possible to define a compositional semantics for
                 programs, we define the exact requirements for when a
                 program can be algorithmically differentiated. This
                 leads to a general algorithmic differentiation system,
```

```
                and is not confined to functions which compute with
                basic data types, such as floating point numbers.",
  acknowledgement = ack-nhfb,
  keywords =     "algorithmic differentiation; axiom; program
                transformation; symbolic-numeric computation",
}
```

———-

— **axiom.bib** —

```
@phdthesis{Smit10,
  author = "Smith, Jacob Nyffeler",
  title = {{Techniques in Active and Generic Software Libraries}},
  school = "Texas A and M University",
  year = "2010",
  link = "\url{oaktrust.library.tamu.edu/bitstream/handle/1969.1/ETD-TAMU-2010-05-7823/SMITH-DISSERTATION.pdf}",
  abstract =
    "Reusing code from software libraries can reduce the time and effort
    to construct software systems and also enable the development of
    larger systems. However, the benefits that come from the use of
    software libraries may not be realized due to limitations in the way
    that traditional software libraries are constructed. Libraries come
    equipped with application programming interfaces (API) that help
    enforce the correct use of the abstraction in those libraries. Writing
    new components and adapting existing ones to conform to library APIs
    may require substantial amounts of ‘‘glue’’ code that potentially
    affects software’s efficiency, robustness, and ease-of-maintenance.
    If, as a result, the idea of reusing functionality from a software
    library is rejected, no benefits of reuse will be realized.

    This dissertation explores and develops techniques that support the
    construction of software libraries with abstraction layers that do not
    impede efficiency. In many situations, glue code can be expected to
    have very low (or zero) performance overhead. In particular, we
    describe advances in the design and development of active libraries --
    software libraries that take an active role in the compilation of the
    user’s code. Common to the presented techniques is that they may
    ‘‘break’’ a library API (in a controlled manner) to adapt the
    functionality of the library for a particular use case.

    The concrete contributions of this dissertation are: a library API
    that supports iterator selection in the Standard Template Library,
    allowing generic algorithms to find the most suitable traversal
    through a container, allowing (in one case) a 30-fold improvement in
    performance; the development of techniques, idioms, and best practices
    for {\tt concepts} and {\tt concept\_maps} in C++, allowing the
    construction of algorithms for one domain entirely in terms of
    formalisms from a second domain; the construction of generic
    algorithms for algorithmic differentiation, implemented as an active
    library in Spad, language of the Open Axiom computer algebra system,
    allowing algorithmic differentiation to be applied to the appropriate
    mathematical object and not just concrete data-types; and the
```

```
    description of a static analysis framework to describe the generic
    programming notion of local specialization with Spad, allowing more
    sophisticated (value-based) control over algorithm selection and
    specialization in categories and domains.

    We will find that active libraries simultaneously increase the
    expressivity of the underlying language and the performance of
    software using those libraries",
  paper = "Smit10.pdf",
  keywords = "axiomref",
}
```

———-

— **axiom.bib** —

```
@misc{Smit12,
  author = "Smith, Robert",
  title = {{Excursions in Mathematics Using Lisp}},
  link = "\url{https://www.youtube.com/watch?v=nTI_d-jS6dI}",
  year = "2012",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Spit11,
  author = "Spitters, Bas and van der Weegen, Eelis",
  title = {{Type Classes for Mathematics in Type Theory}},
  journal = "Math. Struct. Comput. Sci.",
  volume = "21",
  number = "4",
  pages = "795-825",
  year = "2011",
  link = "\url{https://arxiv.org/pdf/1102.1323.pdf}",
  abstract =
    "The introduction of first-class type classes in the Coq system calls
    for a re-examination of the basic interfaces used for mathematical
    formalisation in type theory. We present a new set of type classes for
    mathematics and take full advantage of their unique features to make
    practical a particularly flexible approach that was formerly thought
    to be infeasible. Thus, we address traditional proof engineering
    challenges as well as new ones resulting from our ambition to build
    upon this development a library of constructive analysis in which any
    abstraction penalties inhibiting efficient computation are reduced to
    a minimum.

    The basis of our development consists of type classes representing a
    standard algebraic hierarchy, as well as portions of category theory
    and universal algebra. On this foundation, we build a set of
```

```
    mathematically sound abstract interfaces for different kinds of
    numbers, succinctly expressed using categorical language and universal
    algebra constructions.

    Strategic use of type classes lets us support these high-level
    theory-friendly definitions, while still enabling efficient
    implementations unhindered by gratuitous indirection, conversion or
    projection.

    Algebra thrives on the interplay between syntax and semantics. The
    Prolog-like abilities of type class instance resolution allow us to
    conveniently define a quote function, thus facilitating the use of
    reflective techniques.",
  paper = "Spit11.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Schu92,
  author = "Schu, J. and Seiler, Werner Markus and Calmet, Jacques",
  title = {{Algorithmic Methods For Lie Pseudogroups'}},
  booktitle = "Proc. Modern Group Analysis: Advanced Analytical and
               Computational Methods in Mathematical Physics",
  pages = "337-344",
  location =  "Acireale (Italy)",
  year = "1992",
  publisher = "Kluwer",
  link = "\url{http://www.iks.kti.edu/fileadmin/User/calmet/papers/Acireale-93.ps.gz}",
  abstract =
    "The authors recall the concepts of involutive non-linear systems of
    partial differential equations, the classical Cartan-Riquier-Janet
    criterion for involutiveness, and the Cartan-Kuranishi prolongation
    theorem. Then the algorithm of prolongation into involutive case is
    clearly outlined, the arising computational problems are discussed,
    experience with implementation in the computer algebra system AXIOM
    (1992) is described, and comparison with Maple and REDUCE algorithms
    is made.",
  paper = "Schu92.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Stau95,
  author = "Stauffer, Deitrich",
  title = {{Annual Review of Computational Physics I}},
  publisher = "unknown",
```

```
section = "5.2.2",
year = "1995",
abstract =
  "Based on our experiences with IRENA, we decided to use generic
  inter-process communication tools for the link to AXIOM. This has the
  added advantage that we can operate across a network. The main
  technique we use is the {\sl Remote Procedure Call} (RPC) [Sun
  Microsystems Inc., 1988] which allows us to interact with a server on
  another machine (or on the local machine). RPC takes care of
  differences in data representation (e.g. the byte-order of floating
  point numbers) on different architectures.

  AXIOM is a multi-process package. Normally when a user starts up the
  system they start up the various components which then interact via
  standard socket operations. If they are using the line, they start up
  a new process: the NAG Manager (NAGMAN for short). Additionally, there
  will be a NAG daemon (NAGD) running on any machine on which the user
  may wish to execute NAG routines (which could include the local
  host). NAGMAN commnicates with the running AXIOM system via a socket
  down which is transmitted the details of and data for the particular
  routine to be called. NAGMAN calls a NAGD on another machine via RPC
  and eventually returns the results to AXIOM.

  NAGD consists of the server program, and a set of stub codes designed
  to call individual NAG routines. It is, in effect, a remotely-callable
  version of the NAG library. There is no reason why AXIOM should be the
  only system to use it, and indeed there are plans to incorporate the
  ability to call NAGD into other systems.

  An ASP is treated just like any other piece of data by the AXIOM-NAG
  link. The source code is passed to NAGD and compiled. (There are
  various optimisations to prevent the same code being compiled multiple
  times, but the details needn't concern us here.) This compiled code is
  linked with the NAG Library to make the executable. Thus if a user
  calls the same NAG routine with different ASPs the routine will be
  relinked each time.

  It would be nice if this were not necessary. The authors of the link
  considered two other possibilities:
  \begin{itemize}
  \item Have AXIOM simulate the ASPs, so that the NAG Library would call
  back to AXIOM when it wanted to call an ASP. This was rejected as
  being far too slow across a network.
  \item Give NAGD the ability to interpret AXIOM or Fortran code. Thus
  the NAG routine would call a function which would evaluate a
  representation of an ASP to get the required values. This may happen
  in the future if data interchange mechanisms between systems are
  stanardised, but was rejected for the time being since such a system
  would have to be tailored to match each Fortran compiler that NAGD used.
  \end{itemize}

  By transmitting source code for ASPs we allow the remote Fortran
  compiler to take care of low-level portability problems.
```

```
    Both IRENA and the AXIOM-NAG Link make using the NAG Library easier in
    a functional kind of way. They present a more natural, interactive
    interface and, what is more, they cut out some of the tedium of
    writing code by generating values for some parameters (array
    dimensions etc.) and computing derivatives and
    jacobians. Unfortunately the user still has to choose which algorithm
    to use, and tuen the control parameters to give a result of the
    required accuracy.

    This is not a trivial task. The NAG Library has some thirty quadrature
    routines, and maby forty top-level ODE-solving routines. The criteria
    for selecting a routine can include features of the problem such as
    whether a function is continuous, or they can include requrements from
    the user such as whether speed or accuracy is important. The user
    should also think carefully about whether the problem is being
    presentin in the best way: perhaps an integral can be solved more
    easily if the range of integration is split into several pieces for
    example.

    Many users of numerical software would like to be presented with a
    ``black box'' for solving a given class of problems. In practice many
    users will always try solving a problem with the ``simplest'' method
    first, in the hope that they can escape having to think seriously
    about their problem! This is not always a bad idea anyway, since the
    feedback from the routine can help determine a better technique
    (e.g. by identifying the regions where problems occur).

    We contend that computer algebra has an important role to play in this
    area. The ability of algebra systems to analyse the symbolic features
    of a problem make them idea {\sl agents} for knowledge-based (or
    expert) systems to use in selecting routines. Their ability (via IRENA
    or AXIOM-NAG) to drive the numerical codes is a bonus.",

}
```

———-

— **ignore** —

```
\bibitem[Steele]{Steele} Steele, Guy L.; Gabriel, Richard P.
  title = {{The Evolution of Lisp}},
  link = "\url{http://www.dreamsongs.com/Files/HOPL2-Uncut.pdf}",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Sage 14]{Sage14} Stein, William
  title = {{Sage}},
  link = "\url{http://www.sagemath.org/doc/reference/interfaces/sage/interfaces/axiom.html}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Stei16,
  author = "Stein, William",
  title = {{Axiom as an OSCAS}},
  link = "\url{http://wiki.sagemath.org/Axiom\_as\_an\_OSCAS}",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Stor95,
  author = "Storme, L. and van Maldeghem, H.",
  title = {{Cyclic caps in PG(3,q)}},
  journal = "Geom. Dedicata",
  volume = "56",
  number = "3",
  pages = "271-284",
  year = "1995",
  link = "\url{https://cage.ugent.be/~hvm/artikels/44.pdf}",
  abstract =
    "A $k$-cap $K$ on $PG(n,q)$ is a set of $k$ points, no three of which
    are collinear. $K$ is complete if it cannot be extended to a $k+1$
    cap. If $K$ is invariant under a cyclic subgroup (which acts regularly
    on $K$) of $PGL(n+1,q)$, the $K$ is cyclic.

    This article investigates cyclic complete $k$-caps in
    $PG(3,q)$. Namely, the different types of complete $k$-caps $K$ in
    $PG(3,q)$ stabilized by a cyclic projective group $G$ of order $k$,
    acting regularly on the points of $K$, are determined. We show that in
    $PG(3,q)$, $q$ even, the elliptic quadric is the only cyclic complete
    $k$-cap. For $q$ odd, it is shown that besides the elliptic quadric,
    there also exist cyclic $k$-caps containing $k/2$ points of two
    disjoint elliptic quadrics or two disjoint hyperbolic quadrics and
    that there exist cyclic $k$-caps stabilized by a transitive cyclic
    group $G$ fixed precisely one point and one plane of
    $PG(3,q)$. Concrete examples of such caps, found using AXIOM and
    CAYLEY, are presented.",
  paper = "Stor95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Stro99,
  author = "Stroeker, Roelof J. and Kaashoek, Johan F.",
  title = {{Discovering mathematics with Maple. An interactive exploration for
           mathematicians, engineers and econometricians}},
  year = "1999",
  publisher = "Birkhauser",
  isbn = "0-8176-6091-7",
  abstract =
    "During the past decade, the mathematical computer software packages
    such as Mathematica, Maple, MATLAB (Axiom, Derive, Macsyma, MuPad are
    some further examples of such software) [see Macsyma 2.3. Lite  the
    student edition (1998; Zbl 0911.68089); B. W. Char, K. O. Geddes,
    G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt, Maple V
    Library reference manual (1991; Zbl 0763.68046); J. L. Zachary,
    Introduction to scientific programming. Computational problem solving
    using Mathematica and C (1997; Zbl 0891.68053); The student edition of
    MATLAB. Student user guide. The problem-solving tool for engineers,
    mathematicians, and scientists (1992; Zbl 0782.65001); H. Benker,
    Ingenieurmathematik mit Computeralgebra-Systemen. AXIOM, DERIVE,
    MACSYMA, MAPLE, MATHCAD, MATHEMATICA, MATLAB und MuPAD in der
    Anwendung (1998; Zbl 0909.68109); W. Koepf, Hohere Analysis mit DERIVE
    (1994; Zbl 0819.26003)] have greatly faciliated mathematical
    experiments and have thus become popular tools for the modern
    mathematician. It is a pity that most of these packages are quite
    expensive, and that the frequently upgraded versions are not free for
    the owners of the earlier versions (fortunately, there are inexpensive
    student versions of some of these packages). There is a constant
    demand of instructional textbooks by users of these packages. This
    demand is reflected in the growing number of such textbooks. Many of
    these books provide software support (diskette, CD-ROM, access by
    ftp). Such a textbook should meet, in my opinion, the following
    criteria: (1) The size should be small, not bulky like the complete
    technical descriptions of the software. (2) There should be a lot of
    examples of the use of the software covering a wide range of
    mathematical topics. Electronic versions of these examples should be
    made available for free to the users of the textbook
    (e.g. diskette/CD-ROM, access by ftp). (3) There should be a good
    supply of exercises covering the basic mathematical applications. (4)
    The book should be visually pleasing, easy to read, have good indexes
    and provide pointers to other books and electronic sources of
    information. The book under review provides, in addition to the actual
    text, an interactive exploratorium of its topics, based on the
    mechanism of Maple worksheets. These worksheets can be ‘‘opened’’ by
    the Maple program and they form a mixture of usual text, hypertext,
    and Maple commands and have a nice style appearance. They also can be
    ‘‘exported’’ in a file and included in a file for further treatment.
    The book meets all the aforementioned criteria (1)-(4) with elegance.
    There are many exercises which cover all the usual mathematical topics
    from linear algebra to differential equations and statistics. A
    valuable feature is an appendix with hints and answers for all
    exercises.  One of the highlights of the book is the examination of
    Riemann's non-differentiable function
    \[x \mapsto \sum_{k=1}^\infty{k^{-2}} sin(\pi kx)\]
    which is differentiable only at the rational points $p/q$ with $p$
```

```
    and $q$ odd and relatively prime, where its derivative is $-1/2$.

    The book is intended for students of mathematics, engineering
    sciences, and econometry. This book is an ideal guide for this purpose
    and it could probably be used along, without the bulky technical
    documentation of the Maple language. Note that Maple has a
    comprehensive on-line help program, which contains large parts of the
    original documentation.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Suto85,
  author = "Sutor, Robert S.",
  title = {{The Scratchpad II computer algebra language and system}},
  booktitle = "Research Contributions from the Euro. Conf. on Comp. Alg.",
  series = "Lecture Notes in Computer Science Volume 204",
  volume = "2",
  pages = "32-33",
  year = "1985",
  isbn = "0-387-15983-5 (vol. 1),0-387-15984-3 (vol. 2)",
  paper = "Suto85.pdf",
  keywords = "axiomref, DONE",
  beebe = "Sutor:1985:SIC"
}
```

———-

— **Sutor:1985:SIC** —

```
@InProceedings{Sutor:1985:SIC,
  author =       "R. S. Sutor",
  title =        {{The Scratchpad II Computer Algebra Language and
                 System}},
  crossref =     "Buchberger:1985:EEC",
  pages =        "32--33",
  year =         "1985",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.1.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— **axiom.bib** —

```
@article{Suto87,
  author = "Sutor, Robert S. and Jenks, Richard D.",
  title = {{The type inference and coercion facilities in the Scratchpad II
           interpreter}},
```

```
  journal = "SIGPLAN Notices",
  volume = "22",
  number = "7",
  pages = "56-63",
  year = "1987",
  isbn = "0-89791-235-7",
  comment = "IBM Research Report RC 12595 (\#56575)",
  abstract =
    "The Scratchpad II system is an abstract datatype programming
    language, a compiler for the language, a library of packages of
    polymorphic functions and parametrized abstract datatypes, and an
    interpreter that provides sophisticated type inference and coercion
    facilities. Although originally designed for the implementation of
    symbolic mathematical algorithms, Scratchpad II is a general purpose
    programming language . This paper discusses aspects of the
    implementation of the interpreter and how it attempts to provide a
    user friendly and relatively weakly typed front end for the strongly
    typed programming language.",
  paper = "Suto87.pdf",
  keywords = "axiomref, printed",
  beebe = "Sutor:1987:TICb"
}
```

———-

— Sutor:1987:TICb —

```
@InProceedings{Sutor:1987:TICb,
  author =       "R. S. Sutor and R. D. Jenks",
  title =        {{The Type Inference and Coercion Facilities in the
                 Scratchpad II Interpreter}},
  crossref =     "Wexelblat:1987:IIT",
  pages =        "56--63",
  year =         "1987",
  bibsource =    "/usr/local/src/bib/bibliography/Compiler/bevan.bib;
                 /usr/local/src/bib/bibliography/Misc/sigplan.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The Scratchpad II system is an abstract datatype
                 programming language, a compiler for the language, a
                 library of packages of polymorphic functions and
                 parameterized abstract datatypes, and an interpreter
                 that provides sophisticated type inference and coercion
                 facilities. Although originally designed for the
                 implementation of symbolic mathematical algorithms,
                 Scratchpad II is a general purpose programming
                 language. This paper discusses aspects of the
                 implementation of the interpreter and how it attempts
                 to provide a user friendly ad relatively weakly typed
                 front end for the strongly typed programming
                 language.",
  acknowledgement = ack-nhfb,
  checked =      "19940516",
  keywords =     "scratchpad",
```

```
  refs =         "8",
  subject =      "D.3.4 Software, PROGRAMMING LANGUAGES, Processors,
                 Interpreters \\ I.1.3 Computing Methodologies,
                 ALGEBRAIC MANIPULATION, Languages and Systems,
                 SCRATCHPAD \\ D.3.3 Software, PROGRAMMING LANGUAGES,
                 Language Constructs, Abstract data types",
}
```

———-

— **axiom.bib** —

```
@misc{Suto87b,
  author = "Sutor, Robert S.",
  title = {{The Scratchpad II Computer Algebra System.
           Using and Programming the Interpreter}},
  type = "IBM Course presentation slide deck",
  year = "1987",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Suto87c,
  author = "Sutor, Robert S. and Jenks, Richard D.",
  title = {{The type inference and coercion facilities in the
           Scratchpad II interpreter'}},
  type = "Research Report",
  number = "RC 12595 (\#56575)",
  institution = "Mathematical Sciences Department",
  address = "IBM Thomas J. Watson Research Center, Yorktown Heights, NY",
  year = "1987",
  abstract =
    "The Scratchpad II system is an abstract datatype programming language,
    a compiler for the language, a library of packages of polymorphic
    functions and parameterized abstract datatypes, and an interpreter
    that provides sophisticated type inference and coercion facilities.
    Although originally designed for the implementation of symbolic
    mathematical algorithms, Scratchpad II is a general purpose
    programming language. This paper discusses aspects of the
    implementation of the intepreter and how it attempts to provide a user
    friendly and relatively weakly typed front end for the strongly typed
    programming language.",
  paper = "Suto87c.pdf",
  keywords = "axiomref",
  beebe = "Sutor:1987:TICa"
}
```

———-

— **Sutor:1987:TICa** —

```
@TechReport{Sutor:1987:TICa,
  author =       "Robert S. Sutor and Richard D. Jenks",
  title =        {{The type inference and coercion facilities in the
                 Scratchpad II interpreter}},
  type =         "Research Report",
  number =       "RC 12595 (\#56575)",
  institution =  "IBM Thomas J. Watson Research Center",
  address =      "Yorktown Heights, NY, USA",
  pages =        "11",
  year =         "1987",
  bibdate =      "Sat Dec 30 08:25:26 MST 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  keywords =     "Abstract data types (Computer science); Programming
                 languages (Electronic computers)",
}
```

————-

— **ignore** —

```
\bibitem[Sutor 88]{Su88} Sutor, Robert S.
  title = {{A guide to programming in the scratchpad 2 interpreter}},
IBM Manual, March 1988
  keywords = "axiomref"
```

————-

— **axiom.bib** —

```
@misc{Swmath,
  author = "Unknown",
  title = {{Axiom}},
  link = "\url{https://www.swmath.org/software/63}",
  abstract =
    "Axiom is a general purpose Computer Algebra System (CAS). It is
    useful for research and developement of mathematical algorithms.
    It defines a strongly typed, mathematically correct type hierarchy.
    It has a programming language and a built-in compiler.",
  keywords = "axiomref"
}
```

————-

— **axiom.bib** —

```
@misc{Sympy,
  author = "Certik, Ondrej",
  link = "\url{https://github.com/sympy/sympy/wiki/SymPy-vs.-Axiom}",
```

```
  keywords = "axiomref"
}
```

———-

1.40.20 T

— **axiom.bib** —

```
@InProceedings{Thom01,
  author = "Thompson, Simon",
  title = {{Logic and dependent types in the Aldor Computer Algebra System}},
  booktitle = "Symbolic computation and automated reasoning",
  series = "CALCULEMUS 2000",
  year = "2001",
  location = "St. Andrews, Scotland",
  pages = "205-233",
  link =
    "\url{http://axiom-wiki.newsynthesis.org/public/refs/aldor-calc2000.pdf}",
  abstract =
    "We show how the Aldor type system can represent propositions of
    first-order logic, by means of the 'propositions as types'
    correspondence. The representation relies on type casts (using
    pretend) but can be viewed as a prototype implementation of a modified
    type system with {\sl type evaluation} reported elsewhere. The logic
    is used to provide an axiomatisation of a number of familiar Aldor
    categories as well as a type of vectors.",
  paper = "Thom01.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Tour98,
  author = "Touratier, Emmanuel",
  title = {{Etude du typage dans le syst\'eme de calcul scientifique Aldor}},
  comment = "Study of types in the Aldor scientific computation system",
  year = "1998",
  link =
    "\url{http://axiom-wiki.newsynthesis.org/public/refs/Aldor-T1998_04.pdf}",
  paper = "Tour98.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Tour95,
  author = "Tournier, Evelyne",
  title = {{Summary of organisation, history and possible future}},
  year = "1995",
  link = "\url{ftp://ftp.inf.ethz.ch/org/cathode/workshops/jan95/abstracts/tournier.ps}",
  paper = "Tour95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Tuom96,
  author = "Tuomela, Jukka",
  title = {{On the construction of arbitrary order schemes for the many
           dimensional wave equation}},
  journal = "BIT",
  volume = "36",
  number = "1",
  pages = "158-165",
  year = "1996",
  abstract =
    "The paper is devoted to a problem which was of an interest in the
    beginning of the theory of difference methods. The elementary
    constructed explicit high-order approximations for the wave equation
    (on the simplest cubic grid in space) assume that the solution is very
    smooth and that no boundary conditions are given. Stability is also
    understood in the simplest way (in $L_2$).",
  keywords = "axiomref"
}
```

———-

1.40.21 U

— **axiom.bib** —

```
@article{Uebe94,
  author = "Ueberberg, Johannes",
  title = {{Interactive Theorem Proving and Computer Algebra}},
  journal = "Lecture Notes in Computer Science",
  volume = "958",
  year = "1994",
  abstract =
    "Interactive Theorem Proving, ITP for short, is a new approach for the
    use of current computer algebra systems to support mathematicians in
    proving theorems. ITP grew out of a more general project -- called
    Symbolic Incidence Geometry -- which is concerned with the problem of
    the systematic use of the computer in incidence geometry.",
  paper = "Uebe94.pdf",
```

```
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Unkn13,
  author = "Unknown",
  title = {{Hindley-Milner Type Inference}},
  link = "\url{https://www7.in.tum.de/um/courses/seminar/sove/SS2013/final/hindley-milner.slides.pdf}",
  paper = "Unkn13.pdf",
  keywords = "printed"

}
```

———-

— **axiom.bib** —

```
@misc{Unkn16,
  author = "Unknown",
  title = {{Computer Algebra Systems}},
  link = "\url{http://www.mhtlab.uwaterloo.ca/courses/me755/web\_intro.pdf}",
  paper = "Unkn16.pdf"
}
```

———-

1.40.22 V

— **axiom.bib** —

```
@article{Hoev12,
  author = "van der Hoeven, Joris",
  title = {{Overview of the Mathemagix type system}},
  journal = "ASCM 2102",
  year = "2012",
  link = "\url{http://www.texmacs.org/joris/mmxtyping/mmxtyping.pdf}",
  abstract =
    "The goal of the {\tt MATHEMAGIX} project is to develop a new and free
    software for computer algebra and computer analysis, base on a
    strongly typed and compiled language. In this paper, we focus on the
    nderlying type system of this language, which allows for heavy
    overloading, including parameterized overloading with parameters in so
    called ‘‘categories’’. The exposition is informal and aims at giving
    the reader an overview of the main concepts, ideas and differences
    with existing languages. In a forthcoming paer, we intend to describe
    the formal semantics of the type system in more detail.",
  paper = "Hoev12.pdf",
```

```
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[van der Hoeven 14]{JvdH14} van der Hoeven, Joris
  title = {{Computer algebra systems and TeXmacs}},
  link = "\url{http://www.texmacs.org/tmweb/plugins/cas.en.html}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@article{Hoei94,
  author = "{van Hoeij}, M.",
  title = {{An algorithm for computing an integral basis in an algebraic
           function field}},
  journal = "Journal of Symbolic Computation",
  volume = "18",
  number = "4",
  year = "1994",
  pages = "353-363",
  issn = "0747-7171",
  abstract = "
    Algorithms for computing integral bases of an algebraic function field
    are implemented in some computer algebra systems. They are used e.g.
    for the integration of algebraic functions. The method used by Maple
    5.2 and AXIOM is given by Trager in [Trag84]. He adapted an algorithm
    of Ford and Zassenhaus [Ford, 1978], that computes the ring of
    integers in an algebraic number field, to the case of a function field.

    It turns out that using algebraic geometry one can write a faster
    algorithm. The method we will give is based on Puiseux expansions.
    One cas see this as a variant on the Coates' algorithm as it is
    described in [Davenport, 1981]. Some difficulties in computing with
    Puiseux expansions can be avoided using a sharp bound for the number
    of terms required which will be given in Section 3. In Section 5 we
    derive which denominator is needed in the integral basis. Using this
    result 'intermediate expression swell' can be avoided.

    The Puiseux expansions generally introduce algebraic extensions. These
    extensions will not appear in the resulting integral basis.",
  paper = "Hoei94.pdf",
  keywords = "axiomref",
  beebe = "vanHoeij:1994:ACI"
}
```

———-

— vanHoeij:1994:ACI —

```
@Article{vanHoeij:1994:ACI,
  author =       "M. van Hoeij",
  title =        {{An algorithm for computing an integral basis in an
                 algebraic function field}},
  journal =      j-J-SYMBOLIC-COMP,
  volume =       "18",
  number =       "4",
  pages =        "353--363",
  month =        oct,
  year =         "1994",
  CODEN =        "JSYCEH",
  ISSN =         "0747-7171 (print), 1095-855X (electronic)",
  ISSN-L =       "0747-7171",
  bibdate =      "Fri Dec 29 12:46:02 MST 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Algorithms for computing integral bases of an
                 algebraic function field are implemented in some
                 computer algebra systems. They are used e.g. for the
                 integration of algebraic functions. The method used by
                 Maple 5.2 and AXIOM is given by B. M. Trager (1984). He
                 adapted an algorithm of Ford and Zassenhaus (1978),
                 that computes the ring of integers in an algebraic
                 number field, to the case of a function field. It turns
                 out that using algebraic geometry one can write a
                 faster algorithm. The method we give is based on
                 Puiseux expansions. One can see this as a variant on
                 the Coates' algorithm as it is described by Davenport
                 (1981). Some difficulties in computing with Puiseux
                 expansions can be avoided using a sharp bound for the
                 number of terms required which are given. We derive
                 which denominator is needed in the integral basis.
                 Using this result 'intermediate expression swell' can
                 be avoided. The Puiseux expansions generally introduce
                 algebraic extensions.",
  acknowledgement = ack-nhfb,
  affiliation =  "Dept. of Math., Nijmegen Univ., Netherlands",
  classification = "C6130 (Data handling techniques); C7310 (Mathematics
                 computing)",
  fjournal =     "Journal of Symbolic Computation",
  link =  "\url{http://www.sciencedirect.com/science/journal/07477171}",
  keywords =     "Algebraic function field; Algebraic number field;
                 AXIOM; Computer algebra systems; Integral basis;
                 Intermediate expression swell; Maple 5.2; Puiseux
                 expansions",
  language =     "English",
  pubcountry =   "UK",
  thesaurus =    "Mathematics computing; Symbol manipulation",
}
```

———-

— **axiom.bib** —

```
@misc{Hoei08,
  author = "{van Hoeij}, Mark and Novocin, Andrew",
  title = {{A Reduction Algorithm for Algebraic Function Fields}},
  year = "2008",
  month = "April",
  link = "\url{http://andy.novocin.com/pro/algext.pdf}",
  abstract = "
    Computer algebra systesm often produce large expressions involving
    complicated algebraic numbers. In this paper we study variations of
    the {\tt polred} algorithm that can often be used to find better
    representations for algebraic numbers. The main new algorithm
    presented here is an algorithm that treats the same problem for the
    function field case.",
  paper = "Hoei08.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Vani96,
  author = "Vaninwegen, Myra",
  title = {{The Machine-Assisted Proof of Programming Language
           Properties}},
  year = "1996",
  school = "University of Pennsylvania",
  link = "\url{https://pdfs.semanticscholar.org/4ba2/6cbd3d1600aa82d556d76ce5531db9e8e940.pdf}",
  abstract =
    "The goals of thie project described in this thesis are
    twofold. First, we wanted to demonstrate that if a programming
    language has a semantics that is complete and rigorous
    (mathematical), but not too complex, then substantial theorems can
    be proved about it. Second, we wanted to assess the utiliity of
    using an automated theorem prover to aid in such proofs. We chose
    SML as the language about which to prove theorems: it has a
    published semantics that is complete and rigorous, and while not
    exactly simple, is comprehensible. We encoded the semantics of
    Core SML into the theorem prover HOL (creating new definitional
    packages for HOL in the process). We proved important theorems
    about evaluation and about the type system. We also proved the
    type preservation theorem, which relates evaluation and typing,
    for a good portion of the language. We were not able to complete
    the proof of type preservation because it is not tur: we found
    counterexamples. These proofs demonstrate that a good semantics
    will allow the proof of programming language properties and allow
    the identification of trouble spotes in the language. The use of
    HOL had its plusses and minuses. On the whole the benefits greatly
    outweigh the drawbacks, enough so that we believe that these
    theorems could ot have been proved in amount of time taken by this
    project had we not use automated help.",
  paper = "Vani96.pdf",
```

```
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Vapn20,
  author = "Vapnik, Vladimir",
  title = {{Predicates, Invariants, and the Essence of Intelligence}},
  year = "2020",
  link = "\url{https://lexfridman.com/vladimir-vapnik-2}",
  keywords = "DONE"
}
```

———-

— **ignore** —

```
\bibitem[Vasconcelos 99]{Vas99} Vasconcelos, Wolmer
  title = {{Computational Methods in Commutative Algebra and
           Algebraic Geometry}},
Springer, Algorithms and Computation in Mathematics, Vol 2 1999
ISBN 3-540-21311-2
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Voev16,
  author = "Voevodsky, Vladimir",
  title = {{Univalent Foundations of Mathematics}},
  year = "2016",
  link = "\url{https://www.youtube.com/watch?v=9f4sS9s-X2A}",
  keywords = "DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Voev17,
  author = "Voevodsky, Vladimir and Benedikt, Ahrens and Grayson, Daniel",
  title = {{UniMath: Univalent Mathematics}},
  link = "\url{https://github.com/UniMath/UniMath}",
  year = "2017"
}
```

———-

1.40.23 W

— axiom.bib —

```
@article{Wang89,
  author = "Wang, Dongming",
  title = {{A program for computing the Liapunov functions and Liapunov
           constants in Scratchpad II}},
  journal = "SIGSAM Bulletin",
  volume = "23",
  number = "4",
  pages = "25-31",
  year = "1989",
  abstract =
    "This report describes the implementation and use of a program for
    computing the Liapunov functions and Liapunov constants for a class
    of differential systems in Scratchpad II",
  paper = "Wang89.pdf",
  keywords = "axiomref",
  beebe = "Wang:1989:PCL"
}
```

———-

— Wang:1989:PCL —

```
@Article{Wang:1989:PCL,
  author =       "D. Wang",
  title =        {{A program for computing the {Liapunov} functions and
                 Liapunov constants in Scratchpad II}},
  journal =      j-SIGSAM,
  volume =       "23",
  number =       "4",
  pages =        "25--31",
  month =        oct,
  year =         "1989",
  CODEN =        "SIGSBZ",
  ISSN =         "0163-5824 (print), 1557-9492 (electronic)",
  ISSN-L =       "0163-5824",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/sigsam.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The report describes the implementation and use of a
                 program for computing the Liapunov functions and
                 Liapunov constants for a class of differential systems
                 in Scratchpad II.",
  acknowledgement = ack-nhfb,
  affiliation =  "Res. Inst. for Symbolic Comput., Johannes Kepler
                 Univ., Linz, Austria",
  classification = "C1320 (Stability); C4170 (Differential equations);
                 C7420 (Control engineering)",
  fjournal =     "SIGSAM Bulletin",
  keywords =     "Differential systems, design; Liapunov constants;
```

```
                Liapunov functions; performance; Scratchpad II",
  language =    "English",
  pubcountry =  "USA",
  subject =     "E.4 Data, CODING AND INFORMATION THEORY, Data
                compaction and compression \\ G.2.0 Mathematics of
                Computing, DISCRETE MATHEMATICS, General",
  thesaurus =   "Control system CAD; Differential equations; Lyapunov
                methods; Polynomials",
}
```

———-

— **axiom.bib** —

```
@article{Wang90,
  author = "Wang, Dongming",
  title = {{A Class of Cubic Differential Systems with 6-tuple Focus}},
  journal = "J. Differential Equations",
  publisher = "Academic Press, Inc.",
  volume = "87",
  pages = "305-315",
  year = "1990",
  abstract =
    "This paper presents a class of cubic differential systems with the
    origin as a 6-tuple focus from which 6 limit cycles may be
    constructed. For this class of differential systems the stability of
    the origin is given.",
  paper = "Wang90.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Wang91,
  author = "Wang, Dongming",
  title = {{Mechanical manipulation for a class of differential systems}},
  journal = "Journal of Symbolic Computation",
  volume = "12",
  number = "2",
  pages = "233-254",
  year = "1991",
  comment = "See Wang89 for the implementation code",
  abstract =
    "In this paper we describe a mechanical procedure for computing the
    Liapunov functions and Liapunov constants for a class of differential
    systems. These functions and constants are used for establishing the
    stability criteria, the conditions for the existence of a center and
    for the investigation of limit cycles. Some problems for handling the
    computer constants, which are usually large polynomials in terms of
    the coefficients of the differential system, and an approach towards
```

```
    their solution by using computer algebraic methods are proposed. This
    approach has been successfully applied to check some known results
    mechanically. The author has implemented a system DEMS on an HP1000
    and in Scratchpad II on an IBM4341 for computing and manipulating the
    Liapunov functions and Liapunov constants. As examples, two particular
    cubic systems are discussed in detail. The explicit algebraic
    relations between the computed Liapunov constants and the conditions
    given by Saharnikov are established, which leads to a rediscovery of
    the incompleteness of his conditions. A class of cubic systems with
    6-tuple focus is presented to demonstrate the feasibility of the
    approach for finding systems with higher multiple focus.",
  paper = "Wang91.pdf",
  keywords = "axiomref",
  beebe = "Wang:1991:MMC"
}
```

———-

— **Wang:1991:MMC** —

```
@Article{Wang:1991:MMC,
  author =       "Dongming Wang",
  title =        {{Mechanical manipulation for a class of differential
                 systems}},
  journal =      j-J-SYMBOLIC-COMP,
  volume =       "12",
  number =       "2",
  pages =        "233--254",
  month =        aug,
  year =         "1991",
  CODEN =        "JSYCEH",
  ISSN =         "0747-7171 (print), 1095-855X (electronic)",
  ISSN-L =       "0747-7171",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/cathode.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "The author describes a mechanical procedure for
                 computing the Liapunov functions and Liapunov constants
                 for a class of differential systems. These functions
                 and constants are used for establishing the stability
                 criteria, the conditions for the existence of a center
                 and for the investigation of limit cycles. Some
                 problems for handling the computer constants, which are
                 usually large polynomials in terms of the coefficients
                 of the differential system, and an approach towards
                 their solution by using computer algebraic methods are
                 proposed. This approach has been successfully applied
                 to check some known results mechanically. The author
                 has implemented a system DEMS on an HP1000 and in
                 Scratchpad II on an IBM4341 for computing and
                 manipulating the Liapunov functions and Liapunov
                 constants. As examples, two particular cubic systems
                 are discussed in detail. The explicit algebraic
```

```
                relations between the computed Liapunov constants and
                the conditions given by Saharnikov are established,
                which leads to a rediscovery of the incompleteness of
                his conditions. A class of cubic systems with 6-tuple
                focus is presented to demonstrate the feasibility of
                the approach for finding systems with higher multiple
                focus.",
  acknowledgement = ack-nhfb,
  affiliation =  "Res. Inst. for Symbolic Comput., Johannes Kepler
                Univ., Linz, Austria",
  classification = "C1320 (Stability); C4170 (Differential equations);
                C7420 (Control engineering)",
  fjournal =     "Journal of Symbolic Computation",
  link =  "\url{http://www.sciencedirect.com/science/journal/07477171}",
  keywords =     "6-Tuple focus; Computer algebraic methods; Cubic
                systems; DEMS; Differential systems; HP1000; IBM4341;
                Incompleteness; Large polynomials; Liapunov constants;
                Liapunov functions; Limit cycles; Limit cycles
                SCRATCHPAD, Nonlinear DEs; Mechanical procedure;
                Scratchpad II; Stability criteria",
  language =     "English",
  pubcountry =   "UK",
  thesaurus =    "Control system analysis computing; Lyapunov methods;
                Nonlinear differential equations; Stability; Symbol
                manipulation",
}
```

———-

— axiom.bib —

```
@misc{Wang95,
  author = "Wang, Dongming",
  title = {{Characteristic Sets and Zero Structure of Polynomial Sets}},
  institution = "Johannes Kepler University",
  comment = "Lecture Notes",
  link = "\url{http://www-polsys.lip6.fr/~wang/papers/CharSet.ps.gz}",
  abtract =
    "This paper provides a tutorial on the theory and method of
    characteristic sets and some relevant topics. The basic algorithms as
    well as their generalization for computing the characteristic set and
    characteristic series of a set of multivariate polynomials are
    presented. The characeristic set, which is of certain triangular form,
    reflects in general the major part of zeros, and the characteristic
    series, which is a sequence of polynomial sets of triangular form,
    furnishes a complete zero decomposition of the given polynomial
    set. Using this decomposition, a complete solution to the algebraic
    decision problem and a method for decomposing any algebraic variety
    into irreducible components are described. Some applications of the
    method are indicated.",
  paper = "Wang95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Wang01,
  author = "Wang, Dongming",
  title = {{Elimination Methods}},
  publisher = "Springer-Verlag",
  isbn = "978-3-7091-6202-6",
  year = "2001",
  abstract =
    "The development of polynomial-elimination techniques from classical
    theory to modern algorithms has undergone a tortuous and rugged
    path. This can be observed L. van der Waerden's elimination of the
    ''elimination theory'' chapter from from B. his classic Modern Algebra
    in later editions, A. Weil's hope to eliminate ''from algebraic
    geometry the last traces of elimination theory,'' and S. Abhyankar's
    suggestion to ''eliminate the eliminators of elimination theory.''
    The renaissance and recognition of polynomial elimination owe much to
    the advent and advance of modern computing technology, based on
    which effective algorithms are implemented and applied to diverse
    problems in science and engineering. In the last decade, both
    theorists and practitioners have more and more realized the
    significance and power of elimination methods and their underlying
    theories. Active and extensive research has contributed a great deal
    of new developments on algorithms and soft ware tools to the subject,
    that have been widely acknowledged. Their applications have taken
    place from pure and applied mathematics to geometric modeling and
    robotics, and to artificial neural networks. This book provides a
    systematic and uniform treatment of elimination algorithms that
    compute various zero decompositions for systems of multivariate
    polynomials. The central concepts are triangular sets and systems of
    different kinds, in terms of which the decompositions are
    represented. The prerequisites for the concepts and algorithms are
    results from basic algebra and some knowledge of algorithmic
    mathematics.",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Wang02,
  author = "Wang, Dongming",
  title = {{Epsilon: A Library of Software Tools for Polynomial Elimination}},
  booktitle = "Proc. 1st Int. Congress of Mathematical Software",
  series = "ICMS 2002",
  year = "2002",
  location = "Beijing China",
  pages = "379-389",
```

```
  link = "\url{https://hal.inria.fr/inria-00107607/file/A02-R-314.pdf}",
  abstract =
    "This article presents a Maple library of functions for decomposing
    systems of multivariate polynomials into triangular systems of
    various kinds (regular, simple, or irreducible), with an application
    package for manipulating and proving geometric theorems.",
  paper = "Wang02.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Wang 92]{Wan92} Wang, Paul S. (ed)
International System Symposium on Symbolic and
Algebraic Computation 92 ACM Press, New York, NY 10036, USA, 1992
ISBN 0-89791-489-9 (soft cover), 0-89791-490-2 (hard cover),
LCCN QA76.95.I59 1992
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Watanabe 90]{WN90} Watanabe, Shunro; Nagata, Morio; (ed)
ISSAC '90 Proceedings of the
International Symposium on Symbolic and Algebraic Computation ACM Press,
New York, NY, 10036, USA. 1990 ISBN 0-89791-401-5 LCCN QA76.95.I57 1990
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@phdthesis{Watt85a,
  author = "Watt, Stephen M.",
  title = {{Bounded Parallelism in Computer Algebra}},
  school = "University of Waterloo",
  year = "1985",
  link = "\url{http://www.csd.uwo.ca/~watt/pub/reprints/1985-smw-phd.pdf}",
  abstract =
    "This thesis examines the performance improvements that can be
    made by exploiting parallel processing in symbolic mathmatical
    computation. The study focuses on the use of high-level
    parallelism in the case where the number of processors is fixed
    and independent of the problem size, as in existing
    multiprocessors.

    Since seemingly small changes to the inputs can cause dramatic
    changes in the execution times of many algorithms in computer
```

```
    algebra, it is not generally useful to use static scheduling. We
    find it is possible, however, to exploit the high-level parallelism
    in many computer algebra problems using dynamic scheduling methods
    in which subproblems are treated homogeneously.

    Our investigation considers the reduction of execution time in
    both the case of AND-parallelism, where all of the subproblems
    must be completed, and the less well studied case of
    OR-parallelism, where completing any one of the subproblems is
    sufficient. We examine the use of AND and OR-parallelism in terms
    of the {\sl problem heap} and {\sl collusive} dynamic scheduling
    schemes which allow a homogeneous treatment of subtasks. A useful
    generalization is also investigated in which each of the subtasks
    may either succeed or fail and execution completes when the first
    success is obtained.

    We study certain classic problems in computer algebra within this
    framework. A collusive method for integer factorization is
    presented. This method attempts to find different factors in
    parallel, taking the first one that is discovered. Problem heap
    algorithms are given for the computation of multivariate
    polynomial GCDs and the computation of Grobner bases. The GCD
    algorithm is based on the sparse modular GCD and performs the
    interpolations in parallel. The Grobner basis algorithm exploits
    the independence of the reductions in basis completion to obtain a
    parallel method.

    In order to make evaluations in concrete terms, we have
    constructed a system for running computer algebra programs on a
    multiprocessor. The system is a version of Maple able to
    distribute processes over a local area network. The fact that the
    multiprocessor is a local area network need not be considered by
    the programmer.",
  paper = "Watt85a.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@misc{Watt86,
  author = "Watt, Stephen M. and Della Dora, J. and Wityak, Sandy (ed)",
  title = {{Algebra Snapshot: Linear Ordinary Differential Operators}},
  source = "Scratchpad II Newsletter: Vol 1 Num 2 (Jan 1986)",
  link = "\url{http://www.csd.uwo.ca/~watt/pub/reprints/1986-snews-lodo.pdf}",
  paper = "Watt86.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Watt 87]{Wat87} Watt, Stephen
  title = {{Domains and Subdomains in Scratchpad II}},
in [Wit87], pp3-5
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Watt 87a]{WB87} Watt, Stephen M.; Burge, William H.
  title = {{Mapping as First Class Objects}},
in [Wit87], pp13-17
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Watt87,
  author = "Watt, Stephen M. and Jenks, Richard D.",
  title = {{Abstract Datatypes, Multiple Views and Multiple Inheritance in
           Scratchpad II}},
  year = "1987",
  link = "\url{https://cs.uwaterloo.ca/~smatt/pub/reprints/1987-itl-spadviews.pdf}",
  abstract =
    "Scratchpad II is an abstract datatype language developed at Yorktown
    Heights for the implementation of a new computer algebra system. It
    provides packages of polymorphic functions and parameterized, abstract
    datatypes with operator overloading and multiple inheritance. To
    express the intricate inter-relationships between the datatypes
    necessary for the description of mathematical objects, a number of
    techniques based on the notion of {\sl category} have been
    used. Categories are used to enforce relationships between type
    parameters and to provide the mechanism for multiple inheritance. They
    also allow the language to be statically type checked and the
    generation of efficient code. This paper describes the role of
    categories in Scratchpad II.",
  paper = "Watt87.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Watt89,
  author = "Watt, Stephen M.",
  title = {{A fixed point method for power series computation}},
  booktitle = "Proc. ISSAC '88",
  series = "Lecture Notes in Computer Science 358",
```

```
  location = "Rome, Italy",
  pages = "206-217",
  isbn = "3-540-51084-2",
  year = "1988",
  abstract =
    "This paper presents a novel technique for manipulating structures
    which represents infinite power series.

    When power series are implemented using lazy evaluation, many
    operations can be written as simple recursive procedures. For example,
    the programs to generate the series for the elementary transcendental
    functions are almost transliterations of the defining integral
    equations. However, a naive lazy algorithm provides an implementation
    which may be orders of magnitude slower than a method which
    manipulates the coefficients explicitly.

    The technique described here allows a power series to be defined in a
    very natural but computationally inefficient way and transforms it to
    an equivalent, efficient form. This is achieved by using a fixed point
    operator on the delayed part to remove redundant calculations.

    This paper describes this fixed point method and the class of problems
    to which it is applicable. It has been used in Scratchpad II to
    improve the performance of a number of operations on infinite series,
    including division, reversion, special functions and the solution of
    linear and non-linear ordinary differential equations.

    A few examples are given of the method and of the speed up
    obtained. To illustrate, the computation of the first $n$ terms of
    exp($u$) for a dense, infinite series $u$ is reduced from $O(n^4)$ to
    $O(n^2)$ coefficient operations, the same as required by the standard
    on-line algorithms.",
  paper = "Watt88.pdf",
  keywords = "axiomref, printed",
  beebe = "Watt:1989:FPM"
}
```

———-

— Watt:1989:FPM —

```
@InProceedings{Watt:1989:FPM,
  author =       "S. M. Watt",
  title =        {{A fixed point method for power series computation}},
  crossref =     "Gianni:1989:SAC",
  pages =        "206--217",
  month =        "",
  year =         "1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Presents a novel technique for manipulating structures
                 which represent infinite power series. The technique
                 described allows a power series to be defined in a very
```

```
                natural but computationally inefficient way and
                transforms it to an equivalent, efficient form. This is
                achieved by using a fixed point operator on the delayed
                part to remove redundant calculations. The paper
                describes this fixed point method and the class of
                problems to which it is applicable. It has been used in
                Scratchpad II to improve the performance of a number of
                operations on infinite series, including division,
                reversion, special functions and the solution of linear
                and non-linear ordinary differential equations. A few
                examples are given of the method and of the speed up
                obtained. To illustrate, the computation of the first n
                terms of exp(u) for a dense, infinite series u is
                reduced from O(n/sup 4/) to O(n/sup 2/) coefficient
                operations, the same as required by the standard
                on-line algorithms.",
  acknowledgement = ack-nhfb,
  affiliation =  "IBM Thomas J. Watson Res. Center, Yorktown Heights,
                NY, USA",
  classification = "C4240 (Programming and algorithm theory); C7310
                (Mathematics)",
  keywords =     "Delayed part; Fixed point method; Fixed point
                operator; Infinite power series; Power series
                computation; Redundant calculations; Scratchpad II",
  language =     "English",
  thesaurus =    "Computational complexity; Mathematics computing",
}
```

———-

— **axiom.bib** —

```
@inproceedings{Watt90,
  author = "Watt, Stephen M. and Jenks, Richard D. and Sutor, Robert S. and
            Trager, Barry M.",
  title = {{The Scratchpad II type system: Domains and Subdomains}},
  booktitle = "Computing Tools for Scientific Problem Solving",
  year = "1990",
  publisher = "Academic Press",
  link = "\url{https://cs.uwaterloo.ca/~smwatt/pub/reprints/1990-miola-spadtypes.pdf}",
  abstract =
    "Scratchpad II is a language developed at Yorktown Heights for the
    implementation of a new computer algebra system. The need to model the
    intricate relationships among the datatypes representing mathematical
    objects has provided a number of challenges in the design of a type
    system for the programming language.

    In languages in which a datatype constructor may take multiple
    parameters, ensuring compatibility between them is extremely
    important. Scratchpad II addresses this issue by basing its
    implementation of abstract datatypes on {\sl categories}. Categories
    provide a convenient and useful method for specifying requirements on
    operations from datatypes. These requirements can be very complex when
```

```
    modelling mathematics.

    We show how categories provide multiple inheritance and how
    inheritance of specification is separated from inheritance of
    implementation. We also present implications of the type system on
    compilation of efficient code and flexibility of a weakly typed
    interactive user interface.

    Finally, the mechanisms of Scratchpad II are compared with those of
    traditional abstract datatype and object-oriented programming
    languages.",
  paper = "Watt90.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Watt 91]{Wat91} Watt, Stephen M. (ed)
Proceedings of the 1991 International Symposium on
Symbolic and Algebraic Computation, ISSAC'91, July 15-17, 1991, Bonn, Germany,
ACM Press, New York, NY 10036, USA, 1991 ISBN 0-89791-437-6
LCCN QA76.95.I59 1991
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Watt 94a]{Wat94a} Watt, Stephen M.; Dooley, S.S.; Morrison, S.C.;
Steinback, J.M.; Sutor, R.S.
  title = {{A\# User's Guide}},
Version 1.0.0 O($\epsilon{}^1$) June 8, 1994
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@techreport{Watt94,
  author = "Watt, Stephen M. and Broadbery, Peter A. and Dooley, Samuel S.
            and Iglio, Pietro",
  title = {{A First Report on the A\# Compiler (including benchmarks)}},
  institution = "IBM Research",
  year = "1994",
  type = "technical report",
  number = "RC19529 (85075)",
  link = "\url{http://www.aldor.org/docs/reports/i94acomp/i94acomp.pdf}",
  abstract =
    "The $A^{\#}$ compiler allows users of computer algebra to develop
```

```
    programs in a context where multiple programming languages are
    employed. The compiler translates programs written in the $A^{\#}$
    programming language to a low-level intermediate language, Foam,
    from which it can generate stand-alone programs, native object
    libraries to be linked with other applications, or code to be read
    into closed environments. In addition, Foam code may be directly
    executed using an interpreter provided with the $A^{\#}$ compiler.

    The $A^{\#}$ programming language provides support for object-oriented
    and functional programming styles. It is ``higher-order'' in the sense
    that both types and functions are first class, and may be manipulated
    in the same ways as any other values. The primary considerations in
    the formulation of the language have been generality, composibility,
    and efficiency. The language has been designed to admit a number of
    important optimizations, allowing compilation to machine code which is
    in many instances of efficiency comparable to that produced by a C or
    Fortran compiler.

    The original motivation for $A^{\#}$ comes from the field of computer
    algebra: to provide an improved extension language for the Axiom
    computer algebra system.",
  paper = "Watt94.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Watt 94c]{Wat94c} Watt, Stephen M.
  title = {{A\# Language Reference Version 0.35}},
IBM Research Division Technical Report RC19530 May 1994
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Watt94a,
  author = "Watt, S.M. and Broadbery, P.A. and Dooley, S.S. and  Iglio, P.
            and Steinbach, J.M. and Morrison, S.C. and Sutor, R.S.",
  title = {{AXIOM Library Compiler Users Guide}},
  publisher = "The Numerical Algorithms Group (NAG) Ltd, 1994",
  year = "1994",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Watt95a,
  author = "Watt, Stephen M.",
  title = {{The A\# Programming Language and Compiler}},
  year = "1995",
  link = "\url{ftp://ftp.inf.ethz.ch/org/cathode/workshops/march93/Watt.tex}",
  paper = "Watt95a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Watt01,
  author = "Watt, Stephen M. and  Broadbery, Peter A. and Iglio, Pietro and
            Morrison, Scott C. and Steinbach, Jonathan M",
  title = {{FOAM: A First Order Abstract Machine Version 0.35}},
  year = "2001",
  paper = "Watt01.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Watt00,
  author = "Watt, Stephen M.",
  title = {{Aldor: The language and recent directions}},
  year = "2000",
  institution = "University of Western Ontario",
  link = "\url{http://www.aldor.org/docs/reports/sa2000/aldortalk-sa2000.pdf}",
  paper = "Watt00.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Watt00a,
  author = "Watt, Stephen M.",
  title = {{Aldor: An Introduction to the Language}},
  year = "2000",
  institution = "University of Western Ontario",
  link = "\url{http://www.aldor.org/docs/reports/ukqcd-2000/intro1-ukqcd00.pdf}",
  paper = "Watt00a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Watt00b,
  author = "Watt, Stephen M.",
  title = {{Aldor: Interfaces}},
  year = "2000",
  institution = "University of Western Ontario",
  link = "\url{http://www.aldor.org/docs/reports/ukqcd-2000/intro2-ukqcd00.pdf}",
  paper = "Watt00b.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Watt07,
  author = "Watt, Stephen",
  title = {{What Happened to Languages for Symolic Mathematical Computation?}},
  booktitle = "Proc. Prog. Lang. for Mechanized Mathematics",
  series = "PLMMS 07",
  year = "2007",
  location = "RISC-Linz, Austria",
  pages = "81-90",
  link = "\url{http://www.csd.uwo.ca/~watt/pub/reprints/2007-plmms-what\_happened.pdf}",
  abstract =
    "While the state of the art is relatively sophisticated in programming
    language support for computer algebra, there has been less development
    in programming language support for symbolic computation over the past
    two decades. We summarize certain advances in programming languages
    for computer algebra and propose a set of directions and challenges
    for programming languages for symbolic computation.",
  paper = "Watt07.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Webe93b,
  author = "Weber, Andreas",
  title = {{Type Systems for Computer Algebra}},
  school = "University of Tubingen",
  year = "1993",
  abstract =
    "We study type systems for computer algebra systems, which frequently
    correspond to the ‘‘pragmatically developed’’ typing constructs used
    in AXIOM.

    A central concept is that of type classes which correspond to AXIOM
    categories. We will show that types can be syntactically described as
```

terms of a regular order-sorted signature if no type parameters are allowed. Using results obtained for the functional programming language Haskell we will show that the problem of type inference is decidable. This result still holds if higher-order functions are present and parametric polymorphism is used. These additional typing constructs are useful for further extensions of existing computer algebra systems: These typing concepts can be used to implement category theoretic constructs and there are many well known constructive interactions between category theory and algebra.

On the one hand we will show that there are well known techniques to specify many important type classes algebraically, and we will also show that a formal and algorithmically Feasible treatment of the interactions of algebraically specified data types and type classes is possible. On the other hand we will prove that there are quite elementary examples arising in computer algebra which need very ''strong'' formalisms to be specified and are thus hard to handle algorithmically.

We will show that it is necessary to distinguish between types and elements as parameters of parameterized type classes. The type inference problem for the former remains decidable whereas for the latter it becomes undecidable. We will also show that such a distinction can be made quite naturally.

Type classes are second-order types. Although we will show that there are constructions used in mathematics which imply that type classes have to become first-order types in order to model the examples naturally, we will also argue that this does not seem to be the case in areas currently accessible for an algebra system. We will only sketch some systems that have been developed during the last years in which the concept of type classes as first-order types can be expressed. For some of these systems the type inference problem was proven to be undecidable.

Another fundamental concept for a type system of a computer algebra system at least for the purpose of a user interface are coercions. We will show that there are cases which can be modeled by coercions but not by an ''inheritance mechanism'', i. e. the concept of coercions is not only orthogonal to the one of type classes but also to more general formalisms as are used in object-oriented languages. We will define certain classes of coercions and impose conditions on important classes of coercions which will imply that the meaning of an expression is independent of the particular coercions that are used in order to type it.

We shall also impose some conditions on the interaction between polymorphic operations defined in type classes and coercions that will yield a unique meaning of an expression independent of the type which is assigned to it if coercions are present there will very frequently be several possibilities to assign types to expressions.

Often it is not only possible to coerce one type into another but it will be the case that two types are actually isomorphic . We will

```
    show that isomorphic types have properties that cannot be deduced from
    the properties of coercions and will shortly discuss other possibilities
    to model type isomorphisms.  There are natural examples of type
    isomorphisms occurring in the area of computer algebra that have
    a ``problematic'' behavior. So we will prove for a certain example that
    the type isomorphisms cannot be captured by a finite set of coercions
    by proving that the naturally associated equational theory is not
    finitely axiomatizable.

    Up to now few results are known that would give a clear dividing line
    between classes of coercions which have a decidable type inference
    problem and classes for which type inference becomes undecidable.
    We will give a type inference algorithm for some important classes of
    coercions.

    Other typing constructs which are again quite orthogonal to the
    previous ones are those of {\sl partial functions} and of types
    {\sl depending on elements} . We will link the treatment of {\sl partial
    functions} in AXIOM to the one used in order-sorted algebras and will
    show some problems which arise if a seemingly more expressive solution
    were used. There are important cases in which {\sl types depending on
    elements} arise naturally.  We will show that not only type inference
    but even type checking is undecidable for relevant cases occurring in
    computer algebra.",
  paper = "Webe93b.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **ignore** —

```
\bibitem[Weber 92b]{Webe92b} Weber, Andreas
  title = {{Structuring the Type System of a Computer Algebra System}},
  link = "\url{http://cg.cs.uni-bonn.de/personal-pages/weber/publications/pdf/WeberA/Weber92a.pdf}",
  abstract = "
    Most existing computer algebra systems are pure symbol manipulating
    systems without language support for the occuring types. This is
    mainly due to the fact taht the occurring types are much more
    complicated than in traditional programming languages. In the last
    decade the study of type systems has become an active area of
    research. We will give a proposal for a type system showing that
    several problems for a type system of a symbolic computation system
    can be solved by using results of this research. We will also provide
    a variety of examples which will show some of the problems that remain
    and that will require further research.",
  paper = "Webe92b.pdf",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Webe05,
  author = "Weber, Andreas",
  title = {{A Type-Coercion Problem in Computer Algebra}},
  booktitle = "Artificial Intelligence and Symbolic Mathematical Computing",
  series = "Lecture Notes in Computer Science 737",
  year = "2005",
  publisher = "Springer",
  pages = "188-194",
  abstract =
    "An important feature of modern computer algebra systems is the
    support of a rich type system with the possibility of type inference.

    Basic features of such a system are polymorphism and coercion between
    types. Recently the use of order-sorted rewrite systems was proposed
    as a general framework.

    We will give a quite simple example of a family of types arising in
    computer algebra whose coercion relations cannot be captured by a
    finite set of first-order rewrite rules.",
  paper = "Webe05.pdf",
  keywords = "axiomref, coercion, printed"
}
```

———-

— **ignore** —

```
\bibitem[Weber 93b]{Webe93b} Weber, Andreas
  title = {{Type Systems for Computer Algebra}},
  link = "\url{http://cg.cs.uni-bonn.de/personal-pages/weber/publications/pdf/WeberA/Weber93b.pdf}",
  abstract = "
    We study type systems for computer algebra systems, which frequently
    correspond to the ‘‘pragmatically developed’’ typing constructs used
    in AXIOM.  A central concept is that of {\sl type classes} which
    correspond to AXIOM categories. We will show that types can be
    syntactically described as terms of a regular order-sorted signature
    if no type parameters are allowed. Using results obtained for the
    functional programming language Haskell we will show that the problem
    of {\sl type inference} is decidable. This result still holds if
    higher-order functions are present and {\sl parametric polymorphism}
    is used. These additional typing constructs are useful for further
    extensions of existing computer algebra systems: These typing concepts
    can be used to implement category theoretic constructs and there are
    many well known constructive interactions between category theory and
    algebra.",
  paper = "Webe93b.pdf",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@Inproceedings{Webe94,
  author = "Weber, Andreas",
  title = {{Algorithms for Type Inference with Coercions}},
  booktitle = "Proc ISSAC 94",
  series = "ISSAC 94",
  pages = "324-329",
  year = "1994",
  abstract =
    "This paper presents algorithms that perform a type inference for a
    type system occurring in the context of computer algebra. The type
    system permits various classes of coercions between types and the
    algorithms are complete for the precisely defined system, which can be
    seen as a formal description of an important subset of the type system
    supported by the computer algebra program Axiom.

    Previously only algorithms for much more restricted cases of coercions
    have been described or the frameworks used have been so general that
    the corresponding type inference problems were known to be
    undecidable.",
  paper = "Webe94.pdf",
  keywords = "axiomref, coercion"
}
```

———-

— axiom.bib —

```
@article{Webe95,
  author = "Weber, Andreas",
  title = {{On coherence in computer algebra}},
  journal = "J. Symb. Comput.",
  volume = "19",
  number = "1-3",
  pages = "25-38",
  year = "1995",
  link = "\url{http://cg.cs.uni-bonn.de/personal-pages/weber/publications/pdf/WeberA/Weber94e.pdf}",
  abstract =
    "Modern computer algebra systems (e.g. AXIOM) support a rich type
    system including parameterized data types and the possibility of
    implicit coercions between types. In such a type system it will be
    frequently the case that there are different ways of building
    coercions between types. An important requirement is that all
    coercions between two types coincide, a property which is called {\sl
    coherence}. We will prove a coherence theorem for a formal type system
    having several possibilities of coercions covering many important
    examples. Moreover, we will give some informal reasoning why the
    formally defined restrictions can be satisfied by an actual system.",
  paper = "Webe95.pdf",
  keywords = "axiomref, coercion, printed",
  beebe = "Weber:1993:CCA"
}
```

———-

— **Weber:1993:CCA** —

```
@InProceedings{Weber:1993:CCA,
  author =       "A. Weber",
  title =        {{On coherence in computer algebra}},
  crossref =     "Miola:1993:DIS",
  pages =        "95--106",
  month =        "",
  year =         "1993",
  bibdate =      "Fri Dec 29 12:46:02 MST 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  abstract =     "Modern computer algebra systems (e.g. AXIOM) support a
                 rich type system including parameterized data types and
                 the possibility of implicit coercions between types. In
                 such a type system it will be frequently the case that
                 there are different ways of building coercions between
                 types. An important requirement is that all coercions
                 between two types coincide, a property which is called
                 coherence. The author proves a coherence theorem for a
                 formal type system having several possibilities of
                 coercions covering many important examples. Moreover,
                 he gives some informal reasoning why the formally
                 defined restrictions can be satisfied by an actual
                 system.",
  acknowledgement = ack-nhfb,
  affiliation =  "Wilhelm-Schickard-Inst. fur Inf., Tubingen Univ.,
                 Germany",
  classification = "C4210 (Formal logic); C4240 (Programming and
                 algorithm theory); C7310 (Mathematics)",
  keywords =     "AXIOM; Coherence; Coherence theorem; Computer algebra;
                 Formal type system; Informal reasoning; Parameterized
                 data types; Type system",
  language =     "English",
  thesaurus =    "Symbol manipulation; Theorem proving; Type theory",
}
```

———-

— **ignore** —

```
\bibitem[Weber 96]{Webe96} Weber, Andreas
  title = {{Computing Radical Expressions for Roots of Unity}},
  link = "\url{http://cg.cs.uni-bonn.de/personal-pages/weber/publications/pdf/WeberA/Weber96a.pdf}",
  abstract = "
    We present an improvement of an algorithm given by Gauss to compute a
    radical expression for a $p$-th root of unity. The time complexity of
    the algorithm is $O(p^3m^6log p)$, where $m$ is the largest prime
    factor of $p-1$.",
  paper = "Webe96.pdf",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Weber 99]{Webe99} Weber, Andreas
  title = {{Solving Cyclotomic Polynomials by Radical Expressions}},
  link = "\url{http://cg.cs.uni-bonn.de/personal-pages/weber/publications/pdf/WeberA/WeberKeckeisen99a.pdf}",
  abstract = "
    We describe a Maple package that allows the solution of cyclotomic
    polynomials by radical expressions. We provide a function that is an
    extension of the Maple {\sl solve} command. The major algorithmic
    ingredient of the package is an improvement of a method due to Gauss
    which gives radical expressions for roots of unity. We will give a
    summary for computations up to degree 100, which could be done within
    a few hours of cpu time on a standard workstation.",
  paper = "Webe99.pdf",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Wei-Jiang 12]{WJ12} Wei-Jiang
  title = {{Top free algebra System}},
  link = "\url{http://wei-jiang.com/it/software/top-free-algebra-system-bye-mathematica-bye-maple}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Westxx,
  author = "Wester, Michael J.",
  title = {{Computer Algebra Synonyms}},
  link = "\url{http://math.unm.edu/~wester/cas/synonyms.pdf}",
  abstract =
    "The following is a collection of synonyms for various operations in
    the seven general purpose computer algebra systems {\bf Axiom}, {\bf
    Derive}, {\bf Macsyma}, {\bf Maple}, {\bf Mathematica}, {\bf MuPAD},
    and {\bf Reduce}. This collection does not attempt to be
    comprehensive, but hopefully it will be useful in giving an indication
    of how to translate between the syntaxes used by the different systems
    in many common situations. Note that for a blank entry means that
    there is no exact translation of a particular operation for the
    indicated system, but it may still be possible to work around this
    lack with a related functionality.",
  paper = "Westxx.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{West99,
  author = "Wester, Michael J.",
  title = {{Computer Algebra Systems. A practical guide}},
  year = "1999",
  publisher = "Wiley",
  isbn = "0-471-98353-5",
  abstract =
    "In this book some of the most popular general purpose computer
    algebra systems (CAS), such as Mathematica, Maple, Derive, Axiom,
    MuPAD, and Macsyma, are examined. The strengths and weaknesses of
    these programs are compared and contrasted, and tutorial information
    for using these systems in various ways is given. The different
    packages are quantitatively compared using standard test suites,
    giving the possibility to asses the most appropriate for a particular
    user or application. The origins of these systems are revealed and
    many of their behaviors analyzed. This furnishes a feel for where the
    current computer algebra system state of the art stays and what can be
    expected for existing and future systems. The book is organized in
    several chapters written by different authors. Chapters 1,2, and 3 are
    organized as reviews, comparisons, and critiques of CAS
    capabilities. Then more technical issues are discussed considering
    different approaches taken by different CAS: simplifying square roots
    of square roots by denesting (chapter 4), complex number calculation
    (chapter 5), efficiently computing Chebyshev polynomials (chapter 6),
    solving single equations and systems of polynomial equations (chapters
    7, 8), computing limits (chapter 9), multiple integration (chapter
    10), solving ordinary differential equation (chapter 11), integration
    of nonlinear evolution equations (chapter 12), code generation
    (chapter 13), evaluation of expressions and programs in the embedded
    computer algebra programming language (chapter 14), and computer
    algebra in education (chapter 15). Chapter 16 covers the origin of CA,
    and, finally chapter 17 gives a list of most CAS available today.",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Wexelblat 87]{Wex87} Wexelblat, Richard L. (ed)
Proceedings of the SIGPLAN '87 Symposium on
Interpreter and Interpretive Techniques, St. Paul, Minnesota, June 24-26, 1987
ACM Press, New York, NY 10036, USA, 1987 ISBN 0-89791-235-7
LCCN QA76.7.S54 v22:7 SIGPLAN Notices, vol 22, no 7 (July 1987)
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{Wikixx,
```

```
  author = "Unknown",
  title = {{List of open-source software for mathematics}},
  link = "\url{https://en.wikipedia.org/wiki/List\_of\_open-source\_software\_for\_mathematics}",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@InProceedings{Will95,
  author = "Williamson, Clifton J.",
  title = {{On the algebraic construction of tri-diagonal matrices with given
           characteristic polynomial}},
  booktitle = "4th Conf. of Canadian Number Theory Association",
  year = "1995",
  location = "Halifax, Nova Scotia, Canada",
  pages = "417-431",
  abstract =
    "Let $K$ be a field of characteristic zero and assume
    \[f(x)=x^n-s_1x^{n-1}+\cdots+s_n \in L[x]\]
    where $L=K(s_1,\cdots,s_n)$. Call, for the purposes of this review, an
    $n\times n$ matrix 1-tridiagonal if its entries above and below the
    main diagonal are all 1, the entries on it are $d_1,\cdots,d_n$, and
    all other entries are 0. Under which conditions can
    $d_1,d_2,\cdots,d_n$ be chosen in a radical extension of $L$ such that
    the resulting 1-tridiagonal matrix has $f$ as its characteristic
    polynomial?  The authors answer: for $n=3$ always. The $d_i,s_i$
    are related by a system of algebraic equations of the form
    $\overline{f}_i(d_1,d_2,d_3)=s_i$. If this is satisfied, then by
    a resultant- or Groebner basis computation with respect to the
    lexicographic order, $d_3$ (say) must be a root of a certain
    irreducible polynomial $\varphi$ of degree 6. Joining this
    with a Galois group computation for $\varphi$ over $L$, it is shown
    that a solvable and necessarily transitive Gal($\varphi$) will be a
    necessary and sufficient condition for expressibility of $d_3$ (clear)
    and, thanks to the form of the Groebner basis, also of $d_1,d_2$ in
    radicals. Upon a discriminant computation of $\varphi$ and use of
    results of G. Butler and J. McKay [Commun. Algebra 11,
    863-911 (1983; Zbl 0518.20003)] and J. McKay and L. Soicher [J. Number
    Theory 20, 273-281 (1985; Zbl 0579.12006)], this allows the author to
    deduce the mentioned result. For $n=4$, though the author cannot get as
    complete information about Galois groups etc., he can deduce e.g. that
    if $s_1,\cdots,s_4$ are algebraically independent over $\mathbb{Q}$,
    then $d_1,\cdots,d_4$ are not expressible in radicals over
    $\mathbb{Q}(s_1,\cdots,s_4)$. He mentions, however, a solvable subcase.",
  paper = "Will95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wink85,
  author = "Winkler, Franz",
  title = {{Reducing the Complexity of the Knuth-Bendix Completion
            Algorithm: A ``Unification'' of Different Approaches}},
  booktitle = "European Conference on Computer Algebra (EUROCAL 85)",
  pages = "378-389",
  publisher = "Springer",
  year = "1985",
  isbn = "978-3-540-15983-4",
  abstract =
    "The Knuth-Bendix completion procedure for rewrite rule systems is
    of wide applicability in symbolic and algebraic computation.
    Attempts to reduce the complexity of this completion algorithm are
    reported in the literature. Already in their seminal 1967 paper
    D.E. Knuth and P.B. Bendix have suggested to keep all the rules
    iterreduced during the execution of the algorithm. G. Huet has
    presented a version of the completion algorithm in which every
    rewrite rule is kept in reduced form with respect to all the other
    rules in the system. Borrowing an idea of Buchberger's for the
    completion of bases of polynomial ideals the author has proposed
    in 1983 a criterion for detecting ``unnecessary'' critical
    pairs. If a critical pair is recognized as unnecessary then one
    need not apply the costly process of computing normal forms to
    it. It has been unclear whether these approaches can be
    combined. We demonstrate that it is possible to keep all the
    rewrite rules interreduced and still use a criterion for
    eliminating unnecessary critical pairs.",
  paper = "Wink85.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Wink89,
  author = "Winkler, Franz",
  title = {{Equational Theorem Proving and Rewrite Rule Systems}},
  year = "1989",
  publisher = "Springer-Verlag",
  link = "\url{http://www.risc.jku.at/publications/download/risc_3527/paper_47.pdf}",
  abstract =
    "Equational theorem proving is interesting both from a mathematical
    and a computational point of view. Many mathematical structures like
    monoids, groups, etc. can be described by equational axioms. So the
    theory of free monoids, free groups, etc. is the equational theory
    defined by these axioms. A decision procedure for the equational
    theory is a solution for the word problem over the associated
    algebraic structure. From a computational point of view, abstract data
    types are basically described by equations. Thus, proving properties
    of an abstract data type amounts to proving theorems in the associated
    equational theory.
```

```
    One approach to equational theorem proving consists in associating a
    direction with the equational axioms, thus transforming them into
    rewrite rules. Now in order to prove an equation $a=b$, the rewrite
    rules are applied to both sides, finally yielding reduced versions
    $a^{'}$ and $b^{'}$ of the left and right hand sides, respectively. If
    $a^{'}$ and $b^{'}$ agree syntactically, then the equation holds in
    the equational theory. However, in general this argument cannot be
    reversed; $a^{'}$ and $b^{'}$ might be different even if $a=b$ is a
    theorem. The reason for this problem is that the rewrite system might
    not have the Church-Rosser property. So the goal is to take the
    original rewrite system and transform it into an equivalent one which
    has the desired Church-Rosser property.

    We show how rewrite systems can be used for proving theorems in
    equational and inductive theories, and how an equational specification
    of a problem can be turned into a rewrite program.",
  paper = "Wink89.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inbook{Wink90,
  author = "Winkler, Franz",
  title = {{Solution of Equations I: Polynomial Ideals and Grobner Bases}},
  booktitle = "Computers in Mathematics",
  year = "1990",
  publisher = "Marcel Dekker, Inc",
  pages = "383-407",
  isbn = "0-8247-8341-7",
  paper = "Wink90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Wink93a,
  author = "Winkler, Franz",
  title = {{Computer Algebra}},
  booktitle = "Concise Encyclopedia of Computer Science",
  year = "1993",
  publisher = "Wiley",
  isbn = "0-470-09095-2",
  pages = "227-231",
  paper = "Wink93a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Wink93,
  author = "Winkler, Franz",
  title = {{Algebraic Computation in Geometry}},
  year = "1993",
  journal = "IMACS Symposium SC-1993",
  link =
  "\url{http://www.risc.jku.at/publications/download/risc_3777/paper_35.pdf}",
  abstract =
    "Computation with algebraic curves and surfaces are very well suited
    for being treated with computer algebra. Many aspects of computer
    algebra need to be combined for successfully solving problems in this
    area, e.g. computations with algebraic coefficients, solution of
    algebraic equations and elimination theory, and derivation of power
    series approximations of branches. We will describe the application of
    computer algebra to problems arising in algebraic geometry. The
    program system CASA, which has been developed by the author and a
    group of students will be introduced.",
  paper = "Wink93.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wink95,
  author = "Winkler, Franz",
  title = {{Computer Algebra - Problems and Developments}},
  booktitle = "Computers in Industry 2 (SCAFI'92)",
  pages = "1-17",
  year = "1995",
  isbn = "978-0471955290",
  paper = "Wink95.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Wink96,
  author = "Winkler, Franz",
  title = {{Polynomial Algorithms in Computer Algebra}},
  year = "1996",
  publisher = "Springer-Verlag",
  isbn = "3-211-82759-5",
  abstract =
    "This book surveys algorithms and results of Computer Algebra that are
```

```
    concerned with polynomials. It introduces algorithms from the bottom
    up, starting from very basic problems in computation over the
    integers, and finally leading to, e.g. advanced topics in
    factorization, solution of polynomial equations and constructive
    algebraic geometry. It is not based on a particular computer algebra
    program system.

    After two introductory chapters, the book contains six chapters with
    the following respective topics: computation by homomorphic images,
    gcd computation, factorization and decomposition of polynomials,
    linear systems and Hankel systems, Grbner bases. The last three
    chapters are concerned with applications of polynomial algorithms to
    higher level problems in computer algebra. In particular, a decision
    algorithm in the elementary theory of real closed fields, a
    description of Gospers algorithm for solving summation problems, and
    an algorithm for deciding whether an algebraic curve can be
    parametrized by rational functions (and if so for computing such a
    parametrization) is given. Along the way, the complexity of many of
    the algorithms is investigated. Each chapter ends with rich
    bibliographical notes.

    The book was originally developed from course material. It can easily
    be used as a textbook on the topic. Most subsections contain
    exercises. Solutions of some of the exercises are provided.",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@techreport{Wink99,
  author = "Winkler, Franz",
  title = {{Advances and Problems in Algebraic Computation}},
  type = "technical report",
  number = "99-49",
  year = "1999",
  institution = "RISC, Linz",
  abstract =
    "In the last years there has been dramatic progress in all areas
     of algebraic computation. Many working mathematiians have access
     to and actually use algebraic software systems for their research
     No end of this development is in sight. We report on some active
     topics in algebraic computation, namely the theory of integration
     and symbolic solution of systems of differential equations, the
     solution of systems of algebraic equations, the factorization of
     polynomials, and the design and analysis of algebraic curves and
     surfaces.",
  paper = "Wink99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Wink06,
  author = "Winkler, Franz",
  title = {{Computer Algebra and Geometry - Some Interactions}},
  booktitle = "Proc. of Coll. on Constructive Algebra and System
               Theory (CAST)",
  year = "2006",
  publisher = "unknown",
  isbn = "90-6984-477-x",
  pages = "127-138",
  abstract =
    "Algebraic curves and surfaces have been studied intensively in
    algebraic geometry for centuries. Thus, there exists a huge amount
    of theoretical knowledge about these geometric objects. Recently,
    algebraic curves and surfaces play an important and ever
    increasing role in computer aided geometric design, computer
    vision, computer aided manufacturing, coding theory, and
    cryptography, just to name a few application areas. Consequently,
    theoretical results need to be adapted to practical needs. We need
    efficient algorithms for generating, representing, manipulating,
    analyzing, rendering algebraic curves and surfaces. Exact computer
    algebraic methods can be employed effectively for dealing with
    these geometric problems.",
  paper = "Wink06.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Wity85,
  author = "Wityak, Sandra",
  title = {{The Scratchpad II Newsletter}},
  volume = "1",
  number = "1",
  year = "1985",
  month = "September",
  institution = "IBM Research",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Wity86,
  author = "Wityak, Sandra",
  title = {{The Scratchpad II Newsletter}},
  volume = "1",
  number = "2",
  year = "1986",
```

```
  month = "January",
  institution = "IBM Research",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Wity86a,
  author = "Wityak, Sandra",
  title = {{The Scratchpad II Newsletter}},
  volume = "1",
  number = "3",
  year = "1986",
  month = "May",
  institution = "IBM Research",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Wrig03,
  author = "Wright, Francis J.",
  title = {{Mathematics and Algorithms for Computer Algebra: Part 1}},
  link = "\url{https://people.eecs.berkeley.edu/~fateman/282/F%20Wright%20notes/week1.pdf}",
  year = "2003",
  comment = "full course. week2=Wrig03a, wee3=Wrig03b,...week8=Wrig3g.pdf",
  abstract =
    "This course will be mainly mathematics, some computer science
    and a little computing. Little or no essential use will be made of
    actual computer languages, although I may occasionally use Pascal, C,
    Lisp or REDUCE for concrete examples. The aim of the course is to
    provide an entry into the current research literature, but not to present
    the most recent research results.

    The first half of the course (taught by me) will deal with basic
    mathematics and algorithms for computer algebra, primarily at the level of
    arithmetic and elementary abstract algebra, including an introduction
    to GCDs and the solution of univariate polynomial equations. This
    leads into the second half of the course (taught by Dr Jim Skea) on
    the more advanced problems of polynomial factorization, indefinite
    integration, multivariate polynomial equations, etc.

    The first week provides an introduction to the computing aspects
    of computer algebra, and contains almost no mathematics. It is
    intended to show how the later theory can be implemented for practical
    computation. The second week provides a rapid but superficial survey
    of the abstract algebra that is most important for computer algebra.
    The next five weeks will build on this abstract basis to do some more
```

```
    concrete mathematics in more details, referring back to the basis
    setablished in the first two weeks as necessary.

    At the end of each set of notes will be exercises, one (or more) of
    which will be assessed.",
  paper = "Wrig03.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[WWW1]{WWW1}.
Software Preservation Group
  link = "\url{http://www.softwarepresentation.org/projects/LISP/common_lisp_family}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@misc{WikiG,
  author = "Wikipedia Authors",
  title = {{Gamma Function}},
  year = "2016",
  link = "\url{https://en.wikipedia.org/wiki/Gamma\_function}",
  algebra = "\newline\refto{package DFSFUN DoubleFloatSpecialFunctions}"
}
```

———-

1.40.24 X

— **axiom.bib** —

```
@misc{Xena19,
  author = "Buzzard, Kevin",
  title = {{Xena}},
  link = "\url{https://xenaproject.wordpress.com}",
  year = "2019"
}
```

———-

1.40.25 Y

— **ignore** —

```
\bibitem[Yap 00]{Yap00} Yap, Chee Keng
  title = {{Fundamental Problems of Algorithmic Algebra}},
Oxford University Press (2000) ISBN0-19-512516-9
  keywords = "axiomref",
```

———-

— **ignore** —

```
\bibitem[Yapp 07]{Yapp07} Yapp, Clifford; Hebisch, Waldek; Kaminski, Kai
  title = {{Literate Programming Tools Implemented in ANSI Common Lisp}},
  link = "\url{http://brlcad.org/~starseeker/cl-web-v0.8.lisp.pamphlet}",
  keywords = "axiomref"
```

———-

— **axiom.bib** —

```
@inproceedings{Yunx76,
  author = "Yun, David Y.Y",
  title = {{Algebraic Algorithms using p-adic Constructions}},
  booktitle = "Proc. 1976 Symp. on Symbolic and Algebraic Computation",
  series = "SYMSAC '76",
  publisher = "ACM",
  year = "1976",
  pages = "248-259",
  link = "\url{http://lib.org/by/\_djvu\_Papers/Computer\_algebra/Algebraic\%20numbers}",
  paper = "Yunx76.djvu",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Yunx83,
  author = "Yun, David Y.Y",
  title = {{Computer Algebra and Complex Analysis}},
  booktitle = "Computational Aspects of Complex Analysis",
  pages = "379-393",
  publisher = "D. Reidel Publishing Company",
  year = "1983",
  abstract =
    "Taking complex analysis to mean complex numerical analysis, I
    perceive my mission here to be that of disseminating the algebraic
    approach taken by computer algebraists to many mathematical problems,
    which arise from and are important to complex analysis. In turn,
    complex numerical analysis can be, and have been, providing essential
    theoretical and computational results for computer algebra. The cross
    fertilization should and must continue in order that computational
    mathematics progress with the joint aid of both tools, rather than
```

```
    branching into orthogonal pursuits with disparate approaches. First,
    we discuss the different issues and principal concerns of computer
    algebra. Then, the algebraic approach to a long standing problem in
    calculus or complex analysis, indefinite integration in closed form,
    will be motivated and derived through examples. Algorithmic solution
    to the basic, thought provoking, problem of rational function
    integration as well as theoretical foundation underlying the algorithm
    for elementary function integration will be discussed. Further issues
    and approaches will be illustrated through another central (implicitly
    essential) problem of computer algebra, that is simplification of
    symbolic and algebraic expressions. We conclude by showing a set of
    computer executed problems in integration to reveal some of the new
    capabilities added to the arsenal of a mathematician through the
    efforts of computer algebra.",
  keywords = "axiomref"
}
```

———-

1.40.26 Z

— **axiom.bib** —

```
@mastersthesis{Zeng92,
  author = "Zenger, Christoph",
  title = {{Gr\"obnerbasen f\"ur Differentialformen und ihre
           Implementierung in AXIOM'}},
  type = "Diplomarbeit",
  school = "Universit{\"a}t Karlsruhe",
  address = "Karlsruhe, Germany",
  year = "1992",
  keywords = "axiomref",
  beebe = "Zenger:1992:GFD"
}
```

———-

— **Zenger:1992:GFD** —

```
@MastersThesis{Zenger:1992:GFD,
  author =       "Ch. Zenger",
  title =        {{Gr\"obnerbasen f\"ur Differentialformen und ihre
                 Implementierung in AXIOM}},
  type =         "Diplomarbeit",
  school =       "Universit{\"a}t Karlsruhe",
  address =      "Karlsruhe, Germany",
  pages =        "??",
  year =         "1992",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
}
```

———-

— **axiom.bib** —

```
@article{Zeng97,
  author = "Zenger, Christoph",
  title = {{Indexed types}},
  journal = "Theor. Comput. Sci.",
  volume = "187",
  numbers = "1-2",
  pages = "147-165",
  year = "1997",
  abstract =
    "A new extension of the Hindley/Milner type system is proposed. The
    type system has algebraic types, that have not only type parameters
    but also value parameters (indices). This allows for example to
    parameterize matrices and vectors by their size and to check size
    compatibility statically. This is especially of interest in computer
    algebra.",
  paper = "Zeng97.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Zimm95,
  author = "Zimmermann, Paul",
  title = {{Wester's test suite in MuPAD 1.2.2}},
  year = "1995",
  abstract =
    "A few months ago, Michael Wester made a review of the mathematical
    capabilities of different computer algebra systems, namely Axiom,
    Derive, Macsyma, Maple, Mathematica and Reduce. This review, which is
    available by anonymous ftp from math.unm.edu, file pub/cas/Paper.ps,
    consists of 131 tests in different domains of mathematics (arithmetic,
    algebraic equations, differential equations, integration, operator
    computation, series expansions, limits).

    We describe in this paper the problems that can be solved with MuPAD
    1.2.2, and how to solve them. The problems marked as [New] are solved
    using new functionalities of the version 1.2.2 with respect to 1.2.1",
  paper = "Zimm95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Zimm96,
```

```
  author = "Zimmermann, Paul",
  title = {{Wester's test suite in MuPAD 1.3}},
  year = "1996",
  abstract =
    "In December 1994, Michael Wester made a review of the mathematical
    capabilities of different computer algebra systems, namely Axiom,
    Derive, Macsyma, Maple, Mathematica and Reduce. This review, which is
    available by anonymous ftp from math.unm.edu, file pub/cas/Paper.ps,
    consists of 131 tests in different domains of mathematics (arithmetic,
    algebraic equations, differential equations, integration, operator
    computation, series expansions, limits).

    We describe in this paper the problems that can be solved with MuPAD
    1.3, and how to solve them. The problems marked as [New] are solved
    using new functionalities of the version 1.3 with respect to 1.2.2",
  paper = "Zimm96.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Zipp92a,
  author = "Zippel, Richard",
  title = {{Algebraic Computation}},
  publisher = "Cornell University",
  comment = "Unpublished",
  year = "1992",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Zipp89,
  author = "Zippel, Richard",
  title = {{The Weyl Computer Algebra Substrate}},
  year = "1989",
  link =
 "\url{https://ecommons.cornell.edu/bitstream/handle/1813/6917/90-1077.pdf}",
  paper = "Zipp89.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@book{Zipp92,
```

```
  author = "Zippel, Richard",
  title = {{Symbolic/Numeric Techniques in Modeling and Simulation}},
  link = "\url{http://www.cs.duke.edu/donaldlab/Books/SymbolicNumericalComputation/323-346.pdf}",
  year = "1992",
  publisher = "Academic Press Limited",
  isbn = "0-12-220535-9",
  abstract =
    "Modeling and simulating collections of physical objects that are
    subject to a wide variety of physical forces and interactions is
    exceedingly difficult.  The construction of a single simulator capable
    of dealing with all possible physical processes is completely
    impractical and, it seems to us, wrong-headed.  Instead, we propose
    to build custom simulators designed for a particular collection of
    physical objects, where a particular set of physical phenomena are
    involved.  For such an approach to be practical, an environment must
    be provided that facilitates the quick construction of these
    simulators.  In this paper we describe the essential features of such
    an environment and describe in some detail how a general
    implementation of the weighted residual method, one of the more
    general classes of numerical integration techniques, can be used.",
  paper = "Zipp92.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Zipp93,
  author = "Zippel, Richard",
  title = {{The Weyl Computer Algebra Substrate}},
  booktitle = "Design and Implementation of Symbolic Computation Systems '93",
  series = "DISCO 93",
  pages = "303=318",
  year = "1993",
  abstract =
    "Weyl is a new type of computer algebra substrate that extends an
    existing, object oriented programming language with symbolic computing
    mechanisms.  Rather than layering a new language on top of an existing
    one, Weyl behaves like a powerful subroutine library, but takes heavy
    advantage of the ability to overload primitive arithmetic operations
    in the base language.  In addition to the usual objects manipulated in
    computer algebra systems (polynomial, rational functions, matrices,
    etc.), domains (e.g., Z, Q[x, y, z]) are also first class objects in
    Weyl.",
  paper = "Zipp93.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@book{Zwil92,
  author = "Zwillinger, Daniel",
  title = {{Handbook of Integration}},
  publisher = "Jones and Bartlett",
  year = "1992",
  isbn = "0-86720-293-9",
  keywords = "axiomref"
}
```

———-

1.41 Axiom Citations of External Sources

1.41.1 A

— **axiom.bib** —

```
@article{Abbo95,
  author = "Abbott, John and van Leeuwen, Andre and Strotmann, Andreas",
  title = {{Objectives of OpenMath}},
  journal = "J. Symbolic Computation",
  volume = "11",
  year = "1995",
  abstract =
    "OpenMath aims at providing a universal means of communicating
    mathematical information between applications. In this paper we set
    out the objectives and design goals of OpenMath , and sketch the
    framework of a model that meets these requirements. Based upon this
    model, we propose a structured approach for further development and
    implementation of OpenMath. Throughout, emphasis is on
    extensibility and flexibility, so that OpenMath is not confined to any
    particular area of mathematics, nor to any particular
    implementation. We give some example scenarios to motivate and clarify
    the objectives, and include a brief discussion of the parallels
    between this model and the theory of human language perception.",
  paper = "Abbo95.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Abla98,
  author = "Ablamowicz, Rafal",
  title = {{Spinor Representations of Clifford Algebras: A Symbolic Approach}},
  journal = "Computer Physics Communications",
  volume = "115",
  number = "2-3",
  month = "December",
  year = "1998",
```

```
  pages = "510-535"
}
```

———-

— **axiom.bib** —

```
@article{Abra06,
  author = "Abramov, Sergey A.",
  title = {{In Memory of Manuel Bronstein}},
  journal = "Programming and Computer Software",
  volume = "32",
  number = "1",
  pages = "56-58",
  link = "\url{http://www.ccas.ru/sabramov/ps/PCS56.pdf}",
  publisher = "Pleiades Publishing Inc",
  year = "2006",
  paper = "Abra06.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Abramowitz 64]{AS64} Abramowitz, Milton; Stegun, Irene A.
  title = {{Handbook of Mathematical Functions}},
(1964) Dover Publications, NY ISBN 0-486-61272-4
```

———-

— **ignore** —

```
\bibitem[Abramowitz 68]{AS68} Abramowitz M; Stegun I A
  title = {{Handbook of Mathematical Functions}},
Dover Publications. (1968)
```

———-

— **axiom.bib** —

```
@book{ADAx83,
  author = "U.S. Government",
  title = {{The Programming Language Ada Reference Manual}},
  publisher = "U.S. Government",
  year = "1983",
  comment = "STD-1815A-1983"
}
```

———-

— **axiom.bib** —

```
@misc{Adam17,
  author = "Adamchik, Victor",
  title = {{Modern Computer Algebra}},
  comment = "Carnegie Mellon SCS 15-355",
  year = "2017",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Alfo92,
  author = "Alford, W.R. and Granville, A. and Pomerance, C.",
  title = {{There are Infinitely Many Carmichael Numbers}},
  year = "1992",
  comment = "Preprint"
}
```

———-

— **axiom.bib** —

```
@book{Altm05,
  author = "Altmann, Simon L.",
  title = {{Rotations, Quaternions, and Double Groups}},
  publisher = "Dover Publications, Inc.",
  year = "2005",
  isbn = "0-486-44518-6"
}
```

———-

— **ignore** —

```
\bibitem[Ames 77]{Ames77} Ames W F
  title = {{Nonlinear Partial Differential Equations in Engineering}},
Academic Press (2nd Edition). (1977)
```

———-

— **ignore** —

```
\bibitem[Amos 86]{Amos86} Amos D E
  title = {{Algorithm 644: A Portable Package for Bessel Functions of a
           Complex Argument and Nonnegative Order}},
ACM Trans. Math. Softw. 12 265--273. (1986)
```

———-

— **ignore** —

```
\bibitem[Anderson 00]{And00} Anderson, Edward
  title = {{Discontinuous Plane Rotations and the Symmetric Eigenvalue
            Problem}},
LAPACK Working Note 150, University of Tennessee, UT-CS-00-454,
December 4, 2000.
```

———-

— **ignore** —

```
\bibitem[Anthony 82]{ACH82} Anthony G T; Cox M G; Hayes J G
  title = {{DASL - Data Approximation Subroutine Library}},
National Physical Laboratory. (1982)
```

———-

— **axiom.bib** —

```
@misc{Arma00,
  author = "Armando, Alessandro and Zini, Daniele",
  title = {{Towards Interoperable Mechanized Reasoning Systems:
            The Logic Broker Architecture}},
  year = "2000",
  abstract =
    "There is a growing interest in the integration of mechanized
    reasoning systems such as automated theorem provers, computer algebra
    systems, and model checkers.  State-of-the-art reasoning systems are
    the result of many man-years of careful development and engineering,
    and usually they provide a high degree of sophistication in their
    respective domain.  Yet they often perform poorly when applied outside
    the domain they have been designed for.  The problem of integrating
    mechanized reasoning systems is therefore being perceived as an
    important issue in automated reasoning.  In this paper we present
    the Logic Broker Architecture, a framework which provides the needed
    infrastructure for making mechanized reasoning systems interoperate.
    The architecture provides location transparency, a way to forward
    requests for logical services to appropriate reasoning systems via a
    simple registration/subscription mechanism, and a translation
    mechanism which ensures the transparent and provably sound exchange of
    logical services.",
  paper = "Arma00.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@misc{Arna91,
  author = "Arnault, F.",
  title = {{Le Test de Primalite de Rabin-Miller: Un Nombre compose
            qui le "passe"}},
  comment = "Report 61",
  university = "Universite de Poitiers Departement de Mathematiques",
  year = "1991"
}
```

———-

— **axiom.bib** —

```
@article{Aste02,
  author = "Astesiano, Egidio and Bidoit, Michel and Kirchner, Helene and
            Krieg-Bruckner, Bernd and Mosses, Peter D. and Sannella, Donald
            and Tarlecki, Andrzej",
  title = {{CASL: the Common Algebraic Specification Language}},
  journal = "Theoretical Computer Science",
  volume = "286",
  number = "2",
  pages = "153-196",
  year = "2002",
  abstract =
    "The Common Algebraic Specification Language (CASL) is an expressive
    language for the formal specification of functional requirements and
    modular design of software. It has been designed by COFI, the
    international Common Framework Initiative for algebraic specification
    and development. It is based on a critical selection of features that
    have already been explored in various contexts, including subsorts,
    partial functions, first-order logic, and structured and architectural
    specifications. CASL should facilitate interoperability of many
    existing algebraic prototyping and verification tools.

    This paper gives an overview of the CASL design. The major issues that
    had to be resolved in the design process are indicated, and all the
    main concepts and constructs of CASL are briefly explained and
    illustrated  the reader is referred to the CASL Language Summary for
    further details. Some familiarity with the fundamental concepts of
    algebraic specification would be advantageous.",
  paper = "Aste02.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Aubr99,
  author = "Aubry, Phillippe and Lazard, Daniel and {Moreno Maza}, Marc",
  title = {{On the Theories of Triangular Sets}},
  year = "1999",
```

```
  pages = "105-124",
  journal = "Journal of Symbolic Computation",
  volume =  "28",
  link = "\url{http://www.csd.uwo.ca/~moreno/Publications/Aubry-Lazard-MorenoMaza-1999-JSC.pdf}",
  papers = "Aubr99.pdf",
  algebra =
    "\newline\refto{category TSETCAT TriangularSetCategory}
     \newline\refto{category RSETCAT RegularTriangularSetCategory}
     \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
     \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
     \newline\refto{package LEXTRIPK LexTriangularPackage}
     \newline\refto{package RSDCMPK RegularSetDecompositionPackage}",
  abstract =
    "Different notions of triangular sets are presented. The relationship
    between these notions are studied. The main result is that four
    different existing notions of {\sl good} triangular sets are
    equivalent."
}
```

———-

— **axiom.bib** —

```
@misc{Aubr96,
  author = "Aubry, Phillippe and Maza, Marc Moreno",
  title = {{Triangular Sets for Solving Polynomial Systems:
           a Comparison of Four Methods}},
  year = "1996",
  link = "\url{http://www.lip6.fr/lip6/reports/1997/lip6.1997.009.ps.gz}",
  algebra =
    "\newline\refto{category TSETCAT TriangularSetCategory}
     \newline\refto{category RSETCAT RegularTriangularSetCategory}
     \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
     \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
     \newline\refto{package LEXTRIPK LexTriangularPackage}
     \newline\refto{package RSDCMPK RegularSetDecompositionPackage}",
  abstract =
    "Four methods for solving polynomial systems by means of triangular
    sets are presented and implemented in a unified way. These methods are
    those of Wu, Lazard, Kalkbrener, and Wang. They are compared on
    various examples with emphasis on efficiency, conciseness and
    legibility of the outputs.",
  paper = "Aubr96.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Aubr99a,
  author = "Aubry, Phillippe and Maza, Marc Moreno",
```

```
  title = {{Triangular Sets for Solving Polynomial Systems:
           a Comparison of Four Methods}},
  journal = "J. Symb. Comput.",
  volume = "28",
  number = "1-2",
  pages = "125-154",
  year = "1999",
  link = "\url{http://www.csd.uwo.ca/~moreno/Publications/Aubry-MorenoMaza-1999-JSC.pdf}",
  algebra =
    "\newline\refto{category TSETCAT TriangularSetCategory}
     \newline\refto{category RSETCAT RegularTriangularSetCategory}
     \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
     \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
     \newline\refto{package RSDCMPK RegularSetDecompositionPackage}",
  abstract =
    "Four methods for solving polynomial systems by means of triangular
    sets are presented and implemented in a unified way. These methods are
    those of Wu, Lazard, Kalkbrener, and Wang. They are compared on
    various examples with emphasis on efficiency, conciseness and
    legibility of the outputs.",
  paper = "Aubr99a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Aubr99b,
  author = "Aubry, Philippe",
  title = {{Ensembles triangulaires de polynomes et resolution de systemes
           algebriques. Implantation en Axiom}},
  school = "l'Universite de Paris VI",
  year = "1999",
  month = "January",
  comment = "French",
  paper = "Aubr99b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Ausi79,
  author = "Ausiello, Giovanni Francesco Mascari",
  title = {{On the Design of Algebraic Data Structures with the
           Approach of Abstract Data Types}},
  journal = "LNCS",
  volume = "72",
  year = "1979",
  pages = "514-530",
  abstract =
```

```
    "The problem of giving a formal definition of the representation of
    algebraic data structures is considered and developped in the frame
    work of the abstract data types approach. Such concepts as canonical
    form and simplification are formalized and related to properties of
    the abstract specification and of the associated term rewriting
    system.",
  paper = "Ausi79.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Avig12,
  author = "Avigad, Jeremy",
  title = {{Interactive Theorem Proving, Automated Reasoning, and
           Mathematical Computation}},
  year = "2012",
  comment = "slides",
  paper = "Avig12.pdf",
  keywords ="CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@misc{Avig14a,
  author = "Avigad, Jeremy",
  title = {{Formal Verification, Interactive Theorem Proving, and
           Automated Reasoning}},
  year = "2014",
  comment = "slides",
  paper = "Avig14a.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Avig16a,
  author = "Avigad, Jeremy",
  title = {{Interactive Theorem Proving, Automated Reasoning, and Dynamical
           Systems}},
  year = "2016",
  comment = "slides",
  paper = "Avig16a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Avig17a,
  author = "Avigad, Jeremy",
  title = {{Formal Methods in Mathematics and the Lean Theorem Prover}},
  year = "2017",
  comment = "slides",
  paper = "Avig17a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Avig17d,
  author = "Avigad, Jeremy and de Moura, Leonardo and Ebner, Gabriel
            and Ullrich, Sebastian",
  title = {{An Introduction to Lean}},
  year = "2017",
  link = "\url{https://leanprover.github.io/introduction_to_lean/introduction_to_lean.pdf}",
  paper = "Avig17d.pdf",
  keywords = "printed"
}
```

———-

1.41.2 B

— **axiom.bib** —

```
@article{Bail96,
  author = "Bailey, Anthony",
  title = {{Coercion Synthesis in Computer Implementations of
            Type-Theoretic Frameworks}},
  journal = "LNCS",
  year = "1996",
  pages = "9-27",
  abstract =
    "A coercion is a function that acts on a representation of some object
    in order to alter its type.  The idea is that although applying a
    coercion to an object changes its type, the result still represents
    the ‘‘same’’ abject in some sense; perhaps it is some essential
    underlying part of the original object, or a different representation
    of that object.  This paper examines some of the issues involved in
    the computer implementation of systems that allow a user to define
    coercions that may then be left implicit in the syntax of expressions
    and synthesised automatically.  From a type-theoretic perspective,
    coercions are often left implicit in mathematical texts, so they can
    be used to improve the readability of a formalisation, and to
    implement other tricks of syntax if so desired.",
  paper = "Bail96.pdf",
```

```
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Bailey 66]{Bai66} Bailey P B
  title = {{Sturm-Liouville Eigenvalues via a Phase Function}},
SIAM J. Appl. Math . 14 242--249. (1966)
```

———-

— **axiom.bib** —

```
@phdthesis{Bail98,
  author = "Bailey, Anthony",
  title = {{The Machine-Checked Literate Formalisation of Algebra
            in Type Theory}},
  school = "University of Manchester",
  year = "1998",
  link =
  "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.464.1249&rep=rep1&type=pdf}",
  abstract =
    "I present a large-scale formalisation within a type theory of a
    proof of a result from abstract algebra.  The formalisation body
    consists of files that are machine-checked to ensure their
    correctness, and also processed to produce a report on the proof that
    is human-readable.  The resulting presentation is intended to approach
    being a standard informal account of some mathematics.  In addition
    to presenting this proof, the thesis also identifies and examines
    problems in reconciling the formal nature of the development with the
    wish for it to be easy to read.  It presents some tools and methodologies
    for solving these problems, and discusses the advantages and
    disadvantages of these solutions.  In particular, it addresses the
    implementation and use of implicit coercions within the type theory,
    the styles of proof that can be used, and the borrowing of concepts
    from the literate programming paradigm.  To be more specific, the
    algebra in question is a constructive version of the fundamental
    theorem of Galois Theory. The formalisation is developed within a
    variant of the Unified Theory of Types that is implemented by a
    modified version of the LEGO proof-checker.",
  paper = "Bail98.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Baker 96]{BGM96} Baker, George A.; Graves-Morris, Peter
```

```
  title = {{Pade Approximants}},
Cambridge University Press, March 1996 ISBN 9870521450072
```

———-

— **ignore** —

```
\bibitem[Baker 10]{Ba10} Baker, Martin
  title = {{3D World Simulation}},
  link = "\url{http://www.euclideanspace.com}",
```

———-

— **axiom.bib** —

```
@misc{Bake14,
 author = "Baker, Martin",
 title = {{Axiom Architecture}},
 year = "2014",
 link = "\url{http://www.euclideanspace.com/prog/scratchpad/internals/ccode}",
 keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Banks 68]{BK68} Banks D O; Kurowski I
  title = {{Computation of Eigenvalues of Singular Sturm-Liouville Systems}},
Math. Computing. 22 304--310. (1968)
```

———-

— **ignore** —

```
\bibitem[Bard 74]{Bard74} Bard Y
  title = {{Nonlinear Parameter Estimation}},
Academic Press. 1974
```

———-

— **axiom.bib** —

```
@article{Barr96,
  author = "Barras, Bruno",
  title = {{Verification of the Interface of a Small Proof System in Coq}},
  journal = "LNCS",
  volume = "1512",
  pages = "28-45",
  year = "1996",
```

```
  abstract =
    "This article describes the formalization of the interface of a
    proof-checker.  The latter is based on a kernel consisting of
    type-checking functions for the Calculus of Constructions, but it
    seems the ideas can generalize to other type systems, as fax as they
    axe based on the proofs- as-terms principle.  We suppose that the
    metatheory of the corresponding type system is proved (up to type
    decidability).  We specify and certify the toplevel loop, the system
    invariant, and the error messages.",
  paper = "Barr96.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Barrodale 73]{BR73} Barrodale I; Roberts F D K
  title = {{An Improved Algorithm for Discrete $ll_1$ Linear Approximation}},
SIAM J. Numer. Anal. 10 839--848. (1973)
```

———-

— **ignore** —

```
\bibitem[Barrodale 74]{BR74} Barrodale I; Roberts F D K
  title = {{Solution of an Overdetermined System of Equations
            in the $ll_1-norm$.}},
Comm.  ACM. 17, 6 319--320. (1974)
```

———-

— **axiom.bib** —

```
@article{Bart95,
  author = "Barthe, Gilles",
  title = {{Implicit Coercions in Type Systems}},
  journal = "LNCS",
  volume = "1158",
  year = "1995",
  pages = "1-15",
  abstract =
    "We propose a notion of pure type system with implicit coercions. In
    our framework, judgements are extended with a context of coercions
    and the application rule is modified so as to allow coercions to be
    left implicit. The setting supports multiple inheritance and can be
    applied to all type theories with $\Pi$-types. One originality of our
    work is to propose a computational interpretation for implicit
    coercions. In this paper, we demonstrate how this interpretation
    allows a strict control on the logical properties of pure type systems
    with implicit coecions.",
```

```
  paper = "Bart95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Bart72,
  author = "Barton, D.R. and Fitch, John P.",
  title = {{A Review of Algebraic Manipulative Programs and their
            Application}},
  journal = "The Computer Journal",
  volume = "15",
  number = "4",
  pages = "362-381",
  year = "1972",
  link = "\url{http://comjnl.oxfordjournals.org/content/15/4/362.full.pdf+html}",
  abstract =
    "This paper describes the applications area of computer programs that
    carry out formal algebraic manipulation. The first part of the paper
    is tutorial and severed typical problems are introduced which can be
    solved using algebraic manipulative systems. Sample programs for the
    solution of these problems using several algebra systems are then
    presented. Next, two more difficult examples are used to introduce the
    reader to the true capabilities of an algebra program and these are
    proposed as a means of comparison between rival algebra systems. A
    brief review of the technical problems of algebraic manipulation is
    given in the final section.",
  paper = "Bart72.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Bart94,
  author = "Barton, John J. and Nackman, Lee R.",
  title = {{Scientific and Engineering C++}},
  publisher = "Pearson",
  year = "1994",
  isbn = "97800201533934"
}
```

———-

— **axiom.bib** —

```
@misc{Batu03,
  author = "Batut, C. and Belabas, K. and Bernardi, D. and Cohen, H. and
```

```
           Olivier, M.",
  title = {{User's Guide to PARI/GP}},
  link = "\url{http://math.mit.edu/~brubaker/PARI/PARIusers.pdf}",
  paper = "Batu03.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Baue98,
  author = "Bauer, Andrej and Clarke, Edmund and Zhao, Xudong",
  title = {{Analytica -- An Experiment in Combining Theorem Proving and
           Symbolic Computation}},
  journal = "Journal of Automated Reasoning",
  volume = "21",
  number = "3",
  pages = "295-325",
  year = "1998",
  abstract =
    "Analytica is an automatic theorem prover for theorems in elementary
    analysis. The prover is written in the Mathematica language and runs
    in the Mathematica environment. The goal of the project is to use a
    powerful symbolic computation system to prove theorems that are beyond
    the scope of previous automatic theorem provers. The theorem prover is
    also able to deduce the correctness of certain simplification steps
    that would otherwise not be performed. We describe the structure of
    Analytica and explain the main techniques that it uses to construct
    proofs. Analytica has been able to prove several nontrivial
    theorems. In this paper, we show how it can prove a series of lemmas
    that lead to the Bernstein approximation theorem.",
  paper = "Baue98.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **ignore** —

```
\bibitem[Beauzamy 92]{Bea92} Beauzamy, Bernard
  title = {{Products of polynomials and a priori estimates for
            coefficients in polynomial decompositions: a sharp result}},
J. Symbolic Computation (1992) 13, 463-472
  paper = "Bea92.pdf"
```

———-

— **ignore** —

```
\bibitem[Beauzamy 93]{Bea93} Beauzamy, Bernard; Trevisan, Vilmar;
```

```
Wang, Paul S.
  title = {{Polynomial Factorization: Sharp Bounds, Efficient Algorithms}},
J. Symbolic Computation (1993) 15, 393-413
  paper = "Bea93.pdf"
```

———-

— **axiom.bib** —

```
@article{Bees90,
  author = "Beeson, Michael J.",
  title = {{Review of Implementing Mathematics with the Nuprl Proof
            Development System}},
  journal = "J. of Symbolic Logic",
  volume = "55",
  number = "3",
  pages = "1299-1302",
  year = "1990",
  link = "\url{http://www.jstor.org/stable/2274489}",
  paper = "Bees90.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Beez15,
  author = "Judson, Tom and Beezer, Rob",
  title = {{Abstract Algebra: Theory and Applications}},
  year = "2015",
  publisher = "Tom Judson",
  link = "\url{http://abstract.ups.edu/download/aata-20150812-sage-6.8.pdf}",
  paper = "Beez15.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Berg95,
  author = "Berger, U. and Schwichtenberg, H.",
  title = {{The Greatest Common Divisor: A Case Study for Program
            Extraction from Classical Proofs}},
  journal = "LNCS",
  volume = "1158",
  year = "1995",
  pages = "36-46",
  paper = "Berg95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Bern85,
  author = "Berndt, B.C.",
  title = {{Ramanujan's Notebooks, Part I}},
  publisher = "Springer-Verlag",
  year = "1985",
  pages = "25-43"
}
```

———-

— **axiom.bib** —

```
@book{Bern91,
  author = "Bernays, Paul",
  title = {{Axiomatic Set Theory}},
  publisher = "Dover",
  year = "1991"
}
```

———-

— **axiom.bib** —

```
@article{Bert98,
  author = "Bertoli, P. and Calmet, J. and Guinchiglia, F. and Homann, K.",
  title = {{Specification and Integration of Theorem Provers and Computer
           Algebra Systems}},
  journal = "Lecture Notes in Computer Science",
  volume = "1476",
  year = "1998",
  pages = "94-106",
  abstract =
    "Computer algebra systems (CASs) and automated theorem provers (ATPs)
    exhibit complementary abilities. CASs focus on efficiently solving
    domain-specific problems. ATPs are designed to allow for the
    formalization and solution of wide classes of problems within some
    logical framework. Integrating CASs and ATPs allows for the solution
    of problems of a higher complexity than those confronted by each class
    alone. However, most experiments conducted so far followed an ad-hoc
    approach, resulting in tailored solutions to specific problems. A
    structured and principled approach is necessary to allow for the sound
    integration of systems in a modular way. The Open Mechanized Reasoning
    Systems (OMRS) framework was introduced for the specification and
    implementation of mechanized reasoning systems, e.g. ATPs. The
    approach was recasted to the domain of computer algebra systems. In
    this paper, we introduce a generalization of OMRS, named OMSCS (Open
    Mechanized Symbolic Computation Systems). We show how OMSCS can be
    used to soundly express CASs, ATPs, and their integration, by
```

```
    formalizing a combination between the Isabelle prover and the Maple
    algebra system. We show how the integrated system solves a problem
    which could not be tackled by each single system alone.",
  paper = "Bert98.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@article{Bert95,
  author = "Bertrand, Laurent",
  title = {{Computing a hyperelliptic integral using arithmetic in the
           jacobian of the curve}},
  journal = "Applicable Algebra in Engineering, Communication and Computing",
  volume = "6",
  pages = "275-298",
  year = "1995",
  abstract = "
    In this paper, we describe an efficient algorithm for computing an
    elementary antiderivative of an algebraic function defined on a
    hyperelliptic curve. Our algorithm combines B.M. Trager's integration
    algorithm and a technique for computing in the Jacobian of a
    hyperelliptic curve introduced by D.G. Cantor. Our method has been
    implemented and successfully compared to Trager's general algorithm."
}
```

———-

— **ignore** —

```
\bibitem[Berzins 87]{BBG87} Berzins M; Brankin R W; Gladwell I.
  title = {{Design of the Stiff Integrators in the NAG Library}},
Technical Report. TR14/87 NAG. (1987)
```

———-

— **ignore** —

```
\bibitem[Berzins 90]{Ber90} Berzins M
  title = {{Developments in the NAG Library Software for Parabolic Equations}},
Scientific Software Systems. (ed J C Mason and M G Cox)
Chapman and Hall. 59--72.  (1990)
```

———-

— **ignore** —

```
\bibitem[Birkhoff 62]{BR62} Birkhoff, G; Rota, G C
  title = {{Ordinary Differential Equations}},
```

```
Ginn \& Co., Boston and New York. (1962)
```

———-

— **axiom.bib** —

```
@inproceedings{Bitt94,
  author = "Bittencourt, G. and Calmet, Jacques and Homann, K. and
            Lulay, A.",
  title = {{MANTRA: A Multi-Level Hybrid Knowledge Representation System}},
  booktitle = "Proc. XI Brazilian Symp. on Artificial Intelligence",
  pages = "493-506",
  year = "1994",
  abstract =
    "The intelligent behavior of a system is based upon its represented
    knowledge and inference capabilities. In this paper we report on a
    knowledge representation and reasoning system, developed at the
    University of Karlsruhe, called Mantra. The system provides four
    different knowledge representation methods -- first-order logic,
    terminological language, semantic networks, and production rules --
    distributed into a three levels architecture. The first three methods
    form the lowest level of the architecture, the epistemological
    level. The supported hybrid inferences as well as the management of
    knowledge bases form the second level, called logical level. Finally,
    the third level, the heuristic level, provides representation of
    procedural knowledge of a domain, and the introduction of ad hoc
    rules. This knowledge is represented in terms of production rules
    which are processed by a Ops5-like rule interpreter. This paper mainly
    describes the introduction of this level into the hybrid system. The
    semantics of the knowledge representation methods of the
    epistemological level is dened according to a four-valued logic
    approach. This denition insures that all inference algorithms are
    sound, complete and decidable. The system has been implemented in
    Common Lisp using the object-oriented extension CLOS, and the
    graphical user interface was implemented in C with XToolkit.",
  paper = "Bitt94.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Blai71,
  author = "Blair, Fred W. and Griesmer, James H. and Harry, Joseph and
            Pivovonsky, Mark",
  title = {{Design and Development Document for Lisp on Several S/360
            Operating Systems}},
  year = "1971",
  publisher = "IBM Research",
  paper = "Blai71.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Boge77,
  author = "Bogen, Richard",
  title = {{MACSYMA Reference Manual, Version 9}},
  publisher = "MIT",
  year = "1977",
  link = "\url{http://bitsavers.informatik.uni-stuttgart.de/pdf/mit/macsyma/MACSYMA_RefMan_V9_Dec77.pdf}",
  paper = "Boge77.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Bott17,
  author = "Rademacher, Gunther",
  title = {{Railroad Diagram Generator}},
  link = "\url{http://www.bottlecaps.de/rr/ui}",
  year = "2017"
}
```

———-

— **axiom.bib** —

```
@article{Bove02,
  author = "Bove, Ana",
  title = {{General Recursion in Type Theory}},
  journal = "LNCS",
  volume = "2646",
  pages = "39-58",
  year = "2002",
  abstract =
    "In this work, a method to formalise general recursive algorithms in
    constructive type theory is presented throughout examples.  The method
    separates the computational and logical parts of the definitions. As
    a consequence, the resulting type-theoretic algorithms are clear,
    compact and easy to understand. They are as simple as their
    equivalents in a functional programming language, where there is no
    restriction on recursive calls. Given a general recursive algorithm,
    the method consists in defining an inductive special-purpose
    accessibility predicate that characterises the inputs on which the
    algorithm terminates. The type-theoretic version of the algorithm can
    then be defined by structural recursion on the proof that the input
    values satisfy this predicate. When formalising nested algorithms, the
    special-purpose accessibility predicate and the type-theoretic version
    of the algorithm must be defined simultaneously because they depend on
    each other. Since the method separates the computational part from
    the logical part of a definition, formalising partial functions
```

```
    becomes also possible.",
  paper = "Bove02.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Boyd9 3a]{Boyd93a} Boyd, David W.
  title = {{Bounds for the Height of a Factor of a Polynomial in Terms of
            Bombieri's Norms: I. The Largest Factor}},
J. Symbolic Computation (1993) 16, 115-130
  paper = "Boyd93a.pdf"
```

———-

— **ignore** —

```
\bibitem[Boyd 93b]{Boyd93b} Boyd, David W.
  title = {{Bounds for the Height of a Factor of a Polynomial in Terms of
            Bombieri's Norms: II. The Smallest Factor}},
J. Symbolic Computation (1993) 16, 131-145
  paper = "Boyd93b.pdf"
```

———-

— **axiom.bib** —

```
@inproceedings{Brad86,
  author = "Bradford, Russell J. and Hearn, Anthony C. and Padget, Julian and
            Schrufer, Eberhard",
  title = {{Enlarging the REDUCE domain of computation}},
  booktitle = "Proc SYMSAC 1986",
  series = "SYMSAC '86",
  publisher = "ACM",
  year = "1986",
  pages = "100-106",
  isbn = "0-89791-199-7",
  abstract =
    "We describe the methods available in the current REDUCE system for
    introducing new mathematical domains, and illustrate these by discussing
    several new domains that significantly increase the power of the overall
    system."
}
```

———-

— **axiom.bib** —

```
@misc{Brad92a,
```

```
  author = "Bradford, Russell J.",
  title = {{C78: Computer Algebra Course Lecture Notes}},
  institution = "Univ. of Bath",
  year = "1992"
}
```

———-

— **axiom.bib** —

```
@article{Brad13,
  author = "Brady, Edwin",
  title = {{Idris, a General Purpose Dependently Typed Programming
           Language: Design and Implementation}},
  journal = "J. Functional Programming",
  volume = "23",
  number = "5",
  year = "2013",
  pages = "552-593",
  abstract =
    "Many components of a dependently typed programming language are by
    now well understood, for example, the underlying type theory, type
    checking, unification and evaluation. How to combine these components
    into a realistic and usable high-level language is, however, folklore,
    discovered anew by successive language implementors. In this paper, I
    describe the implementation of Idris, a new dependently typed
    functional programming language. Idris is intended to be a
    general-purpose programming language and as such provides high-level
    concepts such as implicit syntax, type classes and do notation. I
    describe the high-level language and the underlying type theory, and
    present a tactic-based method for elaborating concrete high-level
    syntax with implicit arguments and type classes into a fully explicit
    type theory. Furthermore, I show how this method facilitates the
    implementation of new high-level language constructs.",
  paper = "Brad13.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Bram02a,
  author = "Braman, Karen and Byers, Ralph and Mathias, Roy",
  title = {{The Multi-Shift QR Algorithm Part I: Maintaining Well Focused
           Shifts, and Level 3 Performance}},
  journal = "SIAM Journal of Matrix Analysis",
  year = "2002",
  volume = "23",
  pages = "929-947",
  abstract =
    "This paper presents a small-bulge multishift variation of the
```

```
    multishift QR algorithm that avoids the phenomenon of shift blurring,
    which retards convergence and limits the number of simultaneous
    shifts. It replaces the large diagonal bulge in the multishift QR
    sweep with a chain of many small bulges. The small-bulge multishift QR
    sweep admits nearly any number of simultaneous shifts -- even hundreds
    -- without adverse effects on the convergence rate. With enough
    simultaneous shifts, the small-bulge multishift QR algorithm takes
    advantage of the level 3 BLAS, which is a special advantage for
    computers with advanced architectures."

}
```

———-

— **axiom.bib** —

```
@article{Bram02b,
  author = "Braman, Karen and Byers, Ralph and Mathias, Roy",
  title = {{The Multi-Shift QR Algorithm Part II: Aggressive Early Deflation}},
  journal = "SIAM Journal of Matrix Analysis",
  year = "2002",
  volume = "23",
  pages = "948-973",
  abstract =
    "Aggressive early deflation is a QR algorithm strategy that takes
    advantage of matrix perturbations outside of the subdiagonal entries
    of the Hessenberg QR iterate. It identifies and deflates converged
    eigenvalues long before the classic small-subdiagonal strategy
    would. The new deflation strategy enhances the performance of
    conventional large-bulge multishift QR algorithms, but it is
    particularly effective in combination with the small-bulge multishift
    QR algorithm. The small-bulge multishift QR sweep with aggressive
    early deflation maintains a high rate of execution of floating point
    operations while significantly reducing the number of operations
    required."

}
```

———-

— **axiom.bib** —

```
@misc{Bren10,
  author = "Brent, Richard P. and Zimmermann, Paul",
  title = {{Modern Computer Arithmetic}},
  year = "2010",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Brent 75]{Bre75} Brent, R. P.
  title = {{Multiple-Precision Zero-Finding Methods and the Complexity of
          Elementary Function Evaluation, Analytic Computational Complexity}},
J. F. Traub, Ed., Academic Press, New York 1975, 151-176
```

————-

— **ignore** —

```
\bibitem[Brent 78]{BK78} Brent, R. P.; Kung, H. T.
  title = {{Fast Algorithms for Manipulating Formal Power Series}},
Journal of the Association for Computing Machinery,
Vol. 25, No. 4, October 1978, 581-595
```

————-

— **ignore** —

```
\bibitem[Brigham 73]{Bri73} Brigham E O
  title = {{The Fast Fourier Transform}},
Prentice-Hall. (1973)
```

————-

— **ignore** —

```
\bibitem[Brillhart 69]{Bri69} Brillhart, John
  title = {{On the Euler and Bernoulli polynomials}},
J. Reine Angew. Math., v. 234, (1969), pp. 45-64
```

————-

— **ignore** —

```
\bibitem[Brillhart 90]{Bri90} Brillhart, John
  title = {{Note on Irreducibility Testing}},
Mathematics of Computation, vol. 35, num. 35, Oct. 1980, 1379-1381
```

————-

— **axiom.bib** —

```
@misc{Bron95,
  author = "Bronstein, Manuel",
  title = {{On radical solutions of linear ordinary differential equations}},
  year = "1995",
  link = "\url{ftp://ftp.inf.ethz.ch/org/cathode/workshops/jan95/abstracts/bronstein.ps}",
  algebra =
```

```
  "\newline\refto{category OREPCAT UnivariateSkewPolynomialCategory}
   \newline\refto{category LODOCAT LinearOrdinaryDifferentialOperatorCategory}
   \newline\refto{domain AUTOMOR Automorphism}
   \newline\refto{domain ORESUP SparseUnivariateSkewPolynomial}
   \newline\refto{domain OREUP UnivariateSkewPolynomial}
   \newline\refto{domain LODO LinearOrdinaryDifferentialOperator}
   \newline\refto{domain LODO1 LinearOrdinaryDifferentialOperator1}
   \newline\refto{domain LODO2 LinearOrdinaryDifferentialOperator2}
   \newline\refto{package APPLYORE ApplyUnivariateSkewPolynomial}
   \newline\refto{package OREPCTO UnivariateSkewPolynomialCategoryOps}
   \newline\refto{package LODOF LinearOrdinaryDifferentialOperatorFactorizer}
   \newline\refto{package LODOOPS LinearOrdinaryDifferentialOperatorsOps}",
  paper = "Bron95.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Bronstein 98a]{Bro98a} Bronstein, M.; Grabmeier, J.; Weispfenning, V. (eds)
  title = {{Symbolic Rewriting Techniques}},
Progress in Computer Science and Applied Logic 15, Birkhauser-Verlag, Basel
ISBN 3-7643-5901-3 (1998)
```

———-

— **axiom.bib** —

```
@article{Bron90,
  author = "Bronstein, Manuel",
  title = {{The Transcendental Risch Differential Equation}},
  journal = "J. Symbolic Computation",
  volume = "9",
  pages = "49-60",
  year = "1990",
  comment = "IBM Research Report RC13460 IBM Corp. Yorktown Heights, NY",
  algebra =
    "\newline\refto{package RDETR TranscendentalRischDE}
     \newline\refto{package RDETRS TranscendentalRischDESystem}",
  abstract =
    "We present a new rational algorithm for solving Risch differential
    equations in towers of transcendental elementary extensions. In
    contrast to a recent algorithm by Davenport we do not require a
    progressive reduction of the denominators involved, but use weak
    normality to obtain a formula for the denominator of a possible
    solution. Implementation timings show this approach to be faster than
    a Hermite-like reduction.",
  paper = "Bron90.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Bron98,
  author = "Bronstein, Manuel",
  title = {{The lazy hermite reduction}},
  type = "Rapport de Recherche",
  number = "RR-3562",
  year = "1998",
  institution = "French Institute for Research in Computer Science",
  abstract = "
    The Hermite reduction is a symbolic integration technique that reduces
    algebraic functions to integrands having only simple affine
    poles. While it is very effective in the case of simple radical
    extensions, its use in more general algebraic extensions requires the
    precomputation of an integral basis, which makes the reduction
    impractical for either multiple algebraic extensions or complicated
    ground fields. In this paper, we show that the Hermite reduction can
    be performed without {\sl a priori} computation of either a primitive
    element or integral basis, computing the smallest order necessary for
    a particular integrand along the way.",
  paper = "Bron98.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Bro98b,
  author = "Bronstein, Manuel",
  title = {{Symbolic Integration Tutorial}},
  series = "ISSAC'98",
  year = "1998",
  address = "INRIA Sophia Antipolis",
  link =
"\url{http://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/issac98.pdf}",

}
```

———-

— **axiom.bib** —

```
@article{Brui94,
  author = "de Bruijn, N.G.",
  title = {{A Survey of the project Automath}},
  journal = "Studies in Logic and the Foundations of Mathematics",
  volume = "133",
  year = "1994",
  pages = "141-161",
  abstract =
    "Thus far, much about Automath has been written in separate
    reports. Most of this work has been made available upon request, but
```

```
    only a small part was published in journals, conference proceedings,
    etc. Unfortunately, a general survey in the form of a book is still
    lacking. A short survey was given in [de Bruijn 73c], but the present
    one will be much more extensive. Naturally, this survey will report
    about work that has been done, is going on, or is planned for the
    future. But it will also be used to explain how various parts of the
    project are related. Moreover we shall try to clarify a few points
    which many outsiders consider as uncommon or even wierd. In particular
    we spend quite some attention to our concept of types and the matter
    of ``propositions as types'' (Section 14). Finally the survey will be
    used to ventilate opinions and views in mathematics which are not
    easily set down in more technical reports.",
  paper = "Brui94.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@incollection{Brui94a,
  author = "de Bruijn, N.G.",
  title = {{The Mathematical Vernacular, a Language for Mathematics
            with Typed Sets}},
  booktitle = "Selected Papers on Automath",
  pages = "865-935",
  year = "1994",
  publisher = "North-Holland",
  link = "\url{https://pure.tue.nl/ws/files/2073504/610209.pdf}",
  paper = "Brui94a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Buch87,
  author = "Buchberger, Bruno",
  title = {{Applications of Gr\"obner Bases in Non-Linear
           Computational Geometry}},
  journal = "Lecture Notes in Computer Science",
  volume = "296",
  year = "1987",
  paper = "52-80",
  abstract =
    "Groebner bases are certain finite sets of multivariate
    polynomials. Many problems in polynomial ideal theory (algebra
    geometry, non-linear computational geometry) can be solved by easy
    algorithms after transforming the polynomial sets involved in the
    specification of the problems into Groebner basis form. In this paper
    we give gome examples of applying the Groebner bases method to
    problems in non-linear computational geometry (inverse kinematics in
```

```
    robot programming, collision detection for superellipsoids,
    implicitization of parametric representations of curves and surfaces,
    inversion problem for parametric representations, automated
    geometrical theorem proving, primary decompsotion of implicitly
    defined geometrical objects). The paper starts with a brief summary of
    the Groebner bases method.",
  paper = "Buch87.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Buch97a,
  author = "Buchberger, B. and Jebelean, Tudor and Kriftner, Franz and
            Vasaru, Daniela",
  title = {{A Survey of the Theorema Project}},
  pages = "384-391",
  booktitle = "ISSAC '97",
  year = "1997",
  abstract =
    "The Theorema project aims at extending current computer algebra
    systems by facilities for supporting mathematical proving. The present
    early-prototype version of the Theorema software system is implemented
    in Mathematica 3.0. The system consists of a general higher-order
    predicate logic prover and a collection of special provers that call
    each other depending on the particular proof situations. The
    individual provers imitate the proof style of human mathematicians and
    aim at producing human-readable proofs in natural language presented
    in nested cells that facilitate studying the computer-generated proofs
    at various levels of detail. The special provers are intimately
    connected with the functors that build up the various mathematical
    domains.",
  paper = "Buch97a.pdf",
  keywords = "CAS-Proof"
}
```

———-

— **axiom.bib** —

```
@book{Buch81,
  author = "Buchberger, B. and Lichtenberger, F.",
  title = {{Mathematics for Computer Science I -- The Method of Mathematics}},
  publisher = "Springer",
  year = "1981",
  comment = "German"
}
```

———-

— **axiom.bib** —

```
@book{Buch96,
  author = "Buchberger, B.",
  title = {{Symbolic Computation: Computer Algebra and Logic}},
  booktitle = "Frontiers of Combining Systems",
  publisher = "Kluwer Academic",
  year = "1996",
  pages = "192-220",
  abstract =
    "In this paper we present our personal view of what should be the next
    step in the development of symbolic computation systems. The main
    point is that future systems should integrate the power of algebra and
    logic. We identify four gaps between the future ideal and the systems
    available at present: the logic, the syntax, the mathematics, and the
    prover gap, respectively. We discuss higher order logic without
    extensionality and with set theory as a subtheory as a logic frame for
    future systems and we propose to start from existing computer algebra
    systems and proceed by adding new facilities for closing the syntax,
    mathematics, and the prover gaps. Mathematica seems to be a
    particularly suitable candidate for such an approach. As the main
    technique for structuring mathematical knowledge, mathematical methods
    (including algorithms), and also mathematical proofs, we underline the
    practical importance of functors and show how they can be naturally
    embedded into Mathematica.",
  paper = "Buch96.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@article{Buch14,
  author = "Buchberger, Bruno",
  title = {{Soft Math Math Soft}},
  journal = "LNCS",
  volume = "8592",
  year = "2014",
  pages = "9-15",
  abstract =
    "In this talk we argue that mathematics is essentially software. In
    fact, from the beginning of mathematics, it was the goal of
    mathematics to automate problem solving. By systematic and deep
    thinking, for problems whose solution was difficult in each
    individual instance, systematic procedures were found that allow to
    solve each instance without further thinking. In each round of
    automation in mathematics, the deep thinking on spect ra of problem
    instances is reflected by deep theorems with deep proofs.",
  paper = "Buch14.pdf"
}
```

———-

— **axiom.bib** —

```
@incollection{Bund91,
  author = "Bundy, Alan",
  title = {{A Science of Reasoning (Extended Abstract)}},
  booktitle = "Computational Logic: Essays in Honor of Alan
               Robinson",
  year = "1991",
  publisher = "MIT Press",
  abstract =
    "How can we understand reasoning in general and mathematical proofs
    in particular? It is argued that a high-level understanding of proofs
    is needed to complement the low-level understanding provided by
    Logic. A role for computation is proposed to provide this high-level
    understanding, namely by the association of proof plans with
    proofs. Criteria are given for assessing the association of a proof
    plan with a proof.",
  paper = "Bund91.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Burg74,
  author = "William H. Burge",
  title = {{Stream Processing Functions}},
  year = "1974",
  month = "January",
  journal = "IBM Journal of Research and Development",
  volume = "19",
  issue = "1",
  pages = "12-25",
  papers = "Burg74.pdf",
  abstract = "
    One principle of structured programming is that a program should be
    separated into meaningful independent subprograms, which are then
    combined so that the relation of the parts to the whole can be clearly
    established.  This paper describes several alternative ways to compose
    programs. The main method used is to permit the programmer to denote
    by an expression the sequence of values taken on by a variable. The
    sequence is represented by a function called a stream, which is a
    functional analog of a coroutine. The conventional while and for loops
    of structured programming may be composed by a technique of stream
    processing (analogous to list processing), which results in more
    structured programs than the orignals. This technique makes it
    possible to structure a program in a natural way into its logically
    separate parts, which can then be considered independently."
}
```

———-

— **axiom.bib** —

```
@inproceedings{Burs77,
  author = "Burstall, R.M. and Goguen, J.A.",
  title = {{Putting THeories together to make Specifications}},
  booktitle = "IJCAI 77 Volume 2",
  pages = "1045-1058",
  year = "1977",
  paper = "Burs77.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Busw04,
  author = "Buswell, S. and Caprotti, O. and Carlisle, D.P. and Dewar, M.C.
            and Gaetano, M. and Kohlhase, M.",
  title = {{The OpenMath Standard}},
  year = "2004",
  link = "\url{https://www.openmath.org/standard/om20-2004-06-30/omstd20.pdf}",
  paper = "Busw04.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Butl96,
  author = "Butler, Greg",
  title = {{Software Architectures for Computer Algebra: A Case Study}},
  booktitle = "DISCO '96",
  pages = "277-286",
  year = "1996",
  abstract =
    "The architectures of the existing computer algebra systems have not
    been discussed sufficiently in the literature. Instead, the focus has
    been on the design of the related programming language, or the design
    of a few key data structures.

    We address this deficiency with a case study of the architecture of
    Cayley. Our aim is twofold: to capture this knowledge before the total
    passing of a system now made obsolete by Magma; and to encourage
    others to describe the architecture of the computer algebra systems
    with which they are familiar.

    The long-term goal is a better understanding of how to construct
    computer algebra systems in the future.",
  paper = "Butl96.pdf",
  keywords = "axiomref"
```

```
}
```

———-

1.41.3 C

— **axiom.bib** —

```
@book{Calm96,
  author = "Calmet, Jacques and Homann, Karsten",
  title = {{Classification of Communication and Cooperation Mechanisms for
           Logical and Symbolic Computation Systems}},
  booktitle = "Frontiers of Combining Systems",
  publisher = "Kluwer Academic",
  year = "1996",
  pages = "221-234",
  abstract =
    "The combination of logical and symbolic computation systems has
    recently emerged from prototype extensions of stand-alone systems to
    the study of environments allowing interaction among several
    systems. Communication and cooperation mechanisms of systems
    performing any kind of mathematical service enable one to study and
    solve new classes of problems and to perform efficient computation by
    distributed specialized packages. The classification of communication
    and cooperation methods for logical and symbolic computation systems
    given in this paper provides and surveys different methodologies for
    combining mathematical services and their characteristics,
    capabilities, requirements, and differences. The methods are
    illustrated by recent well-known examples. We separate the
    classification into communication and cooperation methods. The former
    includes all aspects of the physical connection, the flow of
    mathematical information, the communication language(s) and its
    encoding, encryption, and knowledge sharing. The latter concerns the
    semantic aspects of architectures for cooperative problem solving.",
  paper = "Calm96.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@article{Cant87,
  author = "Cantor, D.",
  title = {{Computing in the Jacobian of a HyperellipticCurve}},
  journal = "Mathematics of Computation",
  volume = "48",
  number = "177",
  month = "January",
  year = "1987",
  pages = "95-101",
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Carl14,
  author = "Carlisle, David and Ion, Patrick and Miner, Robert",
  title = {{Mathematical Markup Language}},
  year = "2014",
  link = "\url{https://www.w3.org/TR/MathML3/}"
}
```

———-

— **ignore** —

```
\bibitem[Carlson 65]{Car65} Carlson, B C
  title = {{On Computing Elliptic Integrals and Functions}},
J Math Phys. 44 36--51. (1965)
```

———-

— **ignore** —

```
\bibitem[Carlson 77a]{Car77a} Carlson B C
  title = {{Elliptic Integrals of the First Kind}},
SIAM J Math Anal. 8 231--242. (1977)
```

———-

— **ignore** —

```
\bibitem[Carlson 77b]{Car77b} Carlson B C
  title = {{Special Functions of Applied Mathematics}},
Academic Press. (1977)
```

———-

— **ignore** —

```
\bibitem[Carlson 78]{Car78} Carlson B C,
  title = {{Computing Elliptic Integrals by Duplication}},
(Preprint) Department of Physics, Iowa State University. (1978)
```

———-

— **ignore** —

```
\bibitem[Carlson 88]{Car88} Carlson B C,
  title = {{A Table of Elliptic Integrals of the Third Kind}},
Math. Comput. 51 267--280. (1988)
```

———-

— **axiom.bib** —

```
@article{Carl02,
  author = "Carlstrom, Jesper",
  title = {{Subsets, Quotients, and Partial Functions in Martin-L\"of's
           Type Theory}},
  journal = "LNCS",
  volume = "2646",
  year = "2002",
  pages = "78-94",
  abstract =
    "We treat subsets, equivalence relations, and partial functions,
    with subsets as propositional functions .In order for these three
    notions to work well together, we propose some changes to the theory
    of subsets as propositional functions .The method used is not to make
    any changes to the type theory itself, but to view the new concepts as
    defined ones.",
  paper = "Carl02.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Cauchy 1829]{Cau1829} Augustin-Lux Cauchy
  title = {{Exercices de Math\'ematiques Quatri\`eme Ann\'ee.
           De Bure Fr\`eres}},
Paris 1829 (reprinted Oeuvres, II S\'erie, Tome IX,
Gauthier-Villars, Paris, 1891).
```

———-

— **axiom.bib** —

```
@book{Char92,
  author = "Char, B.W. and Geddes, K.O. and Gonnet, G.H. and Leong, B.L.
           and Monagan, M.B. and Watt, S.M.",
  title = {{A Tutorial Introduction to Maple V}},
  year = "1992",
  publisher = "Springer-Verlag"
}
```

———-

— **axiom.bib** —

```
@article{Cher80,
  author = "Cherlin, Gregory",
  title = {{Rings of Continuous Functions: Decision Problems}},
  journal = "Lecture Notes in Mathematics",
  volume = "834",
  pages = "44-91",
  year = "1980",
  link = "\url{https://link.springer.com/content/pdf/10.1007/BFb0090160.pdf}",
  paper = "Cher80.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Chew95,
  author = "Chew, Paul and Constable, Robert L. and Pingali, Keshav and
            Vavasis, Steve and Zippel, Richard",
  title = {{Collaborative Mathematics Environment}},
  link = "\url{http://www.cs.cornell.edu/rz/MathBus95/TechSummary.html}",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Ch\‘eze 07]{Chez07} Ch\’eze, Guillaume; Lecerf, Gr\’egoire
  title = {{Lifting and recombination techniques for absolute factorization}},
Journal of Complexity, VOl 23 Issue 3 June 2007 pp 380-420
  link = "\url{http://www.sciencedirect.com/science/article/pii/S0885064X07000465}",
  abstract = "
    In the vein of recent algorithmic advances in polynomial factorization
    based on lifting and recombination techniques, we present new faster
    algorithms for computing the absolute factorization of a bivariate
    polynomial. The running time of our probabilistic algorithm is less
    than quadratic in the dense size of the polynomial to be factored.",
  paper = "Chez07.pdf"
```

———-

— **ignore** —

```
\bibitem[Childs 79]{CSDDN79} Childs B; Scott M; Daniel J W; Denman E;
Nelson P (eds)
  title = {{Codes for Boundary-value Problems in Ordinary Differential
           Equations}},
Lecture Notes in Computer Science. 76 (1979) Springer-Verlag
```

———-

— **axiom.bib** —

```
@article{Chur40,
  author = "Church, Alonzo",
  title = {{A Formulation of the Simple Theory of Types}},
  journal = "J. of Symbolic Logic",
  volume = "5",
  number = "2",
  year = "1940",
  pages = "56-68",
  abstract =
    "The purpose of the present paper is to give a formulation of the
    simple theory of types which incorporates certain features of the
    calculus of $\lambda$-conversion. A complete incorporation of the
    calculus of $\lambda$-conversion into the theory of types is
    impossible if we require that $\lambda x$ and juxtaposition shall
    retain their respective meanings as an abstraction operator and as
    denoting the application of function to argument. But the present
    partial incorporation has certain advantages from the point of view of
    type theory and is offered as being of interest on this basis
    (whatever may be thought of the finally satisfactory character of the
    theory of types as a foundation for logic and mathematics).",
  paper = "Chur40.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Clar92,
  author = "Clarke, Edmund and Zhao, Xudong",
  title = {{Analytica -- An Experiment in Combining Theorem Proving and
           Symbolic Computation}},
  type = "technical report",
  institution = "Carnegie Mellon University",
  number = "CMU-CS-92-147",
  year = "1992",
  abstract =
    "Analytica is an automatic theorem prover for theorems in elementary
    analysis. The prover is written in Mathematica language and runs in
    the Mathematica environment. The goal of the project is to use a
    powerful symbolic computation system to prove theorems that are beyond
    the scope of previous automatic theorem provers. The theorem prover is
    also able to guarantee the correctness of certain steps that are made
    by the symbolic computation system and therefore prevent common errors
    like division by a symbolic expression that could be zero. In this
    paper we describe the structure of Analytica and explain the main
    techniques that it uses to construct proofs. Analytica has been able
    to prove several non-trivial examples including the basic properties
    of the stereographic projection and a series of three lemmas that lead
```

```
    to a proof of Weierstrass''s example of a continuous nowhere
    differentiable function. Each of the lemmas in the latter example is
    proved completely automatically.",
  keywords = "CAS-Proof"
}
```

———-

— **axiom.bib** —

```
@techreport{Clar94,
  author = "Clarke, Edmund and Zhao, Xudong",
  title = {{Combining Symbolic Computation and Theorem Proving: Some
           Problems of Ramanujan}},
  year = "1994",
  type = "technical report",
  institution = "Carnegie Mellon University",
  number = "CMU-CS-94-103",
  abstract =
    "One way of building more powerful theorem provers is to use
    techniques from symbolic computation. The challenge problems in this
    paper are taken from Chapter 2 of Ramanujan''s Notebooks. They were
    selected because they are non-trivial and require the use of symbolic
    computation techniques. We have developed a theorem prover based on
    the symbolic computation system Mathematica, that can prove all the
    challenge problems completely automatically. The axioms and inference
    rules for constructing the proofs are also briefly discussed.",
  paper = "Clar94.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **ignore** —

```
\bibitem[Clausen 89]{Cla89} Clausen, M.; Fortenbacher, A.
  title = {{Efficient Solution of Linear Diophantine Equations}},
JSC (1989) 8, 201-216
```

———-

— **ignore** —

```
\bibitem[Clenshaw 55]{Cle55} Clenshaw C W,
  title = {{A Note on the Summation of Chebyshev Series}},
Math. Tables Aids Comput. 9 118--120. (1955)
```

———-

— **ignore** —

```
\bibitem[Clenshaw 60]{Cle60} Clenshaw C W
  title = {{Curve Fitting with a Digital Computer}},
Comput. J. 2 170--173. (1960)
```

———-

— **ignore** —

```
\bibitem[Clenshaw 62]{Cle62} Clenshaw C W
  title = {{Mathematical Tables. Chebyshev Series for Mathematical Functions}},
HMSO. (1962)
```

———-

— **ignore** —

```
\bibitem[Cline 84]{CR84} Cline A K; Renka R L,
  title = {{A Storage-efficient Method for Construction of a Thiessen
           Triangulation}},
Rocky Mountain J. Math. 14 119--139. (1984)
```

———-

— **axiom.bib** —

```
@article{Coen07,
  author = "Coen, Claudio Sacerdoti and Tassi, Enrico",
  title = {{Working with Mathematical Structures in Type Theory}},
  journal = "LNCS",
  volume = "4941",
  year = "2007",
  pages = "157-172",
  abstract =
    "We address the problem of representing mathematical structures in a
    proof assistant which: 1) is based on a type theory with dependent
    types, telescopes and a computational version of Leibniz equality; 2)
    implements coercive subtyping, accepting multiple coherent paths
    between type families; 3) implements a restricted form of higher order
    unification and type reconstruction. We show how to exploit the
    previous quite common features to reduce the ``syntactic'' gap between
    pen and paper and formalised algebra. However, to reach our goal we need
    to propose unification and type reconstruction heuristics that are
    slightly different from the ones usually implemented. We have
    implemented them in Matita.",
  paper = "Coen07.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Cohe80,
  author = "Cohen, J.D. and Jenks, R.D.",
  title = {{On Resolution and Coercion in MODLISP}},
  comment = "in preparation",
  year = "1980"
}
```

———-

— **axiom.bib** —

```
@book{Cohn65,
  author = "Cohn, Paul Moritz",
  title = {{Universal Algebra}},
  publisher = "Harper and Row",
  year = "1965"
}
```

———-

— **axiom.bib** —

```
@book{Cohn91,
  author = "Cohn, Paul Moritz",
  title = {{Algebra (2nd Ed.)}},
  publisher = "Wiley",
  isbn = "978-0471101697",
  year = "1991",
  paper = "Cohn91.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Coll85,
  author = "Collins, George E.",
  title = {{The SAC-2 Computer Algebra System}},
  booktitle = "Proc. Europena Conf. on Computer Algebra",
  year = "1985",
  pages = "34-35",
  paper = "Coll85.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Colm90,
```

```
  author = "Colmerauer, Alain",
  title = {{An Introduction to Prolog III}},
  journal = "CACM",
  volume = "33",
  number = "7",
  year = "1990",
  pages = "69-90",
  abstract =
    "The Prolog III programming language extends Prolog by redefining the
    fundamental process at its heart: unification. This article presents
    the specifications of this new language and illustrates its capabilities.",
  paper = "Colm90.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Conway 87]{CCNPW87} Conway, J.; Curtis, R.; Norton, S.; Parker, R.;
Wilson, R.
  title = {{Atlas of Finite Groups}},
Oxford, Clarendon Press, 1987
```

———-

— **ignore** —

```
\bibitem[Conway 03]{CS03} Conway, John H.; Smith, Derek, A.
  title = {{On Quaternions and Octonions}},
A.K Peters, Natick, MA. (2003) ISBN 1-56881-134-9
```

———-

— **axiom.bib** —

```
@article{Coqu98,
  author = "Coquand, Thierry and Persson, Henrik",
  title = {{Groebner Bases in Type Theory}},
  journal = "LNCS",
  volume = "1657",
  year = "1998",
  pages = "33-46",
  abstract =
    "We describe how the theory of Groebner bases, an important part
    of computational algebra, can be developed within Martin-Lofs
    type theory. In particular, we aim for an integrated development
    of the algorithms for computing Groebner bases: we want to prove,
    constructively in type theory, the existence of Groebner bases and
    from such proofs extract the algorithms. Our main contribution is
    a reformulation of the standard theory of Groebner bases which
    uses generalised inductive definitions. We isolate the main
```

```
    nonconstructive part, a minimal bad sequence argument, and use
    the open induction principle [Rao88,Coq92] to interpret it by
    induction. This leads to short constructive proofs of Dicksons
    lemma and Hilberts basis theorem, which are used to give an
    integrated development of Buchbergers algorithm. An important
    point of this work is that the elegance and brevity of the
    original proofs are maintained while the new proofs also have a
    direct constructive content.  In the appendix we present a
    computer formalisation of Dicksons lemma and an abstract
    existence proof of Groebner bases.",
  paper = "Coqu98.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Corl92,
  author = "Corless, Robert M. and Jeffrey, David J.",
  title = {{Well, it isn't quite that simple}},
  journal = "ACM SIGSAM",
  volume = "26",
  number = "3",
  year = "1992",
  abstract =
    "Present computer algebra systems base their interactive sessions on a
    very simple model of mathematical discourse. The user's input to the
    system is a line containing a mathematical expression (an operation, a
    formula, a set of equations, etc) and the system's response to the
    user is an output line which contains a mathematical expression
    similar to the input. There are many situations, however, in which
    this is too simple a model of mathematics. Algebra systems should be
    allowed to reply 'Well ... it isn't quite that simple'.",
  paper = "Corl92.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Cox 72]{Cox72} Cox M G
  title = {{The Numerical Evaluation of B-splines}},
J. Inst. Math. Appl. 10 134--149. (1972)
```

———-

— **ignore** —

```
\bibitem[CH 73]{CH73} Cox M G; Hayes J G
  title = {{Curve fitting: a guide and suite of algorithms for the
           non-specialist user}},
```

```
Report NAC26. National Physical Laboratory.  (1973)
```

———-

— **ignore** —

```
\bibitem[Cox 74a]{Cox74a} Cox M G
  title = {{A Data-fitting Package for the Non-specialist User}},
Software for Numerical Mathematics. (ed D J Evans) Academic Press. (1974)
```

———-

— **ignore** —

```
\bibitem[Cox 74b]{Cox74b} Cox M G
  title = {{Numerical methods for the interpolation and approximation of
           data by spline functions}},
PhD Thesis. City University, London. (1975)
```

———-

— **ignore** —

```
\bibitem[Cox 75]{Cox75} Cox M G
  title = {{An Algorithm for Spline Interpolation}},
J. Inst. Math. Appl. 15 95--108. (1975)
```

———-

— **ignore** —

```
\bibitem[Cox 77]{Cox77} Cox M G
  title = {{A Survey of Numerical Methods for Data and Function
            Approximation}},
The State of the Art in Numerical Analysis. (ed D A H Jacobs)
Academic Press. 627--668. (1977)
 keywords = "survey",
```

———-

— **ignore** —

```
\bibitem[Cox 78]{Cox78} Cox M G
  title = {{The Numerical Evaluation of a Spline from its B-spline
            Representation}},
J. Inst. Math. Appl. 21 135--143. (1978)
```

———-

— **axiom.bib** —

```
@article{Crai92,
  author = "Craigen, Dan and Kromodimoeljo, Sentot and Meisels, Irwin
            and Pase, Bill",
  title = {{Eves System Description}},
  journal = "Lecture Notes in Computer Science",
  volume = "607",
  pages = "771-775",
  year = "1992",
  paper = "Crai92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Cunn85,
  author = "Cunningham, R.J. and Dick, A.J.J",
  title = {{Rewrite Systems on a Lattice of Types}},
  journal = "Acta Informatica",
  volume = "22",
  pages = "149-169",
  year = "1985",
  abstract =
    "Re-writing systems for partial algebras are developed by modifying
    the Knuth-Bendix completion algorithm to permit the use of
    lattice-structured domains. Some problems with the original algorithm,
    such as the treatment of division rings, are overcome conveniently by
    this means. The use of a type lattice also gives a natural framework
    for specifying data types in Computer Science without over-specifying
    error situations. The soundness and meaning of the major concepts
    involed in re-writing systems are reviewed when applied to such
    structures.",
  paper = "Cunn85.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Curtis 74]{CPR74} Curtis A R; Powell M J D; Reid J K
  title = {{On the Estimation of Sparse Jacobian Matrices}},
J. Inst. Maths Applics. 13 117--119. (1974)
```

———-

1.41.4 D

— **ignore** —

```
\bibitem[Dahlquist 74]{DB74} Dahlquist G; Bjork A
  title = {{Numerical Methods}},
Prentice- Hall. (1974)
```

———-

— **ignore** —

```
\bibitem[Dalmas 98]{DA98} Dalmas, Stephane; Arsac, Olivier
  title = {{The INRIA OpenMath Library}},
Projet SAFIR, INRIA Sophia Antipolis Nov 25, 1998
```

———-

— **axiom.bib** —

```
@inproceedings{Daly88a,
  author = "Daly, Timothy and Kastner, John and Mays, Eric",
  title = {{Integrating Rules and Inheritance Networks in a Knowlege-based
            Financial Marketing Consutation System}},
  booktitle = "Proc. HICSS 21",
  volume = "3",
  number = "5-8",
  pages = "496-500",
  year = "1988",
  comment = "KROPS",
  abstract =
    "This paper describes the integrated use of rule-bascd inference and
    an object-centered knowledge representation (inheritance network) in a
    financial marketing consultation system. The rules provide a highly
    flexible pattern match capability and inference cycle for control. The
    inheritance network provides a convenient way to represent the
    conceptual structure of the domain. By merging the two techniques, our
    financial computation can he shared at the most general level, and
    rule inference is carried out at any appropriate Ievel of
    generalization.  Since domain knowledge is representcd independently
    from control knowledge, knowledge ahout a particular problcm solving
    technique is decoupled from the conditions for its invocation. A Iarge
    financial marketing system has been built and examples are given to
    show our combined use of rules and inheritance networks.",
  paper = "Daly88a.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Damg91,
  author = "Damgard, I. and Landrock, P.",
  title = {{Improved Bounds for the Rabin Primality Test}},
  booktitle = "Proc. 3rd IMA Conf. on Coding and Cryptography",
  year = "1991"
}
```

———-

— **ignore** —

```
\bibitem[Dantzig 63]{Dan63} Dantzig G B
  title = {{Linear Programming and Extensions}},
Princeton University Press. (1963)
```

———-

— **axiom.bib** —

```
@misc{Dave80b,
  author = "Davenport, J.H. and Jenks, R.D.",
  title = {{SCRATCHPAD/370: Modes and Domains}},
  year = "1980",
  comment = "private communication",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Dave81b,
  author = "Davenport, James H.",
  title = {{Effective Mathematics: The Computer Algebra Viewpoint}},
  journal = "Springer Lecture Notes in Mathematics",
  volume = "873",
  pages = "31-43",
  year = "1981",
  abstract =
    "This paper is written in an attempt to explain the field of Computer
    Algebra (see e.g. Stoutemyer and Yun, 1980. Recent progress is
    summarised in the proceedings edited by Ng[1979]) to the
    mathematician, and to explain what its mathematical content is, and
    why it raises interesting questions of the effectiveness of various
    common mathematical operations. The fact that effectiveness is not
    always trivial is illustrated by the result of Richardson [1968]
    quoted in the next section.

    I hope that it will become clear that many operations that are often
    regarded as effective, are in fact not so; while many operations that
    might not be regarded as effective in fact are.",
```

```
  paper = "Dave81b.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Dave87,
  author = "Davenport, James H. and Smith, G.C.",
  title = {{Rabin's Primality Testing Algorithm -- a Group Theory View}},
  school = "University of Bath",
  type = "technical report",
  number = "87-04",
  year = "1987"
}
```

———-

— **ignore** —

```
\bibitem[Davenport]{Dav} Davenport, James
  title = {{On Brillhart Irreducibility.}},
To appear.
```

———-

— **ignore** —

```
\bibitem[Davenport 93]{Ref-Dav93} Davenport, J.H.
  title = {{Primality testing revisited}},
Technical Report TR2/93
(ATR/6)(NP2556) Numerical Algorithms Group, Inc., Downer's Grove, IL, USA
and Oxford, UK, August 1993
  link = "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
```

———-

— **ignore** —

```
\bibitem[Davis 67]{DR67} Davis P J; Rabinowitz P
  title = {{Numerical Integration}},
Blaisdell Publishing Company. 33--52. (1967)
```

———-

— **ignore** —

```
\bibitem[Davis 75]{DR75} Davis P J; Rabinowitz P
  title = {{Methods of Numerical Integration}},
```

```
Academic Press. (1975)
```

———-

— **ignore** —

```
\bibitem[DeBoor 72]{DeB72} De Boor C
  title = {{On Calculating with B-splines}},
J. Approx. Theory. 6 50--62. (1972)
```

———-

— **ignore** —

```
\bibitem[De Doncker 78]{DeD78} De Doncker E,
  title = {{An Adaptive Extrapolation Algorithm for Automatic Integration}},
Signum Newsletter. 13 (2) 12--18. (1978)
```

———-

— **axiom.bib** —

```
@techreport{Deme79,
  author = "Demers, Alan and Donahue, James",
  title = {{Revised Report on RUSSELL}},
  year = "1979",
  type = "technical report",
  institution = "Cornell",
  number = "TR 79-389"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Deme80,
  author = "Demers, Alan and Donahue, James",
  title = {{Type Completeness as a Language Principle}},
  booktitle = "POPL 80",
  publisher = "ACM",
  pages = "234-244",
  year = "1980",
  abstract =
    "The problem of Von Neumann languages is that their changeable
    parts have so little expressive power -- John Backus",
  paper = "Deme80.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Demmel 89]{Dem89} Demmel J W
  title = {{On Floating-point Errors in Cholesky}},
LAPACK Working Note No. 14. University of Tennessee, Knoxville. 1989
```

———-

— **ignore** —

```
\bibitem[Dennis 77]{DM77} Dennis J E Jr; More J J
  title = {{Quasi-Newton Methods, Motivation and Theory}},
SIAM Review. 19 46--89. 1977
```

———-

— **ignore** —

```
\bibitem[Dennis 81]{DS81} Dennis J E Jr; Schnabel R B
  title = {{A New Derivation of Symmetric Positive-Definite Secant Updates}},
Nonlinear Programming 4. (ed O L Mangasarian, R R Meyer and S M. Robinson)
Academic Press. 167--199. (1981)
```

———-

— **ignore** —

```
\bibitem[Dennis 83]{DS83} Dennis J E Jr; Schnabel R B
  title = {{Numerical Methods for Unconstrained Optimixation and
            Nonlinear Equations}},
Prentice-Hall.(1983)
```

———-

— **axiom.bib** —

```
@article{Denz94,
  author = "Denzinger, Jorg and Fuchs, Matthias",
  title = {{Goal oriented equational theorem proving using team work}},
  journal = "Lecture Notes in Computer Science",
  volume = "861",
  pages = "343-354",
  year = "1994",
  abstract =
    "The team work method is a concept for distributing automated theorem
    provers by activating several experts to work on a problem. We have
    implemented this for pure equational logic using the unfailing
    Knuth-Bendix completion procedure as basic prover. In this paper we
    present three classes of experts working in a goal oriented
    fashion. In general, goal oriented experts perform their job unfair
```

```
    and so are often unable to solve a given problem alone. However, as
    team members in the team work method they perform highly efficiently,
    as we demonstrate by examples, some of which can only be proved using
    team work.",
  paper = "Denz94.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Dewa00,
  author = "Dewar, Mike",
  title = {{Special Issue on OPENMATH}},
  publisher = "ACM SIGPLAN Bulletin",
  volume = "34",
  number = "2",
  year = "2000"
}
```

———-

— **ignore** —

```
\bibitem[Dierckx 75]{Die75} Dierckx P
  title = {{An Algorithm for Smoothing, Differentiating and Integration of
            Experimental Data Using Spline Functions}},
J. Comput. Appl. Math. 1 165--184. (1975)
```

———-

— **ignore** —

```
\bibitem[Dierckx 81]{Die81} Dierckx P
  title = {{An Improved Algorithm for Curve Fitting with Spline Functions}},
Report TW54. Dept. of Computer Science, Katholieke Universiteit Leuven. 1981
```

———-

— **ignore** —

```
\bibitem[Dierckx 82]{Die82} Dierckx P
  title = {{A Fast Algorithm for Smoothing Data on a Rectangular Grid
            while using Spline Functions}},
SIAM J. Numer. Anal. 19 1286--1304. (1982)
```

———-

,

— **axiom.bib** —

```
@misc{Dolz96,
  author = "Dolzmann, Andreas and Sturm, Thomas",
  title = {{Redlog User Manual Edition 1.0}},
  year = "1996",
  abstract =
    "REDLOG stands for REDUCE LOGIC system.  It provides an extension
    of the computer algebra system reduce to a ``computer logic system''
    implementing symbolic algorithms on first-order formulas wrt.
    temporarily fixed first-order languages and theories.  Underlying
    theories currently available are ordered fields and discretely
    valued fields.  Though the focus of the implemented algorithms is on
    effective quantier elimination and simplication of quantier-free
    formulas, REDLOG is intended and designed as an all-purpose system.
    REDLOG is freely available to the scientic community.",
  paper = "Dolz96.pdf",
  keywords = "printed"
}
```

———-

,

— **axiom.bib** —

```
@article{Dolz97b,
  author = "Dolzmann, Andreas and Sturm, Thomas",
  title = {{REDLOG: Computer Algebra meets Computer Logic}},
  journal = "ACM SIGSAM Bulletin",
  volume = "31",
  number = "2",
  year = "1997",
  pages = "2-9",
  abstract =
    "REDLOG is a package that extends the computer algebra system REDUCE
    to a computer logic system, i.e., a system that provides algorithms
    for the symbolic manipulation of first-order formulas over some
    temporarily fixed language and theory. In contrast to theorem provers,
    the methods applied know about the underlying algebraic theory and
    make use of it. We illustrate some applications of REDLOG, describe
    its functionality as it appears to the user, and explain the design
    issues and implementation techniques. REDLOG is available on the WWW.",
  paper = "Dolz97b.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **ignore** —

```
\bibitem[Dongarra 79]{DMBS79} Dongarra J J; Moler C B; Bunch J R;
Stewart G W
  title = {{LINPACK Users' Guide}},
SIAM, Philadelphia. (1979)
```

———-

— **ignore** —

```
\bibitem[Dongarra 85]{DCHH85} Dongarra J J; Du Croz J J; Hammarling S;
Hanson R J
  title = {{A Proposal for an Extended set of Fortran Basic Linear
           Algebra Subprograms}},
SIGNUM Newsletter. 20 (1) 2--18. (1985)
```

———-

— **ignore** —

```
\bibitem[Dongarra 88]{REF-DON88} Dongarra, Jack J.; Du Croz, Jeremy;
Hammarling, Sven; Hanson, Richard J.
  title = {{An Extended Set of FORTRAN Basic Linear Algebra Subroutines}},
ACM Transactions on Mathematical Software, Vol 14, No 1, March 1988,
pp 1-17
```

———-

— **ignore** —

```
\bibitem[Dongarra 88a]{REF-DON88a} Dongarra, Jack J.; Du Croz, Jeremy;
Hammarling, Sven; Hanson, Richard J.
  title = {{ALGORITHM 656: An Extended Set of Basic Linear Algebra
           Subprograms: Model Implementation and Test Programs}},
ACM Transactions on Mathematical Software, Vol 14, No 1, March 1988,
pp 18-32
```

———-

— **ignore** —

```
\bibitem[Dongarra 90]{REF-DON90} Dongarra, Jack J.; Du Croz, Jeremy;
Hammarling, Sven; Duff, Iain S.
  title = {{A Set of Level 3 Basic Linear Algebra Subprograms}},
ACM Transactions on Mathematical Software, Vol 16, No 1, March 1990,
pp 1-17
```

———-

— **ignore** —

```
\bibitem[Dongarra 90a]{REF-DON90a} Dongarra, Jack J.; Du Croz, Jeremy;
Hammarling, Sven; Duff, Iain S.
  title = {{ALGORITHM 679: A Set of Level 3 Basic Linear Algebra
           Subprograms: Model Implementation and Test Programs}},
ACM Transactions on Mathematical Software, Vol 16, No 1, March 1990,
```

pp 18-28

———-

— **axiom.bib** —

```
@misc{blas01,
  author = "Dongarra, Jack and et al.",
  title = {{Basic Linear Algebra Subprograms Technical (BLAST)
           Forum Standard}},
  year = "2001",
  link = "\url{http://www.netlib.org/blas/blast-forum/blas-report.pdf}",
  paper = "blas01.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dowe00,
  author = "Dowek, Gilles",
  title = {{Axioms vs Rewrite Rules: From Completeness to Cut Elimination}},
  journal = "Lecture Notes in Computer Science",
  volume = "1794",
  pages = "62-72",
  year = "2000",
  abstract =
    "Combining a standard proof search method, such as resolution or
    tableaux, and rewriting is a powerful way to cut off search space in
    automated theorem proving, but proving the completeness of such
    combined methods may be challenging. It may require in particular to
    prove cut elimination for an extended notion of proof that combines
    deductions and computations. This suggests new interactions between
    automated theorem proving and proof theory.",
  paper = "Dowe00.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Dowe96,
  author = "Dowek, Gilles",
  title = {{A Type-Free Formalization of Mathematics Where Proofs are
            Objects}},
  journal = "LNCS",
  volume = "1512",
  year = "1996",
  pages = "88-111",
  abstract =
    "We present a first-order type-free axiomatization of mathematics
```

```
    where proofs are objects in the sense of Heyting-Kolmogorov functional
    interpretation.  The consistency of this theory is open.",
  paper = "Dowe96.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Ducos 00]{Duc00} Ducos, Lionel
  title = {{Optimizations of the subresultant algorithm}},
Journal of Pure and Applied Algebra V145 No 2 Jan 2000 pp149-163
```

———-

— **ignore** —

```
\bibitem[Duff 77]{Duff77} Duff I S,
  title = {{MA28 -- a set of Fortran subroutines for sparse
           unsymmetric linear equations}},
A.E.R.E. Report R.8730. HMSO. (1977)
```

———-

— **axiom.bib** —

```
@misc{Duns01,
  author = "Dunstan, M.N. and Gottliebsen, H. and Kelsey, T.W. and
            Martin, U.",
  title = {{Computer Algebra meets Automated Theorem Proving: A
           Maple-PVS Interface}},
  booktitle = "Proc. Calculemus, 2001",
  year = "2001",
  paper = "Duns01.pdf",
  keywords = "axiomref, CAS-proof, printed"
}
```

———-

— **axiom.bib** —

```
@article{Dute96,
  author = "Dutertre, Bruno",
  title = {{Elements of Mathematical Analysis in PVS}},
  journal = "LNCS",
  volume = "1125",
  pages = "141-156",
  year = "1996",
  abstract =
    "This paper presents the formalization of some elements of
```

```
    mathematical analysis using the PVS verification system. Our main
    motivation was to extend the existing PVS libraries and provide means
    of modelling and reasoning about hybrid systems. The paper focuses on
    several important aspects of PVS including recent extensions of the
    type system and discusses their merits and effectiveness. We conclude
    by a brief comparison with similar developments using other theorem
    provers.",
  paper = "Dute96.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Duva96a,
  author = {Duval, Dominique and Gonz\'alez-Vega, L.},
  title = {{Dynamic Evaluation and Real Closure}},
  journal = "Mathematics and Computers in Simulation",
  volume = "42",
  pages = "551-560",
  year = "1996",
  abstract = "
    The aim of this paper is to present how the dynamic evaluation method
    can be used to deal with the real closure of an ordered field. Two
    kinds of questions, or tests, may be asked in an ordered field:
    equality tests $(a=b?)$ and sign tests $(a > b?)$. Equality tests are
    handled through splittings, exactly as in the algebraic closure of a
    field. Sign tests are handled throug a structure called ``Tarski data
    type''.",
  paper = "Duva96a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Duva96,
  author = "Duval, D. and Reynaud, Jean-Claude",
  title = {{Sketches and Computations over Fields}},
  journal = "Mathematics and Computers in Simulation",
  volume = "42",
  pages = "363-373",
  year = "1996",
  abstract =
    "The goal of this short paper is to describe one possible use of
    sketches in computer algebra. We show that sketches are a powerful
    tool for the description of mathematical structures and for the
    description of computations.",
  paper = "Duva96.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Duva94a,
  author = "Duval, D. and Reynaud, J.C.",
  title = {{Sketches and Computation (Part I):
           Basic Definitions and Static Evaluation}},
  journal = "Mathematical Structures in Computer Science",
  volume = "4",
  pages = "185-238",
  publisher =  "Cambridge University Press",
  year = "1994",
  link = "\url{http://journals.cambridge.org/abstract_S0960129500000438}",
  abstract =
    "We define a categorical framework, based on the notion of {\sl
    sketch}, for specification and evaluation in the sense of algebraic
    specifications and algebraic programming. This framework goes far
    beyond our initial motivations, which was to specify computation with
    algebraic numbers. We begin by redefining sketches in order to deal
    explicitly with programs. Expressions and terms are carefully defined
    and studied, then {\sl quasi-projective sketches} are introduced. We
    describe {\sl static evaluation} in these sketches: we propose a
    rigorous basis for evaluation in the corresponding structures. These
    structures admit an initial model, but are not necessarily
    equational. In Part II (Duval and Reynaud 1994), we study a more
    general process, called {\sl dynamic evaluation}, for structures that
    may have no initial model.",
  paper = "Duva94a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Duva94b,
  author = "Duval, Dominique and Reynaud, Jean-Claude",
  title = {{Sketches and Computation (Part II):
           Dynamic Evaluation and Applications}},
  journal = "Mathematical Structures in Computer Science",
  volume = "4",
  pages = "239-271",
  publisher = "Cambridge University Press",
  year = "1994",
  link = "\url{http://journals.cambridge.org/abstract_S096012950000044X}",
  abstract =
    "In the first part of this paper (Duval and Reynaud 1994), we defined a
    categorical framework, based on the notion of {\sl sketch}, for
    specification and evaluation in the senses of algebraic specification
    and algebraic programming. {\sl Static evaluation} in {\sl
    quasi-projective sketches} was defined in Part I; in this paper, {\sl
    dynamic evaluation} is introduced. It deals with more general
    structures, which may have no initial model. Until now, this process
```

```
    has not been used in algebraic specification systems, but computer
    algebra systems are beginning to use it as a basic tool. Finally, we
    give some applications of dynamic evaluation to computation in field
    extensions.",
  paper = "Duva94b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Duva94c,
  author = "Duval, Dominique",
  title = {{Algebraic Numbers: An Example of Dynamic Evaluation}},
  journal = "J. Symbolic Computation",
  volume = "18",
  pages = "429-445",
  year = "1994",
  link = "\url{http://www.sciencedirect.com/science/article/pii/S0747717106000551}",
  abstract = "
    Dynamic evaluation is presented through examples: computations
    involving algebraic numbers, automatic case discussion according to
    the characteristic of a field. Implementation questions are addressed
    too. Finally, branches are presented as ‘‘dual’’ to binary functions,
    according to the approach of sketch theory.",
  paper = "Duva94c.pdf",
  keywords = "axiomref"
}
```

———-

1.41.5 E

— **axiom.bib** —

```
@phdthesis{Elbe98,
  author = "Elbers, Hugo Johannes",
  title = {{Connecting Informal and Formal Mathematics}},
  year = "1998",
  school = "Technische Universiteit Eindhoven",
  paper = "Elbe98.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Enge65,
  author = "Engelman, C.",
  title = {{A Program for On-Line Assistance in Symbolic Computation}},
```

```
  booktitle = "Proc. Fall Joint Comput. Conf. 2",
  year = "1965",
  publisher = "Spartan Books"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Eras10,
  author = "Erascu, Madalina and Jebelean, Tudor",
  title = {{A Purely Logical Approach to Program Termination}},
  booktitle = "11th Int. Workshop on Termination",
  year = "2010",
  comment = "Extended Abstract",
  link =
  "\url{http://www.risc.jku.at/publications/download/risc_4089/ErascuJebeleanWSTFinal.pdf}",
  paper = "Eras10.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Eras10a,
  author = "Erascu, Madalina and Jebelean, Tudor",
  title = {{A Purely Logical Approach to Imperative Program Verification}},
  year = "2010",
  institution = "RISC Linz",
  type = "technical report",
  number = "10-07",
  link =
  "\url{http://www.risc.jku.at/publications/download/risc_4088/techRep.pdf}",
  paper = "Eras10a.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Eras10b,
  author = "Erascu, Madalina and Jebelean, Tudor",
  title = {{A Purely Logical Approach to Termination of Imperative Loops}},
  booktitle = "Proc. 12th Int. Symp. on SYmbolic and Numeric
               Algorithms for Scientific Computing",
  pages = "142-149",
  year = "2010",
  link = "\url{http://www.risc.jku.at/publications/download/risc_4181/synasc_postproceedings.pdf}",
  abstract =
    "We present and illustrate a method for the generation of the
    termination conditions for nested loops with abrupt termination
```

```
    statements.  The conditions are (first-order) formulae obtained by
    certain transformations of the program text.  The loops are treated
    similarly to calls of recursively defined functions.  The program text
    is analyzed on all possible execution paths by forward symbolic
    execution using certain meta-level functions which define the syntax,
    the semantics, the verification conditions for the partial
    correctness, and the termination conditions. The termination
    conditions are expressed as induction principles, however, still in
    first-order logic.

    Our approach is simpler than others because we use
    neither an additional model for program execution, nor a fixpoint
    theory for the definition of program semantics.  Because the
    meta-level functions are fully formalized in predicate logic, it is
    possible to prove in a purely logical way and at object level that the
    verification conditions are necessary and sufficient for the existence
    and uniqueness of the function implemented by the program.",
  paper = "Eras10b.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Eras11,
  author = "Erascu, Madalina",
  title = {{Symbolic Computation and Program Verification. Proving
            Partial Correctness and Synthesizing Optimal Algorithms}},
  type = "technical report",
  number = "11-15",
  institution = "RISC Linz",
  year = "2011",
  abstract =
    "We present methods for checking the partial correctness of,
    respectively to optimize, imperative programs, using polynomial
    algebra methods, namely resultant computation and quantifier
    elimination (QE) by cylindrical algebraic decomposition (CAD). The
    result are very promising but also show that there is room for
    improvement of algebraic algorithms.",
  paper = "Eras11.pdf"
}
```

———-

1.41.6 F

— **axiom.bib** —

```
@inproceedings{Fate90a,
  author = "Fateman, Richard J.",
  title = {{Advances and Trends in the Design of Algebraic
```

```
            Manipulation Systems}},
  booktitle = "Proc. ISSAC'90",
  pages = "60-67",
  year = "1990"
}
```

———-

— **axiom.bib** —

```
@misc{Fate08,
  author = "Fateman, Richard J.",
  title = {{Revisiting numeric/symbolic indefinite integration of rational
           functions, and extensions}},
  link = "\url{http://www.eecs.berkeley.edu/~fateman/papers/integ.pdf}",
  abstract = "
    We know we can solve this problem: Given any rational function
    $f(x)=p(x)/q(x)$, where $p$ and $q$ are univariate polynomials over
    the rationals, compute its {\sl indefinite} integral, using if
    necessary, algebraic numbers. But in many circumstances an approximate
    result is more likely to be of use. Furthermore, it is plausible that
    it would be more useful to solve the problem to allow definite
    integration, or introduce additional parameters so that we can solve
    multiple definite integrations.  How can a computer algebra system
    best answer the more useful questions?  Finally, what if the integrand
    is not a ratio of polynomials, but something more challenging?",
  paper = "Fate08.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Fevr98,
  author = "Fevre, Stephane and Wang, Dongming",
  title = {{Proving Geometric Theorems using Clifford Algebra and Rewrite
           Rules}},
  journal = "LNCS",
  volume = "1421",
  year = "1998",
  pages = "17-32",
  abstract =
    "We consider geometric theorems that can be stated constructively by
    introducing points, while each newly introduced point may be
    represented in terms of the previously constructed points using
    Clifford algebraic operators. To prove a concrete theorem, one first
    substitutes the expressions of the dependent points into the
    conclusion Clifford polynomial to obtain an expression that involves
    only the free points and parameters. A term-rewriting system is
    developed that can simplify such an expression to 0, and thus prove
    the theorem. A large class of theorems can be proved effectively in
    this coordinate-free manner. This paper describes the method in
```

```
    detail and reports on our preliminary experiments.",
  paper = "Fevr98.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Fill98,
  author = "Filliatre, Jean-Christophe",
  title = {{Proof of Imperative Programs in Type Theory}},
  journal = "LNCS",
  volume = "1657",
  year = "1998",
  pages = "78-92",
  abstract =
    "We present a new approach to certifying functional programs with
    imperative aspects, in the context of Type Theory. The key is a
    functional translation of imperative programs, based on a combination
    of the type and effect discipline and monads. Then an incomplete proof
    of the specification is built in the Type Theory, whose gaps would
    correspond to proof obligations. On sequential imperative programs,
    we get the same proof obligations as those given by Floyd-Hoare
    logic. Compared to the latter, our approach also includes functional
    constructions in a straight-forward way. This work has been
    implemented in the Coq Proof Assistant and applied on non-trivial
    examples.",
  paper = "Fill98.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Fitc74,
  author = "Fitch, J.P.",
  title = {{CAMAL Users Manual}},
  institution = "University of Cambridge Computer Laboratory",
  year = "1974"
}
```

———-

— **axiom.bib** —

```
@misc{Flet01,
  author = "Fletcher, John P.",
  title = {{Symbolic processing of Clifford Numbers in C++}},
  year = "2001",
  journal = "Paper 25, AGACSE 2001."
```

```
}
```

———-

— **axiom.bib** —

```
@misc{Flet09,
  author = "Fletcher, John P.",
  title = {{Clifford Numbers and their inverses calculated using the matrix
           representation}},
  publisher = "Chemical Engineering and Applied Chemistry, School of
    Engineering and Applied Science, Aston University, Aston Triangle,
    Birmingham B4 7 ET, U. K.",
  link = "\url{http://www.ceac.aston.ac.uk/research/staff/jpf/papers/paper24/index.php}"
}
```

———-

— **ignore** —

```
\bibitem[Fletcher 81]{Fle81} Fletcher R
  title = {{Practical Methods of Optimization}},
Vol 2. Constrained Optimization. Wiley. (1981)
```

———-

— **axiom.bib** —

```
@article{Floy63,
  author = "Floyd, R. W.",
  title = {{Semantic Analysis and Operator Precedence}},
  journal = "JACM",
  volume = "10",
  number = "3",
  pages = "316-333",
  year = "1963"
}
```

———-

— **ignore** —

```
\bibitem[Forsythe 57]{For57} Forsythe G E,
  title = {{Generation and use of orthogonal polynomials for data
           fitting with a digital computer}},
J. Soc. Indust. Appl. Math. 5 74--88. (1957)
```

———-

— **ignore** —

```
\bibitem[Fortenbacher 90]{REF-For90} Fortenbacher, A.
``Efficient type inference and coercion in computer algebra''
Design and Implementation of Symbolic Computation Systems (DISCO 90)
A. Miola, (ed) vol 429 of Lecture Notes in Computer Science
Springer-Verlag, pp56-60
  abstract = "
    Computer algebra systems of the new generation, like Scratchpad, are
    characterized by a very rich type concept, which models the
    relationship between mathematical domains of computation. To use these
    systems interactively, however, the user should be freed of type
    information. A type inference mechanism determines the appropriate
    function to call. All known models which allow to define a semantics
    for type inference cannot express the rich ``mathematical'' type
    structure, so presently type inference is done heuristically.  The
    following paper defines a semantics for a subproblem thereof, namely
    coercion, which is based on rewrite rules. From this definition, and
    efficient coercion algorith for Scratchpad is constructed using graph
    techniques."
```

———-

— **axiom.bib** —

```
@book{Fran73,
  author = "Frankel, A.A. and Bar-Hillel, Y. and Levy, A.",
  title = {{Foundations of Set Theory}},
  publisher = "Elsevier Science",
  year = "1973",
  isbn = "978-0720422702"
}
```

———-

— **ignore** —

```
\bibitem[Fox 68]{Fox68} Fox L.; Parker I B.
  title = {{Chebyshev Polynomials in Numerical Analysis}},
Oxford University Press. (1968)
```

———-

— **ignore** —

```
\bibitem[Franke 80]{FN80} Franke R.; Nielson G
  title = {{Smooth Interpolation of Large Sets of Scattered Data}},
Internat. J. Num. Methods Engrg. 15 1691--1704. (1980)
```

———-

— **ignore** —

```
\bibitem[Fritsch 82]{Fri82} Fritsch F N
  title = {{PCHIP Final Specifications}},
Report UCID-30194. Lawrence Livermore National Laboratory. (1982)
```

———-

— **ignore** —

```
\bibitem[Fritsch 84]{FB84} Fritsch F N.; Butland J.
  title = {{A Method for Constructing Local Monotone Piecewise Cubic
           Interpolants}},
SIAM J. Sci. Statist. Comput. 5 300--304. (1984)
```

———-

— **ignore** —

```
\bibitem[Froberg 65]{Fro65} Froberg C E.
  title = {{Introduction to Numerical Analysis}},
Addison-Wesley. 181--187. (1965)
```

———-

— **axiom.bib** —

```
@misc{Fult08,
  author = "Fulton, William",
  title = {{Algebraic Curves: An Introduction to Algebraic Geometry}},
  link = "\url{http://www.math.lsa.umich.edu/~wfulton/CurveBook.pdf}",
  year = "2008",
  algebra =
    "\newline\refto{package GPAFF GeneralPackageForAlgebraicFunctionField}
\newline\refto{package PAFFFF PackageForAlgebraicFunctionFieldOverFiniteField}
     \newline\refto{package PAFF PackageForAlgebraicFunctionField}",
  paper = "Fult08.pdf"
}
```

———-

1.41.7 G

— **axiom.bib** —

```
@inproceedings{Garc16,
  author = "Garcia, Ronald and Clark, Alison M. and Tanter, Eric",
  title = {{Abstracting Gradual Typing}},
  booktitle = "POPL 16",
  publisher = "ACM",
```

```
  year = "2016",
  pages = "429-442",
  abstract =
    "Language researchers and designers have extended a wide variety
    of type systems to support {\sl gradual typing}, which enables
    languages to seemlessly combine dynamic and static checking. These
    efforts consistently demonstrate that designing a satisfactory
    gradual counterpart to a static type system is challenging, and
    this challenge only increases with the sophistication of the type
    system. Graudal type system designers need more formal tools to
    help them conceptualize, structure, and evaluate their designs.

    In this paper, we propose a new formal foundation for graudal
    typing, drawing on principles from abstract interpretation to give
    gradual types a semantics in terms of pre-existing static
    types. Abstracting Gradual Typing (AGT for short) yields a formal
    account of {\sl consistency} -- one of the cornerstones of the
    gradual typing approach -- that subsumes existing notions of
    consistency, which were developed through intuition and ad hoc
    reasoning.

    Given a syntax-directed static typing judgment, the AGT approach
    induces a corresponding gradual typing judgment. Then the type
    safety proof for the underlying static discipline induces a
    dynamic semantics for gradual programs defined over
    source-language typing derivations. The AGT approach does not
    resort to an externally justified cast calculus; instead, run-time
    check naturally arise by deducing evidence for consistent
    judgements during proof reduction.

    To illustrate the approach, we develop a novel gradually-typed
    counterpart for a language with record subtyping. Gradual
    languages designed with the AGT approach satisfy {\sl by
    construction} the refined criteria for gradual typing set forth by
    Siek and colleagues.",
  keywords = "printed"
}
```

———-

— axiom.bib —

```
@inproceedings{Garc17,
  author = "Garcia, Ronald and Cimini, Matteo",
  title = {{Principal Type Schemes for Gradual Programs}},
  booktitle = "POPL 15",
  link = "\url{https://www.cs.ubc.ca/~rxg/ptsgp.pdf}",
  comment = "Updated paper",
  year = "2017",
  abstract =
    "Gradual typing is a discipline for integrating dynamic checking
    into a static type system. Since its introduction in functional
    languages, it has been adapted to a variety of type systems,
```

```
    including object-oriented, security, and substructural. This work
    studies its application to implictly typed languages based on type
    inference. Siek and Vachharajani designed a gradual type inference
    system and algorithm that infers gradual types but still rejects
    ill-typed static programs. However, the type system requires local
    reasoning about type substitutions, an imperative inference
    algorithm, and a subtle correctness statement.

    This paper introduces a new approach to gradual type inference,
    driven by the principle that gradual inference should only produce
    static types. We present a static implicitly typed language, its
    gradual counterpart, and a type inference procedure. The gradual
    system types the same programs as Siek and Vachharajani, but has a
    modular structure amenable to extension. The language admits
    let-polymorphism, and its dynamics are defined by translation to
    the Polymorphic Blame Calculus (Ahmed et al. 2009, 2011).

    The principal types produced by our initial type system mask the
    distinction between static parametric polymorphism and
    polymorphism that can be attributed to gradual typing. To expose
    this difference, we distinguish static type parameters from
    gradual type parameters and reinterpret gradual type consistency
    accordingly. The resulting extension enables programs to be
    interpreted using either the polymorphic or monomorphic Blame
    Calculi.",
  paper = "Garc17.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Garcia 95]{Ga95} Garcia, A.; Stichtenoth, H.
  title = {{A tower of Artin-Schreier extensions of function fields
           attaining the Drinfeld-Vladut bound}},
Invent. Math., vol. 121, 1995, pp. 211--222.
```

———-

— **axiom.bib** —

```
@misc{Gari09,
  author = "Garillot, Francois and Gonthier, Georges and Mahboubi, Assia and
           Rideau, Laurence",
  title = {{Packaging Mathematical Structures}},
  year = "2009",
  abstract =
    "This paper proposes generic design patterns to define and combine
    algebraic structures, using dependent records, coercions and type
    inference, inside the Coq system. This alternative to telescopes in
    particular allows multiple inheritance, maximal sharing of notations
```

```
    and theories, and automated structure inference. Our methodology is
    robust enough to support a hierarchy comprising a broad variety of
    algebraic structures, from types with a choice operator to algebraically
    closed fields. Interfaces for the structures enjoy the handiness of a
    classical setting, without requiring any axiom. Finally, we show how
    externally extensible some of these instances are by discussing a
    lemma seminal in defining the discrete logarithm, and a matrix
    decomposition problem.",
  paper = "Gari09.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Gathen 90a]{Gat90a} {von zur Gathen}, Joachim; Giesbrecht, Mark
``Constructing Normal Bases in Finite Fields''
J. Symbolic Computation pp 547-570 (1990)
  abstract = "
    An efficient probabilistic algorithm to find a normal basis in a
    finite field is presented. It can, in fact, find an element of
    arbitrary prescribed additive order. It is based on a density estimate
    for normal elements. A similar estimate yields a probabilistic
    polynomial-time reduction from finding primitive normal elements to
    finding primitive elements.",
  paper = "Gat90a.pdf"
```

———-

— **ignore** —

```
\bibitem[Gathen 90]{Gat90} von zur Gathen, Joachim
  title = {{Functional Decomposition Polynomials: the Tame Case}},
Journal of Symbolic Computation (1990) 9, 281-299
```

———-

— **axiom.bib** —

```
@book{Gath99,
  author = {{von zur Gathen}, Joachim and Gerhard, J\"urgen},
  title = {{Modern Computer Algebra}},
  publisher = "Cambridge University Press",
  year = "1999",
  isbn = "0-521-64176-4",
  algebra = "\newline\refto{package PGCD PolynomialGcdPackage}"
}
```

———-

— **ignore** —

```
\bibitem[Gautschi 79a]{Gau79a} Gautschi W.
  title = {{A Computational Procedure for Incomplete Gamma Functions}},
ACM Trans. Math. Softw. 5 466--481. (1979)
```

————-

— **ignore** —

```
\bibitem[Gautschi 79b]{Gau79b} Gautschi W.
  title = {{Algorithm 542: Incomplete Gamma Functions}},
ACM Trans. Math. Softw. 5 482--489.  (1979)
```

————-

— **ignore** —

```
\bibitem[Gentlemen 69]{Gen69} Gentlemen W M
  title = {{An Error Analysis of Goertzel's (Watt's) Method for
           Computing Fourier Coefficients}},
Comput. J. 12 160--165. (1969)
```

————-

— **ignore** —

```
\bibitem[Gentleman 73]{Gen73} Gentleman W M.
  title = {{Least-squares Computations by Givens Transformations
           without Square Roots}},
J. Inst. Math. Applic. 12 329--336. (1973)
```

————-

— **ignore** —

```
\bibitem[Gentleman 74]{Gen74} Gentleman W M.
  title = {{Algorithm AS 75. Basic Procedures for Large Sparse or
           Weighted Linear Least-squares Problems}},
Appl. Statist. 23 448--454. (1974)
```

————-

— **ignore** —

```
\bibitem[Gentlemen 74a]{GM74a} Gentleman W. M.; Marovich S. B.
  title = {{More on algorithms  that reveal properties of floating
           point arithmetic units}},
Comms. of the ACM, 17, 276-277.  (1974)
```

———-

— **ignore** —

```
\bibitem[Genz 80]{GM80} Genz A C.;  Malik A A.
  title = {{An Adaptive Algorithm for Numerical Integration over an
            N-dimensional Rectangular Region}},
J. Comput. Appl. Math. 6 295--302.  (1980)
```

———-

— **axiom.bib** —

```
@article{Ghel91,
  author = "Ghelli, Giorgio",
  title = {{Modeling Features of Object-Oriented Languages in Second
            Order Functional Languages with Subtypes}},
  journal = "LNCS",
  volume = "489",
  pages = "311-340",
  year = "1991",
  abstract =
    "Object oriented languages are an important tool to achieve software
    reusability in any kind of application and can increase dramatically
    software productivity in some special fields; they are also considered
    a logical step in the evolution of object oriented languages.  But
    these languages lack a formal foundation, which is needed both to
    develop tools and as a basis for the future evolution of these
    languages; they lack also a strong type system, which would be
    essential to assure that level of reliability which is required to
    large evolving systems.  Recently some researches have tried to
    insulate the basic mechanisms of object oriented languages and to
    embed them in strongly typed functional languages, giving to these
    mechanism a mathematical semantics and a set of strong type
    rules. This is a very promising approach which also allows a
    converging evolution of both the typed functional and the object
    oriented programming paradigms, making it possible a technology
    transfer in both directions.  Most works in this field are very
    technical and deal just with one aspect of object oriented
    programming; many of them use a very similar framework.  In this work
    we describe and exemplify that common framework and we survey some of
    the more recent and promising works on more specific features, using
    the framework introduced.  We describe the resuks achieved and point
    out some problems which are still open, especially those arising from
    the interactions between different mechanisms.",
  paper = "Ghel91.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gies05,
  author = "Giese, Martin",
  title = {{A Calculus for Type Predicates and Type Coercion}},
  booktitle = "A Calculus for Type Predicates and Type Coercion",
  series = "LNAI",
  volume = "3702",
  pages = "123-137",
  isbn = "3540289313",
  year = "1985",
  abstract =
    "We extend classical first-order logic with subtyping by type
    predicates and type coercion. Type predicates assert that the value of
    a term belongs to a more special type than the signature guarantees,
    while type coercion allows using terms of a more general type where
    the signature calls for a more special one. These operations are
    important e.g. in the specification and verification of
    object-oriented programs. We present a tableau calculus for this logic
    and prove its completeness.",
  paper = "Gies05.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Gill 72]{GM72} Gill P E.; Miller G F.
  title = {{An Algorithm for the Integration of Unequally Spaced Data}},
Comput. J. 15 80--83. (1972)
```

———-

— **ignore** —

```
\bibitem[Gill 74b]{GM74b} Gill P E.; Murray W. (eds)
  title = {{Numerical Methods for Constrained Optimization}},
Academic Press. (1974)
```

———-

— **ignore** —

```
\bibitem[Gill 76a]{GM76a} Gill P E.; Murray W.
  title = {{Minimization subject to bounds on the variables}},
Report NAC 72. National Physical Laboratory. (1976)
```

———-

— **ignore** —

```
\bibitem[Gill 76b]{GM76b} Gill P E.; Murray W.
  title = {{Algorithms for the Solution of the Nonlinear
```

```
           Least-squares Problem}},
NAC 71 National Physical Laboratory. (1976)
```

———-

— **ignore** —

```
\bibitem[Gill 78]{GM78} Gill P E.; Murray W.
  title = {{Algorithms for the Solution of the Nonlinear
           Least-squares Problem}},
SIAM J. Numer. Anal. 15 977--992. (1978)
```

———-

— **ignore** —

```
\bibitem[Gill 79]{GM79} Gill P E.;  Murray W;
  title = {{Conjugate-gradient Methods for Large-scale Nonlinear
           Optimization}},
Technical Report SOL 79-15. Department of Operations Research,
Stanford University. (1979)
```

———-

— **ignore** —

```
\bibitem[Gill 81]{GMW81} Gill P E.; Murray W.; Wright M H.
  title = {{Practical Optimization}},
Academic Press. 1981
```

———-

— **ignore** —

```
\bibitem[Gill 82]{GMW82} Gill P E.; Murray W.; Saunders M A.; Wright M H.
  title = {{The design and implementation of a quadratic programming
           algorithm}},
Report SOL 82-7. Department of Operations Research,
Stanford University. (1982)
```

———-

— **ignore** —

```
\bibitem[Gill 84a]{GMSW84a} Gill P E.; Murray W.; Saunders M A.; Wright M H
  title = {{User's Guide for SOL/QPSOL Version 3.2}},
Report SOL 84-5. Department of Operations Research, Stanford University. 1984
```

———-

— **ignore** —

```
\bibitem[Gill 84b]{GMSW84b} Gill P E.; Murray W.; Saunders M A.; Wright M H
  title = {{Procedures for Optimization Problems with a Mixture of
            Bounds and General Linear Constraints}},
ACM Trans. Math. Softw. 10 282--298. 1984
```

————-

— **ignore** —

```
\bibitem[Gill 86a]{GMSW86a} Gill P E.; Hammarling S.; Murray W.;
Saunders M A.; Wright M H.
  title = {{User's Guide for LSSOL (Version 1.0)}},
Report SOL 86-1. Department of Operations Research, Stanford University. 1986
```

————-

— **ignore** —

```
\bibitem[Gill 86b]{GMSW86b} Gill P E.; Murray W.; Saunders M A.; Wright M H.
  title = {{Some Theoretical Properties of an Augmented Lagrangian
            Merit Function}},
Report SOL 86-6R. Department of Operations Research, Stanford University. 1986
```

————-

— **axiom.bib** —

```
@book{Gira03,
  author = "Girard, Jean-Yves and Taylor, Paul and Lafont, Yves",
  title = {{Proofs and Types}},
  publisher = "Cambridge University Press",
  link = "\url{http://www.paultaylor.eu/stable/prot.pdf}",
  year = "2003",
  paper = "Gira03.pdf",
  keywords = "printed"
}
```

————-

— **ignore** —

```
\bibitem[Gladwell 79]{Gla79} Gladwell I
  title = {{Initial Value Routines in the NAG Library}},
ACM Trans Math Softw. 5 386--400. (1979)
```

————-

— **ignore** —

```
\bibitem[Gladwell 80]{GS80} Gladwell I.; Sayers D K
  title = {{Computational Techniques for Ordinary Differential Equations}},
Academic Press. 1980
```

———-

— **ignore** —

```
\bibitem[Gladwell 86]{Gla86} Gladwell I
  title = {{Vectorisation of one dimensional quadrature codes}},
Techincal Report. TR7/86  NAG. (1986)
```

———-

— **ignore** —

```
\bibitem[Gladwell 87]{Gla87} Gladwell I
  title = {{The NAG Library Boundary Value Codes}},
Numerical Analysis Report. 134 Manchester University. (1987)
```

———-

— **ignore** —

```
\bibitem[Goedel 40]{God40} Goedel
  title = {{The consistency of the continuum hypothesis}},
Ann. Math. Studies, Princeton Univ. Press, 1940
```

———-

— **axiom.bib** —

```
@misc{Gode16,
  author = "Godek, Panicz Maciej",
  title = {{A Pamphlet Against R}},
  month = "February 6",
  year = "2016",
  link = "\url{http://github.com/panicz/pamphlet/raw/master/pamphlet.pdf}",
  abstract =
   "R is a programming language that has been gaining popularity in the
    domains such as machine learning and artificial intelligence over the
    recent years. It has been praised for its simplicity and elasticity,
    and its creators were even assigned a rockstar status.

    In my pamphlet, I claim that this ‘‘blazing new technology’’ is
    actually a step backwards in the development of the domains that it
    tries to tackle, and the better means of expression, or better tools
    for doing the job, were available at least since the 70s, and recently
    they were only getting better.
```

```
    This pamphlet explores and expresses various {\sl computational
    intelligence methods} using {\sl Guile}, a very pleasant
    implementation of the {\sl Scheme} programming language.",
  paper = "Gode16.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Goldman 87]{Gold87} Goldman, L.
  title = {{Integrals of multinomial systems of ordinary differential
           equations}},
J. of Pure and Applied Algebra, 45, 225-240 (1987)
  link = "\url{http://www.sciencedirect.com/science/article/pii/0022404987900727/pdf?md5=5a0c70643eab514ccf47d80
  paper = "Gold87.pdf"
```

———-

— **axiom.bib** —

```
@techreport{Gont09,
  author = "Gonthier, Georges and Mahboubi, Assia and Tassi, Enrico",
  title = {{A Small Scale Reflection Extension for the Coq System}},
  type = "technical report",
  institution = "Inria Microsoft",
  number = "RR-6455",
  year = "2009",
  abstract =
    "This is the user manual de SSReflect , a set of extensions to the
    proof scripting language of the Coq proof assistant. While these
    extensions were developed to support a particular proof methodology -
    small-scale reflection - most of them actually are of a quite general
    nature, improving the functionality of Coq in basic areas such as
    script layout and structuring, proof context management, and
    rewriting.  Consequently, and in spite of the title of this document,
    most of the extensions described here should be of interest for all
    Coq users, whether they embrace small-scale reflection or not.",
  paper = "Gont09.pdf"

}
```

———-

— **axiom.bib** —

```
@misc{Grab91a,
  author = "Grabmeier, Johannes",
  title = {{Groups, finite fields and algebras, constructions and
```

```
            calculations}},
  location = "IBM Europe Institute",
  year = "1991"
}
```

———-

— **ignore** —

```
\bibitem[Grabmeier]{Grab} Grabmeier, J.
  title = {{On Plesken's root finding algorithm}},
in preparation
```

———-

— **ignore** —

```
\bibitem[Grebmeier 87]{GK87} Grabmeier, J.; Kerber, A.;
  title = {{The Evaluation of Irreducible Polynomial Representations of
            the General Linear Groups and of the Unitary Groups over
            Fields of Characteristic 0}},
Acta Appl. Math. 8 (1987), 271-291
```

———-

— **ignore** —

```
\bibitem[Grabmeier 92]{REF-GS92} Grabmeier, J.; Scheerhorn, A.
  title = {{Finite fields in Axiom}},
AXIOM Technical Report TR7/92 (ATR/5)(NP2522),
Numerical Algorithms Group, Inc., Downer's
Grove, IL, USA and Oxford, UK, 1992
  link = "\url{http://www.nag.co.uk/doc/TechRep/axiomtr.html}",
```

———-

— **ignore** —

```
\bibitem[Granville 1911]{Gran1911} Granville, William Anthony
  title = {{Elements of the Differential and Integral Calculus}},
  link = "\url{http://djm.cc/library/Elements_Differential_Integral_Calculus_Granville_edited_2.pdf}",
  paper = "Gran1911.pdf"
```

———-

— **axiom.bib** —

```
@article{Gray98,
  author = "Gray, Simon and Kajler, Norbert and Wang, Paul S.",
  title = {{Design and Implementation of MP, a Protocol for
```

```
            Efficient Exchange of Mathematical Expressions}},
  journal = "J. Symboic Computation",
  volume = "25",
  pages = "213-237",
  year = "1998",
  abstract =
    "The Multi Project is an ongoing research effort at Kent State
    University aimed at providing an environment for distributed scientific
    computing. An integral part of this environment is the Multi-
    Protocol (MP) which is designed to support scientific communication of
    mathematical data between scientifically-oriented software tools. MP
    exchanges data in the form of linearized annotated syntax
    trees. Syntax trees provide a simple, flexible and tool-independent
    way to represent and exchange data, and annotations provide a powerful
    and generic expressive facility for transmitting additional
    information.  At a level above the data exchange protocol,
    dictionaries provide definitions for operators and constants, providing
    shared semantics across heterogeneous packages. A clear distinction
    between MP-defined and user-defined entities is enforced. Binary
    encodings are used for efficiency. Commonly used values and blocks of
    homogeneous data are further optimized. The protocol is independent of
    the underlying communication paradigm and can support parallel
    computation, distributed problem-solving environments, and the
    coupling of tools for specific applications.",
  paper = "Gray98.pdf",
  keywords = "printed, axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Grie11,
  author = "Griesmer, James",
  title = {{James Griesmer 1929--2011}},
  journal = "ACM Communications in Computer Algebra",
  volume = "46",
  number = "1/2",
  year = "2012",
  pages = "10-11",
  comment = "Obituary",
  paper = "Grie11.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Gris76,
  author = "Griss, Martin L.",
  title = {{The Definition and Use of Data Structures in REDUCE}},
  booktitle = "SYMSAC '76",
```

```
  pages = "53-59",
  year = "1976",
  abstract =
    "This paper gives a brief description and motivation of the mode
    analyzing and data-structuring extensions to the algebraic language
    REDUCE. These include generic functions, user defined recursive data
    structures, mode transfer functions and user modifiable automatic
    coercion. A number of examples are given to illustrate the style and
    features of the language, and how it will aid in the construction of
    more efficient and reliable programs."
}
```

———-

— **axiom.bib** —

```
@misc{Grog91,
  author = "Grogono, Peter",
  title = {{Issues in the Design of an Object Oriented Programming
            Language}},
  link = "\url{http://users.encs.concordia.ca/~grogono/Writings/oopissue.pdf}",
  year = "1991",
  comment = "Published in Structured Programming",
  abstract =
    "The object oriented paradigm, which advocates bottom-up program
    development, appears at first sight to run counter to the classical,
    top-down approach of structured programming.  The deep requirement of
    structured programming, however, is that programming should be based
    on well-defined abstractions with clear meaning rather than on
    incidental characteristics of computing machinery. This requirement
    can be met by object oriented programming and, in fact, object
    oriented programs may have better structure than programs obtained by
    functional decomposition.

    The definitions of the basic components of object oriented
    programming, object, class, and inheritance, are still sufficiently
    fluid to provide many choices for the designer of an object oriented
    language. Full support of objects in a typed language requires a
    number of features, including classes, inheritance, genericity,
    renaming, and redefinition. Although each of these features is simple
    in itself, interactions between features can become complex. For
    example, renaming and redefinition may interact in unexpected ways.
    In this paper, we show that appropriate design choices can yield a
    language that fulfills the promise of object oriented programming
    without sacrificing the requirements of structured programming.",
  paper = "Grog91.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Gruntz 93]{Gru93} Gruntz, Dominik
  title = {{Limit computation in computer algebra}},
  link = "\url{http://algo.inria.fr/seminars/sem92-93/gruntz.pdf}",
  abstract = "
    The automatic computation of limits can be reduced to two main
    sub-problems. The first one is asymptotic comparison where one must
    decide automatically which one of two functions in a specified class
    dominates the other one asymptotically. The second one is asymptotic
    cancellation and is usually exemplified by
    \[e^x[exp(1/x+e^{-x})-exp(1/x)],\quad{}x \leftarrow \infty\]

    In this example, if the sum is expanded in powers of $1/x$, the
    expansion always yields $O(x^{-k})$, and this is not enough to
    conclude.

    In 1990, J.Shackell found an algorithm that solved both these problems
    for the case of $exp-log$ functions, i.e. functions build by recursive
    application of exponential, logarithm, algebraic extension and field
    operations to one variable and the rational numbers. D. Gruntz and
    G. Gonnet propose a slightly different algorithm for exp-log
    functions.  Extensions to larger classes of functions are also
    discussed.",
  paper = "Gru93.pdf"
```

———-

— **axiom.bib** —

```
@article{Gues87,
  author = "Guessarian, Irene and Meseguer, Jose",
  title = {{On the Axiomatization of ``IF-THEN-ELSE''}},
  journal = "SIAM J. Comput.",
  volume = "16",
  numbrrer = "2",
  year = "1987",
  pages = "332-357",
  abstract =
    "The equationally complete proof system for ``if-then-else'' of Bloom
    and Tindell is extended to a complete proof system for many-osrted
    algebras with extra operations, predicates and equations among
    those. We give similar completeness results for continuous algebras
    and program schemes (infinite trees) by the methods of algebraic
    semantics. These extensions provide a purely equational proof system
    to prove properties of functional programs over user-definable data
    types.",
  paper = "Gues87.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@techreport{Gutt91,
  author = "Guttman, J.D.",
  title = {{A Propoed Interface Logic for Verification Environments}},
  type = "technical report",
  number = "M91-19",
  institution = "The MITRE Corporation",
  year = "1991"
}
```

———-

1.41.8 H

— **axiom.bib** —

```
@article{Hach95,
  author = "Hach\'e, G. and Le Brigand, D.",
  title = {{Effective construction of algebraic geometry codes}},
  journal = "IEEE Transaction on Information Theory",
  volume = "41",
  number = "6",
  month = "November",
  year = "1995",
  pages = "1615--1628",
  link = "\url{https://hal.inria.fr/inria-00074404/file/RR-2267.pdf}",
  algebra =
    "\newline\refto{package GPAFF GeneralPackageForAlgebraicFunctionField}
\newline\refto{package PAFFFF PackageForAlgebraicFunctionFieldOverFiniteField}
     \newline\refto{package PAFF PackageForAlgebraicFunctionField}",
  abstract =
    "We intend to show that algebraic geometry codes (AG-codes, introduced
    by Goppa in 1977 [5]) can be constructed easily using blowing-up
    theory. This work is based on a paper by Le Brigand and Risler. Given
    a plane curve, we desingularize the curve by means of blowing-up, and
    then using the desingularisation trees and the monoidal
    transformations associated to the blowing-up morphisms, we compute the
    adjoint divisor of the curve. Finally we show how to use the algorithm
    of Brill-Noether to compute a basis of the vector space associated to
    a divisor of the curve. Two examples of constructions of AG-codes are
    given at the end.",
  paper = "Hach95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Hach95a,
  author = "Hach\'e, G.",
  title = {{Computation in algebraic function fields for effective
```

```
           construction of algebraic-geometric codes}},
  journal = "Lecture Notes in Computer Science",
  volume = "948",
  year = "1995",
  pages = "262--278"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Hach96,
  author = "Hach\'e, G.",
  title = {{Construction effective des codes g\'eom\'etriques}},
  school = "l'Universit\'e Pierre et Marie Curie (Paris 6)",
  year = "1996",
  month = "Septembre"
}
```

———-

— **axiom.bib** —

```
@article{Haft06,
  author = "Haftmann, Florian and Wenzel, Makarius",
  title = {{Constructive Type Classes in Isabelle}},
  journal = "LNCS",
  volume = "4502",
  year = "2006",
  abstract =
    "We reconsider the well-known concept of Haskell-style type classes
    within the logical framework of Isabelle. So far, axiomatic type
    classes in Isabelle merely account for the logical aspect as
    predicates over types, while the operational part is only a
    convention based on raw overloading. Our more elaborate approach to
    constructive type classes provides a seamless integration with
    Isabelle locales, which are able to manage both operations and logical
    properties uniformly. Thus we combine the convenience of type
    classes and the flexibility of locales. Furthermore, we construct
    dictionary terms derived from notions of the type system. This
    additional internal structure provides satisfactory foundations of
    type classes, and supports further applications, such as code
    generation and export of theories and theorems to environments without
    type classes.",
  paper = "Haft06.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Hall17,
  author = "Halleck, John",
  title = {{Logic System Interrelationships}},
  year = "2017",
  link = "\url{http://www.cc.utah.edu/~nahaj/logic/structures/index.html}"
}
```

———-

— **ignore** —

```
\bibitem[Hall 76]{HW76} Hall G.; Watt J M. (eds),
  title = {{Modern Numerical Methods for Ordinary Differential Equations}},
Clarendon Press. (1976)
```

———-

— **ignore** —

```
\bibitem[Hamdy 04]{Ham04} Hamdy, S.
  title = {{LiDIA A library for computational number theory}},
Reference manual Edition 2.1.1 May 2004
  link = "\url{http://www.cdc.informatik.tu-darmstadt.de/TI/LiDIA}",
```

———-

— **ignore** —

```
\bibitem[Hammersley 67]{HH67} Hammersley J M; Handscomb D C.
‘‘Monte-Carlo Methods’’
Methuen. (1967)
```

———-

— **axiom.bib** —

```
@inproceedings{Haml02,
  author = "Hamlet, Dick",
  title = {{Continuity in Software Systems}},
  booktitle = "ISSTA 2002 Int. Symp. on Software Testing and Analysis",
  pages = "196-200",
  year = "2002",
}
```

———-

— **axiom.bib** —

```
@article{Harr06,
  author = "Harrison, John",
```

```
  title = {{Proof Style}},
  journal = "LNCS",
  volume = "1512",
  year = "2006",
  pages = "154-172",
  abstract =
    "We are concerned with how computer theorem provers should expect
    users to communicate proofs to them.  There are many stylistic choices
    that still allow the machine to generate a completely formal proof
    object.  The most obvious choice is the amount of guidance required
    from the user, or from the machine perspective, the degree of
    automation provided.  But another important consideration, which we
    consider particularly significant, is the bias towards a
    'procedural' or 'declarative' proof style.  We will explore this
    choice in depth, and discuss the strengths and weaknesses of
    declarative and procedural styles for proofs in pure mathematics and
    for verification applications.  We conclude with a brief summary of
    our own experiments in trying to combine both approaches.",
  paper = "Harr06.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Harr07,
  author = "Harrison, John",
  title = {{A Short Survey of Automated Reasoning}},
  booktitle = "Proc. 2nd Int. Conf. on Algebraic Biology",
  pages = "334-349",
  year = "2007",
  publisher = "Springer-Verlag",
  isbn = "978-3-540-73432-1",
  abstract =
    "This paper surveys the field of automated reasoning, giving some
    historical background and outlining a few of the main current research
    themes. We particularly emphasize the points of contact and the
    contrasts with computer algebra. We finish with a discussion of the
    main applications so far.",
  paper = "Harr07.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Hath1896,
  author = "Hathway, Arthur S.",
  title = {{A Primer Of Quaternions}},
  year = "1896"
}
```

———-

— **axiom.bib** —

```
@book{Haya05,
  author = "Hayashi, K. and Kangkook, J. and Lascu, O. and Pienaar, H. and
            Schreitmueller, S. and Tarquinio, T. and Thompson, J.",
  title = {{AIX 5L Practical Performance Tools and Tuning Guide}},
  publisher = "IBM",
  year = "2005",
  link = "\url{http://www.redbooks.ibm.com/redbooks/pdfs/sg246478.pdf}",
  paper = "Haya05.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Hayes 70]{Hay70} Hayes J G.
  title = {{Curve Fitting by Polynomials in One Variable}},
Numerical Approximation to Functions and Data.
(ed J G Hayes) Athlone Press, London. (1970)
```

———-

— **ignore** —

```
\bibitem[Hayes 74]{Hay74} Hayes J G.
  title = {{Numerical Methods for Curve and Surface Fitting}},
Bull Inst Math Appl. 10 144--152. (1974)
```

———-

— **ignore** —

```
\bibitem[Hayes 74a]{HH74} Hayes J G.; Halliday J,
  title = {{The Least-squares Fitting of Cubic Spline Surfaces to
            General Data Sets}},
J. Inst. Math. Appl. 14 89--103. (1974)
```

———-

— **axiom.bib** —

```
@inproceedings{Haye74,
  author = "Hayes, Patrick J.",
  title = {{Some Problems and Non-problems in Representation Theory}},
  booktitle = "AI and Simulation of Behaviour Conference",
  year = "1974",
```

```
  pages = "63-79"
}
```

———-

— **axiom.bib** —

```
@misc{Hear87,
  author = "Hearn, Anthony",
  title = {{REDUCE User's Manual}},
  version = "3.3",
  institution = "Rand Corporation",
  year = "1987"
}
```

———-

— **axiom.bib** —

```
@misc{Hebi10,
  author = "Hebisch, Waldemar and Rubey, Martin",
  title = {{Extended Rate, more GFUN}},
  year = "2010",
  link = "\url{https://arxiv.org/abs/math/0702086v2}",
  algebra =
    "\newline\refto{package GUESS Guess}
     \newline\refto{package GUESSAN GuessAlgebraicNumber}
     \newline\refto{package GUESSF GuessFinite}
     \newline\refto{package GUESSF1 GuessFiniteFunctions}
     \newline\refto{package GUESSINT GuessInteger}
     \newline\refto{package GUESSP GuessPolynomial}
     \newline\refto{package GUESSUP GuessUnivariatePolynomial}",
  abstract =
    "We present a software package that guesses formulae for sequences of,
    for example, rational numbers or rational functions, given the first
    few terms. We implement an algorithm due to Bernhard Beckermann and
    George Labahn, together with some enhancements to render our package
    efficient. Thus we extend and complement Christian Krattenthaler's
    program Rate, the parts concerned with guessing of Bruno Salvy and
    Paul Zimmermann's GFUN, the univariate case of Manuel Kauers' Guess.m
    and Manuel Kauers' and Christoph Koutschan's qGeneratingFunctions.m.",
  paper = "Hebi10.pdf",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Henrici 56]{Hen56} Henrici, Peter
  title = {{Automatic Computations with Power Series}},
```

```
Journal of the Association for Computing Machinery, Volume 3, No. 1,
January 1956, 10-15
```

———-

— **ignore** —

```
\bibitem[Hock 81]{HS81} Hock W.; Schittkowski K.
  title = {{Test Examples for Nonlinear Programming Codes}},
Lecture Notes in Economics and Mathematical Systems. 187 Springer-Verlag. 1981
```

———-

— **axiom.bib** —

```
@article{Homa94,
  author = "Homann, Karsten and Calmet, Jacques",
  title = {{Combining Theorem Proving and Symbolic Mathematical Computing}},
  journal = "LNCS",
  volume = "958",
  pages = "18-29",
  year = "1994",
  abstract =
    "An intelligent mathematical environment must enable symbolic
    mathematical computation and sophisticated reasoning techniques on the
    underlying mathematical laws. This paper disscusses different possible
    levels of interaction between a symbolic calculator based on algebraic
    algorithms and a theorem prover. A high level of interaction requires
    a common knowledge representation of the mathematical knowledge of the
    two systems. We describe a model for such a knowledge base mainly
    consisting of type and algorithm schemata, algebraic algorithms and
    theorems.",
  paper = "Homa94.pdf",
  keywords = "axiomref, CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@article{Homa96,
  author = "Homann, Karsten and Calmet, Jacques",
  title = {{Structures for Symbolic Mathematical Reasoning and Computation}},
  journal =  "LNCS",
  volume = "1128",
  year = "1996",
  pages = "216-227",
  abstract =
    "Recent research towards integrating symbolic mathematical reasoning
    and computation has led to prototypes of interfaces and
    environments. This paper introduces computation theories and
```

```
    structures to represent mathematical objects and applications of
    algorithms occuring in algorithmic services. The composition of
    reasoning and computation theories and structures provide a formal
    framework for the specification of symbolic mathematical problem
    solving by cooperation of algorithms and theorems.",
  paper = "Homa96.pdf",
  keywords = "CAS-Proof, printed"
}
```

———-

— **ignore** —

```
\bibitem[Householder 70]{Hou70} Householder A S.
  title = {{The Numerical Treatment of a Single Nonlinear Equation}},
McGraw-Hill. (1970)
```

———-

— **axiom.bib** —

```
@book{Hous81,
  author = "Householder, Alston S.",
  title = {{Principles of Numerical Analysis}},
  publisher = "Dover Publications, Mineola, NY",
  year = "1981",
  isbn = "0-486-45312-X"
}
```

———-

— **ignore** —

```
\bibitem[Huang 96]{HI96} Huang, M.D.; Ierardi, D.
  title = {{Efficient algorithms for Riemann-Roch problem and for addition
            in the jacobian of a curve}},
Proceedings 32nd Annual Symposium on Foundations of Computer Sciences.
IEEE Comput. Soc. Press, pp. 678--687.
```

———-

— **axiom.bib** —

```
@misc{Hubb,
  author = "Hubbard, John H. and Lundell, Benjamin",
  title = {{A First Look at Differential Algebra}},
  link = "\url{http://www.math.cornell.edu/~hubbard/diffalg1.pdf}",
  algebra = "\newline\refto{category DVARCAT DifferentialVariableCategory}",
  abstract =
    "The object of the paper is to prove that the differential equation
    \[u^{'}(t)=t-[u(t)]^2\]
```

```
    has no solutions which can be written using elementary functions, or
    anti-derivatives of elementary functions, or exponentials of such
    anti-derivatives, or anti-derivative of those, etc. We should note
    that Equation 1 can be solved using power series, integrals which
    depend on a parameter, or Bessel functions of order $1/3$.  However,
    as we will see, none of these methods of solution are ‘‘algebraic’’ in
    nature.

    We aim to give a precise definition of ‘‘algebraic’’ by developing the
    theory of {\sl differential algebra}, which is largely the work of
    Ritt. Other contributors are Liouville, Picard, Vessoit, Kolchin,
    Rosenlicht, ... The part of differential Galois theory which leads to
    a proof of Abel's celebrated result that a general polynomial equation
    of degree five or higher cannot be solved by radicals. In effort to
    derive these two areas in parallel, we will also explain why the
    polynomial equation
    \[x^5-4x^2-2=0\]
    has no solutions which can be written as radicals of solutions to
    lower degree polyomial equations.",
  paper = "Hubb.pdf"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Hube03,
  author = "Hubert, Evelyne",
  title = {{Notes on Triangular Sets and Triangulation-Decomposition I:
           Polynomial Systems}},
  booktitle = "Symbolic and Numerical Scientific Computing",
  series = "Lecture Notes in Computer Science 2630",
  year = "2003",
  pages = "1-39",
  link = "\url{http://www-sop.inria.fr/members/Evelyne.Hubert/publications/sncsp.pdf}",
  abstract =
    "This is the first in a series of two tutorial articles devoted to
    triangulation-decomposition algorithms. The value of these notes
    resides in the uniform presentation of triangulation-decomposition
    of polynomial and differential radical ideals with detailed proofs of
    all the presented results.We emphasize the study of the mathematical
    objects manipulated by the algorithms and show their properties in
    independently of those. We also detail a selection of algorithms, one
    for each task.  We address here polynomial systems and some of the
    material we develop here will be used in the second part, devoted to
    differential systems.",
  paper = "Hube03.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@InProceedings{Hube03a,
  author = "Hubert, Evelyne",
  title = {{Notes on Triangular Sets and Triangulation-Decomposition II:
           Differential Systems}},
  booktitle = "Symbolic and Numerical Scientific Computing",
  series = "Lecture Notes in Computer Science 2630",
  year = "2003",
  pages = "40-87",
  link = "\url{http://www-sop.inria.fr/members/Evelyne.Hubert/publications/sncsd.pdf}",
  abstract =
    "This is the second in a series of two tutorial articles devoted to
    triangulation-decomposition algorithms. The value of these notes
    resides in the uniform presentation of triangulation-decomposition of
    polynomial and differential radical ideals with detailed proofs of all
    the presented results.We emphasize the study of the mathematical
    objects manipulated by the algorithms and show their properties
    independently of those. We also detail a selection of algorithms, one
    for each task. The present article deals with differential systems. It
    uses results presented in the first article on polynomial systems but
    can be read independently.",
  paper = "Hube03a.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{Hulz82,
  author = "van Hulzen, J.A.",
  title = {{Computer Algebra Systems Viewed by a Notorious User}},
  journal = "LNCS",
  volume = "144",
  pages = "166-180",
  year = "1982",
  abstract =
    "Are design and use of computer algebra systems disjoint or
    complementary activities? Raising and answering this question are
    equally controversial, since a clear distinction between languages
    features and library facilities is hard to make.  Instead of even
    attempting to answer this rather academic question it is argued why it
    is reasonable to raise related questions: Is SMP a paradox? Is it
    realistic to neglect inaccurate input data? Is a very high level
    programming language instrumental for equal opportunity employment in
    scientific research?",
  paper = "Hulz82.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@inbook{Hulz82a,
  author = "Hulzen, J.A. van and Calmet, J.",
  title = {{Computer Algebra Systems}},
  booktitle = {{Computer Algebra: Symbolic and Algebraic Computation}},
  publisher = "Springer",
  year = "1982",
  isbn = "978-3-211-81684-4",
  pages = "221-244",
  abstract =
    "A survey is given of computer algebra systems, with emphasis on
    design and implementation aspects, by presenting a review of the
    development of ideas and methods in a historical perspective, by
    us considered as instrumental for a better understanding of the
    rich diversity of now available facilities. We first indicate
    which classes of mathematical expressions can be stated and
    manipulated in different systms before we touch on different
    general aspects of usage, design and implementation, such as
    language design, encoding, dynamic storage allocation and a
    symbolic-numeric interface. Then we discuss polynomial and
    rational function systems, by describing ALTRAN and SAC-2. This is
    followed by a comparison of some of the features of MATHLAB-68,
    SYMBAL and FORMAC, which are pretended general purpose
    systems. Before considering giants (MACSYMA and SCRATCHPAD) and
    gnomes (muMATH-79), we give the main characteristics of TRIGMAN,
    CAMAL and REDUCE, systems we tend to consider as grown out special
    purpose facilities. Finally we mention some modern algebra systems
    (CAYLEY and CAMAC-79) in relation to recent proposals for a
    language for computational algebra. We conclude by stipulating the
    importance of documentation. Throughout this discussion related
    systems and facilities will be mentioned. Noticeable are ALKAHEST
    II, ALPAK, ANALITIK, ASHMEDAI, NETFORM, PM, SAC-1, SCHOONSCHIP,
    SHEEP, SML, SYCOPHANTE and TAYLOR.",
  paper = "Buch82.pdf",
  keywords = "axiomref"
}
```

———-

1.41.9 I

— **axiom.bib** —

```
@misc{IBMx91,
  author = "Computer Algebra Group",
  title = {{The AXIOM Users Guide}},
  publisher = "NAG Ltd., Oxford",
  year = "1991"
}
```

———-

— **ignore** —

```
\bibitem[IBM]{IBM}.
SCRIPT Mathematical Formula Formatter User's Guide, SH20-6453,
IBM Corporation, Publishing Systems Information Development,
Dept. G68, P.O. Box 1900, Boulder, Colorado, USA 80301-9191.
```

———-

— **axiom.bib** —

```
@article{Ioak94,
  author = "Ioakimidis, N.I.",
  title = {{Symbolic computations for the solution of inverse/design
           problems with Maple}},
  journal = "Comput. Struct.",
  volume = "53",
  number = "1",
  pages = "63-68",
  year = "1994",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Iyan60,
  author = "Iyanaga, Shokichi and Iyanaga, Yukiyosi Kawada",
  title = {{Encyclopedia Dictionary of Mathematics}},
  publisher = "Mathematical Society of Japan",
  year = "1960",
  isbn = "978-0262090261",
  algebra = "\newline\refto{category GRALG GradedAlgebra}"
}
```

———-

1.41.10 J

— **axiom.bib** —

```
@article{Jaco51,
  author = "Jacobson, Nathan",
  title = {{General Representation Theory of Jordan Algebras}},
  journal = "Trans. of the American Mathematical Society",
  volume = "70",
  number = "3",
```

```
  year = "1951",
  pages = "509-530",
  link = "\url{http://www.math.uci.edu/~brusso/jacobson1951.pdf}",
  algebra = "\newline\refto{category MONAD Monad}",
  paper = "Jaco51.pdf"
}
```

———-

,

— **axiom.bib** —

```
@article{Jaco68,
  author = "Jacobson, Nathan",
  title = {{Structure and Representations of Jordan Algebras}},
  journal = "Bull. Amer. Math. Soc",
  volume = "79",
  number = "3",
  year = "1973",
  pages = "509-514",
  link = "\url{http://projecteuclid.org/euclid.bams/1183534656}",
  algebra =
    "\newline\refto{category MONAD Monad},
     \newline\refto{category MONADWU MonadWithUnit}"
}
```

———-

— **axiom.bib** —

```
@misc{Jaes91,
  author = "Jaeschke, G.",
  title = {{Private Communication}},
  comment = "to James Davenport",
  year = "1991"
}
```

———-

— **ignore** —

```
\bibitem[James 81]{JK81} James, Gordon; Kerber, Adalbert
  title = {{The Representation Theory of the Symmetric Group}},
Encyclopedia of Mathematics and its Applications Vol. 16
Addison-Wesley, 1981
```

———-

— **ignore** —

```
\bibitem[Jaswon 77]{JS77} Jaswon, M A.; Symm G T.
  title = {{Integral Equation Methods in Potential Theory and Elastostatics}},
```

```
Academic Press. (1977)
```

———-

— **ignore** —

```
\bibitem[Jeffrey 04]{Je04} Jeffrey, Alan
  title = {{Handbook of Mathematical Formulas and Integrals}},
Third Edition, Elsevier Academic Press ISBN 0-12-382256-4
```

———-

— **ignore** —

```
\bibitem[Jenning 66]{Jen66} Jennings A
  title = {{A Compact Storage Scheme for the Solution of Symmetric
            Linear Simultaneous Equations}},
Comput. J. 9 281--285. (1966)
```

———-

— **axiom.bib** —

```
@techreport{Jenk86c,
  author = "Jenks, Richard D.",
  title = {{A History of the SCRATCHPAD Project (1977-1986)}},
  institution = "IBM Research",
  year = "1986",
  month = "May",
  type = "Scratchpad II Newsletter",
  volume = "1",
  number = "3",
}
```

———-

— **axiom.bib** —

```
@article{Jone96a,
  author = "Jones, Alex and Luo, Zhaohui and Soloviev, Sergei",
  title = {{Some Algorithmic and Proof-Theoretical Aspects of
            Coercive Subtyping}},
  journal = "LNCS",
  volume = "1512",
  year = "1996",
  pages = "173-195",
  abstract =
    "Coercive subtyping offers a conceptually simple but powerful
    framework to understand subtyping and subset relationships in type
    theory.  In this paper we study some of its proof-theoretic and
    computational properties.",
```

```
  paper = "Jone96a.pdf",
  keywords = "printed"
}
```

———-

@miscJoyn08a, author = "Joyner, W. D.", title = Open Source Mathematical Software: A White Paper, link = "http://wdjoyner.com/writing/research/oscas-nsf-white-paper12.tex", year = "2008", paper = "Joyn08a.tex", keywords = "axiomref"

@bookJuds15, author = "Judson, Tom", title = Abstract Algebra: Theory and Applications, year = "2015", publisher = "Tom Judson", link = "http://abstract.ups.edu/download/aata-20150812.pdf", paper = "Juds15.pdf"

1.41.11 K

— **axiom.bib** —

```
@phdthesis{Kalk91,
  author = "Kalkbrener, M.",
  title = {{Three contributions to elimination theory}},
  school = "University of Linz, Austria",
  year = "1991",
  algebra =
    "\newline\refto{category TSETCAT TriangularSetCategory}
     \newline\refto{category RSETCAT RegularTriangularSetCategory}
     \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
     \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
     \newline\refto{package RSDCMPK RegularSetDecompositionPackage}"
}
```

———-

— **axiom.bib** —

```
@article{Kalk98,
  author = "Kalkbrener, M.",
  title = {{Algorithmic properties of polynomial rings}},
  journal = "Journal of Symbolic Computation",
  algebra =
    "\newline\refto{category TSETCAT TriangularSetCategory}
     \newline\refto{category RSETCAT RegularTriangularSetCategory}
     \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
     \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
     \newline\refto{package RSDCMPK RegularSetDecompositionPackage}",
  year = "1998",
  abstract =
    "In this paper we investigate how algorithms for computing heights,
    radicals, unmixed and primary decompositions of ideals can be lifted
    from a Noetherian commutative ring $R$ to polynomial rings over $R$.",
  paper = "Kalk98.pdf"
```

```
}
```

———-

— **ignore** —

```
\bibitem[Kantor 89]{Kan89} Kantor,I.L.; Solodovnikov, A.S.
  title = {{Hypercomplex Numbers}},
Springer Verlag Heidelberg, 1989, ISBN 0-387-96980-2
```

———-

— **axiom.bib** —

```
@misc{Karp14,
  author = "Karpinski, Stefan",
  title = {{Re: Symbolic Math: try a translation of Axiom to Julia?}},
  link = "\url{https://groups.google.com/forum/#!msg/julia-dev/NTfS9fJuIcE/MINQuQuGfoUJ}",
  year = "2016",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@book{Kauf00,
  author = "Kaufmann, Matt and Manolios, Panagiotis and Moore, J Strother",
  title = {{Computer-Aided Reasoning: An Approach}},
  publisher = "Kluwer Academic Publishers",
  year = "2000",
  isbn = "0-7923-7744-3",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@book{Kauf00a,
  author = "Kaufmann, Matt and Manolios, Panagiotis and Moore, J Strother",
  title = {{Computer-Aided Reasoning: ACL2 Case Studies}},
  publisher = "Kluwer Academic Publishers",
  year = "2000",
  isbn = "0-7923-7849-0",
  keywords = "shelf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Khan11a,
  author = "Khan, Muhammad Taimoor and Schreiner, Wolfgang",
  title = {{Towards a Behavioral Analysis of Computer Algebra
            Programs}},
  booktitle = "NWPT'11",
  publisher = "ACM Press",
  pages = "42-44",
  year = "2011",
  paper = "Khan11a.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Khan11b,
  author = "Khan, Muhammad Taimoor",
  title = {{Towards a Behavioral Analysis of Computer Algebra
            Programs}},
  type = "technical report",
  institution = "RISC-Linz",
  number = "11-13",
  year = "2011",
  abstract =
    "In this paper, we present our initial results on the behavioral
    analysis of computer algebra programs.  The main goal of our work is
    to find typing and behavioral errors in such programs by static
    analysis of the source code.  This task is more complex for widely
    used computer algebra languages (Maple and Mathematica) as these are
    fundamentally different from classical languages: for example they
    support non-standard types of objects, e.g.  symbols, unevaluated
    expressions, polynomials etc.; moreover they use type information to
    direct the flow of control in the program and have no clear
    difference between declaration and assignment.  For this purpose, we
    have defined the syntax and the formal type system for a substantial
    subset of the computer algebra language Maple, which we call MiniMaple.
    The type system is implemented by a type checker, which verifies
    the type correctness and respectively reports typing errors.  We have
    applied the type checker to the Maple package DifferenceDifferential
    developed at our institute.  Currently we are working on a formal
    specification language of MiniMaple and the specification of this
    package.  The specification language will be used to find errors in
    computer algebra programs with respect to their specifications.",
  paper = "Khan11b.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Khan11,
  author = "Khan, Muhammad Taimoor",
  title = {{A Type Checker for MiniMaple}},
  type = "technical report",
  institution = "RISC-Linz",
  number = "11-05",
  year = "2011",
  abstract =
    "In this paper, we present the syntactic definition and the formal type
    system for a substantial subset of the language of the computer
    algebra system Maple, which we call MiniMaple .  The goal of the type
    system is to prevent runtime typing errors by static analysis of the
    source code of MiniMaple programs.  The type system is implemented
    by a type checker, which veries the type correctness of MiniMaple
    programs respectively reports typing errors.",
  paper = "Khan11.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Khan12,
  author = "Khan, Muhammad Taimoor",
  title = {{Formal Semantics of MiniMaple}},
  year = "2012",
  type = "technical report",
  number = "12-04",
  institution = "RISC Linz",
  abstract =
    "In this paper, we give the complete definition of a formal
    denotational) semantics of a subset of the language of the
    computer algebra systems Maple which we call MiniMaple .  As a
    next step we will develop a verification calculus for this
    language.  The verification conditions generated by the calculus
    must be sound with respect to the formal semantics.",
  paper = "Khan12.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Khan12a,
  author = "Khan, Muhammad Taimoor",
  title = {{Formal Semantics of a Specification Language MiniMaple}},
  year = "2012",
  type = "technical report",
  number = "12-05",
  institution = "RISC Linz",
  abstract =
    "In this paper, we give the complete definition of a formal semantics
```

```
    of a specification language for MiniMaple.  This paper is an update
    of the previously reported formal (denotational) semantics of
    MiniMaple.  As a next step we will develop a verification calculus
    for MiniMaple and its specification language.  The verification
    conditions generated by the calculus must be sound with respect to
    both the formal semantics of MiniMaple and its corresponding
    specification language.",
  paper = "Khan12a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Khan12b,
  author = "Khan, Muhammad Taimoor and Schreiner, Wolfgang",
  title = {{Towards the Formal Semantics and Verification of Maple
           Programs}},
  journal = "LNAI",
  volume = "7362",
  pages = "231-247",
  year = "2012",
  isbn = "978-3-642-31373-8",
  abstract =
    "In this paper, we present our ongoing work and initial results on the
    formal specification and verification of MiniMaple (a substantial
    subset of Maple with slight extensions) programs. The main goal of our
    work is to find behavioral errors in such programs w.r.t. their
    specifications by static analysis. This task is more complex for widely
    used computer algebra languages like Maple as these are fundamentally
    different from classical languages: they support non-standard types
    of objects such as symbols, unevaluated expressions and polynomials
    and require abstract computer algebraic concepts and objects such as
    rings and orderings etc. As a starting point we have defined and
    formalized a syntax, semantics, type system and specification language
    for MiniMaple.",
  paper = "Khan12b.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Khan12c,
  author = "Khan, Muhammad Taimoor and Schreiner, Wolfgang",
  title = {{On Formal Specification of Maple Programs}},
  journal = "LNAI",
  volume = "7362",
  pages = "442-446",
  year = "2012",
  isbn = "978-3-642-31373-8",
  abstract =
```

```
    "This paper is an example-based demonstration of our initial results
    on the formal specification of programs written in the computer
    algebra language MiniMaple (a substantial subset of Maple with slight
    extensions). The main goal of this work is to define a verification
    framework for MiniMaple. Formal specification of MiniMaple programs
    is rather complex task as it supports non-standard types of objects,
    e.g. symbols and unevaluated expressions, and additional functions
    and predicates, e.g. runtime type tests etc. We have used the
    specification language to specify various computer algebra concepts
    respective objects of the Maple package DifferenceDifferential
    developed at our institute.",
  paper = "Khan12c.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Khan13,
  author = "Khan, Muhammad Taimoor",
  title = {{On the Formal Verification of Maple Programs}},
  year = "2013",
  type = "technical report",
  number = "13-07",
  institution = "RISC Linz",
  abstract =
    "In this paper, we present an example-based demonstration of our
    recent results on the formal verification of programs written in the
    computer algebra language (Mini)Maple (a slightly modified subset of
    Maple).  The main goal of this work is to develop a verification
    framework for behavioral analysis of MiniMaple programs.  For
    verification, we translate an annotated MiniMaple program into the
    language Why3ML of the intermediate verification tool Why3 developed
    at LRI, France.  We generate verification conditions by the
    corresponding component of Why3 and later prove the correctness of
    these conditions by various supported by the Why3 back-end automatic
    and interactive theorem provers.  We have used the verification
    framework to verify some parts of the main test example of our
    verification framework, the Maple package DifferenceDifferential
    developed at our institute to compute bivariate
    difference-differential polynomials using relative Groebner bases.",
  paper = "Khan13.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Khan14,
  author = "Khan, Muhammad Taimoor",
  title = {{Formal Specification and Verification of Computer Algebra
```

```
          Software}},
  school = "Johannes Kepler University, Linz",
  year = "2014",
  link =
    "\url{http://www.risc.jku.at/publications/download/risc_4981/main.pdf}",
  abstract =
    "In this thesis, we present a novel framework for the formal
    specification and verification of computer algebra programs and its
    application to a non-trivial computer algebra package.  The programs
    are written in the language MiniMaple which is a substantial subset of
    the language of the commercial computer algebra system Maple.  The
    main goal of the thesis is the application of light-weight formal
    methods to MiniMaple programs (annotated with types and behavioral
    specifications) for finding internal inconsistencies and violations of
    methods preconditions by employing static program analysis.  This task
    is more complex for a computer algebra language like Maple that for
    conventional programming languages, as Maple supports non-standard
    types of objects and also requires abstract data types to model
    algebraic concepts and notions.

    As a starting point, we have defined
    and formalized a syntax, semantics, type system and specification
    language for MiniMaple .  For verification, we automatically
    translate the (types and specification) annotated MiniMaple program into a
    behaviorally equivalent program in the intermediate language Why3ML of
    the verification tool Why3; from the translated program, Why3
    generates verification conditions whose correctness can be proved by
    various automated and interactive theorem provers (e.g.  Z3 and Coq).
    Furthermore, we have defined a denotational semantics of MiniMaple and
    its specification language and proved the soundness of the translation
    with respect to the operational semantics of Why3ML. Finally, we
    discuss the application of our verification framework to the Maple
    package DifferenceDifferential developed at our institute to compute
    bivariate difference-differential dimension polynomials using relative
    Groebner bases.",
  paper = "Khan14.pdf",
  keywords = "axiomref"
}
```

———-

— axiom.bib —

```
@article{Kies03,
  author = "Kiessling, Robert and Luo, Zhaohui",
  title = {{Coercions in Hindley-Milner Systems}},
  journal = "LNCS",
  volume = "3085",
  year = "2003",
  abstract =
    "Coercive subtyping is a theory of abbreviation for dependent type
    theories. In this paper, we incorporate the idea of coercive subtyping
    into the traditional Hindley-Milner type systems in functional
```

```
    programming languages. This results in a typing system with coercions,
    an extension of the Hindley-Milner type system. A type inference
    algorithm is developed and shown to be sound and complete with respect
    to the typing system. A notion of derivational coherence is developed
    to deal with the problem of ambiguity and the corresponding type
    inference algorithm is shown to be sound and complete.",
  paper = "Kies03.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Knuth 71]{Knu71} Knuth, Donald
  title = {{The Art of Computer Programming}},
2nd edition Vol. 2 (Seminumerical Algorithms) 1st edition, 2nd printing,
Addison-Wesley 1971, p. 397-398
```

———-

— **ignore** —

```
\bibitem[Knuth 84]{Knu84} Knuth, Donald
{\it The \TeX{}book}.
Reading, Massachusetts, Addison-Wesley Publishing Company, Inc.,
1984. ISBN 0-201-13448-9
```

———-

— **axiom.bib** —

```
@book{Knut92,
  author = "Knuth, Donald E.",
  title = {{Literate Programming}},
  publisher = "Center for the Study of Language and Information, Stanford CA",
  year = "1992",
  isbn = "0-937073-81-4"
}
```

———-

— **ignore** —

```
\bibitem[Knu98]{Knu98} Donald Knuth
  title = {{The Art of Computer Programming}},
Vol. 3 (Sorting and Searching)
Addison-Wesley 1998
```

———-

— **ignore** —

```
\bibitem[Kobayashi 89]{Koba89} Kobayashi, H.; Moritsugu, S.; Hogan, R.W.
  title = {{On Radical Zero-Dimensional Ideals}},
J. Symbolic Computations 8, 545-552 (1989)
  link = "\url{http://www.sciencedirect.com/science/article/pii/S0747717189800604/pdf?md5=f06dc6269514c90dcae57f
  paper = "Koba88.pdf"
```

———-

— **axiom.bib** —

```
@book{Kobl87,
  author = "Koblitz, N.",
  title = {{A Cource in Number Theory and Cryptography}},
  publisher = "Springer-Verlog",
  year = "1987"
}
```

———-

— **axiom.bib** —

```
@article{Kohl16,
  author = "Kohlhase, Michael and Rabe, Florian",
  title = {{QED Reloaded: Towards a Pluralistic Formal Library of
            Mathematical Knowledge}},
  journal = "Journal of Formalized Reasoning",
  volume = "9",
  number = "1",
  pages = "201-234",
  year = "2016",
  abstract =
   "Proposed in 1994, the ‘‘QED project’’ was one of the seminally
   inflnuential initiatives in automated reasoning: It envisioned the
   formalization of ‘‘all of mathematics’’ and the assembly of these
   formalizations in a single coherent database. Even though it never led
   to the concrete system, communal resource, or even joint research
   envisioned in the QED manifesto, the idea lives on and shapes the
   research agendas of a significant part of the community This paper
   surveys a decade of work on representation languages and knowledge
   management tools for mathematical knowledge conducted in the KWARC
   research group at Jacobs University Bremen. It assembles the various
   research strands into a coherent agenda for realizing the QED dream
   with modern insights and technologies.",
  paper = "Kohl16.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Kolchin 73]{Kol73} Kolchin, E.R.
  title = {{Differential Algebra and Algebraic Groups}},
(Academic Press, 1973).
```

———-

— **axiom.bib** —

```
@article{Kome12,
  author = "Komendantsky, V. and Konovalov, A. and Linton, S.A.",
  title = {{Interfacing Coq + SSReflect with GAP}},
  journal = "Electronic Notes in Theoretical Computer Science",
  year = "2012",
  volume = "295",
  number = "19",
  pages = "17-28",
  abstract =
    "We report on an extendable implementation of the communication
    interface connecting Coq proof assistant to the computational algebra
    system GAP using the Symbolic Computation Software Composability
    Protocol (SCSCP). It allows Coq to issue OpenMath requests to a local
    or remote GAP instances and represent server responses as Coq terms.",
  paper = "Kome12.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Kono17,
  author = "Konovalov, Alexander and Linton, Steve",
  title = {{Symbolic Computation Software Composability Protocol}},
  version = "2.2.2",
  year = "2017",
  link = "\url{}",
  paper = "Kono17.pdf",
  keywords = "CAS-Proof"
}
```

———-

— **ignore** —

```
\bibitem[Koutschan 10]{Kou10} Koutschan, Christoph
  title = {{Axiom / FriCAS}},
  link = "\url{http://www.risc.jku.at/education/courses/ws2010/cas/axiom.pdf}",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Kozen 86]{KL86} Kozen, Dexter; Landau, Susan
  title = {{Polynomial Decomposition Algorithms}},
Journal of Symbolic Computation (1989) 7, 445-456
```

———-

— **axiom.bib** —

```
@misc{Krei14,
  author = "Kreinovich, Vladik",
  title = {{Constructive Mathematics in St. Petersburg, Russia:
            A (Somewhat Subjective) View from Within}},
  link = "\url{}",
  abstract =
    "In the 1970 and 1980s, logic and constructive mathematics were an
    important part of my life; its what I defended in my Masters thesis,
    it was an important part of my PhD dissertation. I was privileged to
    work with the giants. I visited them in their homes. They were who I
    went to for advice.  And this is my story.",
  paper = "Krei14.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kryv13,
  author = "Kryvolap, Andrii and Nikitchenko, Mykola and Schreiner,
            Wolfgang",
  title = {{Extending Floyd-Hoare Logic for Partial Pre- and
            Postconditions}},
  booktitle = "ICTERI 2013: 9th Int. Conf. on ICT in Education,
               Research and Industrial Applications",
  pages = "0-23",
  publisher = "Springer",
  isbn = "978-3-319-03997-8",
  year = "2014",
  abstract =
    "Traditional (classical) Floyd-Hoare logic is defined for a case of
    total pre- and postconditions while programs can be partial. In the
    chapter we propose to extend this logic for partial conditions. To
    do this we first construct and investigate special program algebras of
    partial predicates, functions, and programs. In such algebras
    program correctness assertions are presented with the help of a
    special composition called Floyd-Hoare composition.  This composition
    is monotone and continuous. Considering the class of constructed
    algebras as a semantic base we then define an extended logic  Partial
    Floyd-Hoare Logic  and investigate its properties. This logic has
    rather complicated soundness constraints for inference rules,
    therefore simpler sufficient constraints are proposed. The logic
```

```
    constructed can be used for program verification.",
  paper = "Kryv13.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Kuts12,
  author = "Kutsia, Temur and Marin, Mircea",
  title = {{Solving, Reasoning, and Programming in Common Logic}},
  type = "technical report",
  institution = "RISC Linz",
  year = "2012",
  abstract =
    "Common Logic (CL) is a recent ISO standard for exchanging logic-based
    information between disparate computer systems.  Sharing and reasoning
    upon knowledge represented in CL require equation solving over terms
    of this language. We study computationally well-behaved fragments of
    such solving problems and show how they can influence reasoning in CL
    and transformations of CL expressions.",
  paper = "Kuts12.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Kuts12a,
  author = "Kutsia, Temur and Marin, Mircea",
  title = {{Solving, Reasoning, and Programming in Common Logic}},
  booktitle = "SYNASC '12",
  isbn = "978-0-7695-4934-7",
  pages = "119-126",
  year = "2012",
  abstract =
    "Common Logic (CL) is a recent ISO standard for exchanging logic-based
    information between disparate computer systems.  Sharing and reasoning
    upon knowledge represented in CL require equation solving over terms
    of this language. We study computationally well-behaved fragments of
    such solving problems and show how they can influence reasoning in CL
    and transformations of CL expressions.",
  paper = "Kuts12a.pdf"
}
```

———-

1.41.12 L

— **axiom.bib** —

```
@misc{Lamp95,
  author = "Lamport, Leslie",
  title = {{Types Are Not Harmless}},
  year = "1995",
  paper = "Lamp95.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Lamp86,
  author = "Lamport, Leslie",
  title = {{LaTeX: A Document Preparation System}},
  publisher = "Addison-Wesley Publishing Company, Reading, Massachusetts",
  year = "1986",
  isbn = "0-201-15790-X"
}
```

———-

— **ignore** —

```
\bibitem[Lautrup 71]{Lau71} Lautrup B.
  title = {{An Adaptive Multi-dimensional Integration Procedure}},
Proc. 2nd Coll. on Advanced Methods in Theoretical Physics, Marseille. (1971)
```

———-

— **ignore** —

```
\bibitem[Lawson 77]{Law77} Lawson C L.
  title = {{Software for C  Surface Interpolation}},
Mathematical Software III. (ed J R Rice) Academic Press. 161--194. (1977)
```

———-

— **ignore** —

```
\bibitem[Lawson 74]{LH74} Lawson C L.;  Hanson R J.
  title = {{Solving Least-squares Problems}},
Prentice-Hall. (1974)
```

———-

— **axiom.bib** —

```
@article{Laws79,
  author = "Lawson, C.L. and Hanson R.J. and Kincaid, D.R. and Krogh, F.T.",
  title = {{Algorithm 539: Basic linear algebra subprograms for
```

```
           FORTRAN usage}},
  journal = "ACM Transactions on Mathematical Software",
  volume = "5",
  number = "3",
  month = "September",
  year = "1979",
  pages = "308-323"
}
```

———-

— **axiom.bib** —

```
@article{Laws79a,
  author = "Lawson, C. L. and Hanson, R. J. and Kincaid, D.R. and
           Krogh, F.T.",
  title = {{Basic Linear Algebra Subprograms for Fortran Usage"}},
  journal = "Transactions on Mathematical Software",
  volume = "5",
  number = "3",
  year = "1979",
  month = "September",
  pages = "308-325",
  abstract =
    "A package of 38 low level subprograms for many of the basic
    operations of numerical linear algebra is presented. The package is
    intended to be used with Fortran. The operations in the package
    include dot product, elementary vector operations, Givens
    transformations, vector copy and swap, vector norm, vector scaling,
    and the determination of the index of the vector component of largest
    magnitude.",
  paper = "Laws79.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Laza91,
  author = "Lazard, Daniel",
  title = {{A New Method for Solving Algebraic Systems of Positive Dimension}},
  journal = "Discrete. Applied. Mathematics",
  volume = "33",
  year = "1991",
  pages = "147-160",
  algebra =
    "\newline\refto{category TSETCAT TriangularSetCategory}
     \newline\refto{category RSETCAT RegularTriangularSetCategory}
     \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
     \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
     \newline\refto{package RSDCMPK RegularSetDecompositionPackage}",
  abstract =
```

```
    "A new algorithm is presented for solving algebraic systems of
    equations, which is designed from the structure which is wanted for
    the result. This algorithm is not yet implemented; thus technical
    details and proofs are omitted, for emphasising on the relation
    between the algorithm design and a good representation of the
    result. The algorithm is based on a new theorem of decomposition for
    algebraic varieties.",
  paper = "Laza91.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Laza92,
  author = "Lazard, Daniel",
  title = {{Solving Zero-dimensional Algebraic Systems}},
  journal = "J. of Symbolic Computation",
  volume = "13",
  pages = "117-131",
  year = "1992",
  abstract =
    "It is shown that a good output for a solver of algebraic systems of
     dimension zero consists of a family of ‘‘triangular sets of
     polynomials’’. Such an output is simple, readable, and consists
     of all information which may be wanted.

     Different algorithms are described for handling triangular systems
     and obtaining them from Groebner bases. These algorithms are
     practicable, and most of them are polynomial in the number of
     solutions",
  paper = "Laza92.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Laza90,
  author = "Lazard, Daniel and Rioboo, Renaud",
  title = {{Integration of rational functions: Rational computation of the
           logarithmic part}},
  journal = "Journal of Symbolic Computation",
  volume = "9",
  number = "2",
  year = "1990",
  month = "February",
  pages = "113-115",
  abstract = "
    A new formula is given for the logarithmic part of the integral of a
    rational function, one that strongly improves previous algorithms and
    does not need any computation in an algebraic extension of the field
```

```
    of constants, nor any factorisation since only polynomial arithmetic
    and GCD computations are used. This formula was independently found
    and implemented in SCRATCHPAD by B.M. Trager.",
  paper = "Laza90.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@article{LeBr88,
  author =  "Le Brigand, D.; Risler, J.J.",
  title = {{Algorithme de Brill-Noether et codes de Goppa}},
  journal = "Bull. Soc. Math. France",
  volume = "116",
  year = "1988",
  pages = "231--253"
}
```

———-

— **axiom.bib** —

```
@misc{Leec92,
  author = "Leech, J.",
  title = {{Private Communication}},
  comment = "to James Davenport",
  year = "1992"
}
```

———-

— **axiom.bib** —

```
@book{Lege11,
  author = "Legendre, George L. and Grazini, Stefano",
  title = {{Pasta by Design}},
  publisher = "Thames and Hudson",
  isbn = "978-0-500-51580-8",
  year = "2011"
}
```

———-

— **axiom.bib** —

```
@article{Lens81,
  author = "Lenstra Jr., H.W.",
  title = {{Primality Testing Algorithms}},
  journal = "Lecture Notes in Mathematics",
```

```
  volume = "901",
  publisher = "Springer-Verlag",
  pages = "243-257",
  year = "1981"
}
```

———-

— **axiom.bib** —

```
@article{Lesc87,
  author = "Lescanne, Pierre",
  title = {{Current Trends in Rewriting Techniques and Related Problems}},
  journal = "Lecture Notes in Computer Science",
  volume = "296",
  year = "1987",
  pages = "38-51",
  paper = "Lesc87.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Lesc95,
  author = "Lescanne, Pierre",
  title = {{The Lambda Calculus as an Abstract Data Type}},
  booktitle = "Lecture Notes in Computer Science",
  publisher = "Springer",
  volume = "1130",
  year = "1995",
  abstract =
    "Calculi of explicit substitutions are theoretical tools to
    describe lambda calculus as first-order rewrite systems. Therefore
    a presentation as an abstract data type is possible. During an
    invited lecture at the Workshop on Abstract Data Types, we
    presented calculi of explicit substitutions, especially the
    calculus $\lambda$v. The theoretical background of $\lambda$v can
    be found in [1], therefore we are not entering in the details of
    the theory as we did during the oral presentation. However, we
    take the essence of the talk, namely the formalizatino of
    $\lambda$v in a specification language based on ADT. We have
    chosen LSL, the LARCH Shared Language [7] to illustrate the beauty
    of abstract data types which lies in its precision and
    concision. Moreoever we take freedom with respect to this
    language.",
  paper = "Lesc95.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Leop03,
  author = "Leopardi, Paul",
  title = {{A quick introduction to Clifford Algebras}},
  publisher = "School of Mathematics, University of New South Wales",
  year = "2003",
  paper = "Leop03.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Lehn10,
  author = "Lehner, Franz",
  title = {{FriCAS Tutorium}},
  year = "2010",
  link = "\url{https://www.math.tugraz.at/mathc/compmath2/Demo/fricas-tutorium-0.6.pdf}",
  paper = "Lehn10.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Lest01,
  author = "Lester, D.R.",
  title = {{Effective Continued Fractions}},
  booktitle = "15th IEEE Symp. on Computer Arithmetic",
  publisher = "IEEE Computer Society Press",
  year = "2001",
  pages = "163-170",
  abstract =
    "Only the leading seven terms of a continued fraction are needed to
    perform on-line arithmetic, provided the continued fractions are of
    the correct form.  This forms the basis of a proof that there is an
    effective representation of the computable reals as continued
    fractions: we also demonstrate that the basic arithmetic operations
    are computable using this representation.",
  paper = "Lest01.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Leve80,
  author = "Levenworth, B.",
  title = {{ADAPT Reference Manual}},
  comment = "IBM Research",
```

```
  year = "1980"
}
```

———-

‘

— **ignore** —

```
\bibitem[Lewis 77]{Lew77} Lewis J G,
  title = {{Algorithms for sparse matrix eigenvalue problems}},
Technical Report STAN-CS-77-595. Computer Science Department,
Stanford University. (1977)
```

———-

— **ignore** —

```
\bibitem[Linger 79]{LMW79} Linger, Richard C.; Mills, Harlan D.;
Witt, Bernard I.
  title = {{Structured Programming: Theory and Practice}},
Addison-Wesley (March 1979) ISBN 0201144611
```

———-

— **axiom.bib** —

```
@article{Lint02,
  author = "Linton, S. and Sebastiani, R.",
  title = {{Editorial: The Integration of Automated Reasoning and Computer
           Algebra Systems}},
  journal = "J. Symbolic Computation",
  volume = "34",
  pages = "239-239",
  year = "2002",
  paper = "Lint02.pdf",
  keywords = "CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Lisk77,
  author = "Liskov, Barbara and Synder, Alan and Atkinson, Russell and
            Schaffert, Craig",
  title = {{Abstraction Mechanisms in CLU}},
  journal = "CACM",
  volume = "20",
  number = "8",
  year = "1977",
  link = "\url{https://www.cs.virginia.edu/~weimer/615/reading/liskov-clu-abstraction.pdf}",
  abstract =
```

```
    "CLU is a new programming language designed to support the use of
    abstractions in program construction. Work in programming methodology
    has led to the realization that three kinds of abstractions --
    procedural, control, and especially data abstractions -- are useful in
    the programming process. Of these, only the procedural abstraction is
    supported well by conventional languages, through the procedure or
    subroutine. CLU provides, in addition to procedures, novel linguistic
    mechanisms that support the use of data and control abstractions. This
    paper provides an introduction to the abstraction mechanisms of
    CLU. By means of programming examples, the utility of the three kinds
    of abstractions in program construction is illustrated, and it is
    shown how CLU programs may be written to use and implement
    abstractions. The CLU library, which permits incremental program
    development with complete type checking performed at compile time, is
    also discussed.",
  paper = "Lisk77.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Lisk79,
  author = "Liskov, Barbara and Atkinson, Russ and Bloom, Toby and
            Moss, Eliot and Schaffert, Craig and Scheifler, Bob and
            Snyder, Alan",
  title = {{CLU Reference Manual}},
  institution = "Massachusetts Institute of Technology",
  year = "1979",
  paper = "Lisk79.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Loba08,
  author = "Lobachev, Oleg and Loogen, Rita",
  title = {{Towards an Implementation of a Computer Algebra System in a
           Functional Language}},
  journal = "LNAI",
  volume = "5144",
  pages = "141-208",
  year = "2008",
  publisher = "Springer-Verlag",
  abstract =
    "This paper discusses the pros and cons of using a functional language
    for implementing a computer algebra system. The contributions of the
    paper are twofold. Firstly, we discuss some language-centered design
    aspects of a computer algebra system -- the ``language unity''
    concept.  Secondly, we provide an implementation of a fast polynomial
    multiplication algorithm, which is one of the core elements of a
```

```
    computer algebra system.  The goal of the paper is to test the
    feasibility of an implementation of (some elements of) a computer
    algebra system in a modern functional language.",
  paper = "Loba08.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@misc{Loet09,
  author = "Loetzsch, Martin and Bleys, Joris and Wellens, Pieter",
  title = {{Understanding the Dynamics of Complex Lisp Programs}},
  year = "2009",
  link = "\url{http://www.martin-loetzsch.de/papers/loetzsch09understanding.pdf}",
  paper = "Loet09.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Loet00,
  author = "Loetzsch, M.",
  title = {{GTFL - A graphical terminal for Lisp}},
  year = "2000",
  link = "\url{http://martin-loetzsch.de/gtfl}"
}
```

———-

— **axiom.bib** —

```
@book{Losc60,
  author = {L\"osch, Friedrich},
  title = {{Tables of Higher Functions}},
  publisher = "McGraw-Hill Book Company",
  year = "1960",
  algebra = "\newline\refto{package DFSFUN DoubleFloatSpecialFunctions}"
}
```

———-

— **ignore** —

```
\bibitem[LTU10]{LTU10}.
  title = {{Lambda the Ultimate}},
  link = "\url{http://lambda-the-ultimate.org/node/3663#comment-62440}",
```

———-

— **axiom.bib** —

```
@book{Luke69a,
  author = "Luke, Yudell L.",
  title = {{The Special Functions and their Approximations}},
  volume = "1",
  publisher = "Academic Press",
  year = "1969",
  booktitle = "Mathematics in Science and Engineering Volume 53-I",
  algebra = "\newline\refto{package DFSFUN DoubleFloatSpecialFunctions}"
}
```

———-

— **axiom.bib** —

```
@book{Luke69b,
  author = "Luke, Yudell L.",
  title = {{The Special Functions and their Approximations}},
  volume = "2",
  publisher = "Academic Press",
  year = "1969",
  booktitle = "Mathematics in Science and Engineering Volume 53-I",
  algebra = "\newline\refto{package DFSFUN DoubleFloatSpecialFunctions}"
}
```

———-

— **axiom.bib** —

```
@book{Luke77,
  author = "Luke, Yudell L.",
  title = {{Algorithms for the Computation of Mathematical Functions}},
  publisher = "Academic Press",
  year = "1977",
  isbn = "978-0124599406"
}
```

———-

— **axiom.bib** —

```
@article{Luox99,
  author = "Luo, Zhaohui and Soloviev, Sergei",
  title = {{Dependent Coercions}},
  journal = "Electr. Notes Theor. Comput. Sci.",
  volume = "29",
  year = "1999",
  pages = "152-168",
```

```
  abstract =
    "A notion of dependent coercion is introduced and studied in the
    context of dependent type theories. It extends our earlier work on
    coercive subtyping into a uniform framework which increases the
    expressive power with new applications.  A dependent coercion
    introduces a subtyping relation between a type and a family of types
    in that an object of the type is mapped into one of the types in the
    family.  We present the formal framework, discuss its meta-theory, and
    consider applications such as its use in functional programming with
    dependent types.",
  paper = "Luox99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Lutz90,
  author = "Lutzen, Jesper",
  title = {{Joseph Liouville. 1809-1882: Master of Pure and Applied
           Mathematics}},
  publisher = "Springer",
  year = "1990",
  paper = "Lutz90.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Lutz90a,
  author = "Lutzen, Jesper",
  title = {{Integration in Finite Terms}},
  publisher = "Springer",
  year = "1990",
  comment = "Chapter IX",
  pages = "351-422",
  paper = "Lutz90a.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Lyness 83]{Lyn83} Lyness J N.
  title = {{When not to use an automatic quadrature routine}},
SIAM Review. 25 63--87. (1983)
```

———-

1.41.13 M

— **ignore** —

```
\bibitem[Mac Lane 79]{MB79} Mac Lane, Saunders; Birkhoff, Garret
  title = {{Algebra}},
AMS Chelsea Publishing ISBN 0821816462
```

———-

— **axiom.bib** —

```
@article{Maga00,
  author = "Magaud, Nicolas and Bertot, Yves",
  title = {{Changing Data Structures in Type Theory: A Study of
            Natural Numbers}},
  journal = "LNCS",
  volume = "2277",
  pages = "181-196",
  year = "2000",
  abstract =
    "In type-theory based proof systems that provide inductive
    structures, computation tools are automatically associated to
    inductive definitions. Choosing a particular representation for a
    given concept has a strong influence on proof structure. We
    propose a method to make the change from one representation to
    another easier, by systematically translating proofs from one
    context to another. We show how this method works by using it on
    natural numbers, for which a unary representation (based on Peano
    axioms) and abinary representation are available. This method
    leads to an automatic translation tool that we have implemented in
    Coq and successfully applied to several arithmetical theorems.",
  paper = "Maga00.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Maga08,
  author = "Magaud, Nicolas and Narboux, Julien and Schreck, Pascal",
  title = {{Formalizing Projective Plane Geometry in Coq}},
  journal = "LNCS",
  volume = "6301",
  pages = "141-162",
  year = "2008",
  abstract =
    "We investigate how projective plane geometry can be formalized in a
    proof assistant such as Coq. Such a formalization increases the
    reliability of textbook proofs whose details and particular cases are
    often overlooked and left to the reader as exercises. Projective plane
```

```
    geometry is described through two different axiom systems which are
    formally proved equivalent. Usual properties such as decidability of
    equality of points (and lines) are then proved in a constructive
    way. The duality principle as well as formal models of projective
    plane geometry are then studied and implemented in Coq. Finally, we
    formally prove in Coq that Desargues property is independent of the
    axioms of projective plane geometry.",
  paper = "Maga08.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Magu17,
  author = "Maguire, Camm and Schelter, William",
  title = {{Gnu Common Lisp}},
  link = "\url{https://savannah.gnu.org/projects/gcl}"
}
```

———-

— **axiom.bib** —

```
@misc{Magu20,
  author = "Maguire, Camm",
  title = {{Axiom Build Status}},
  link = "\url{https://buildd.debian.org/status/package.php?p=axiom}"
}
```

———-

— **ignore** —

```
\bibitem[Malcolm 72]{Mal72} Malcolm M. A.
  title = {{Algorithms to reveal properties of floating-point arithmetic}},
Comms. of the ACM, 15, 949-951.  (1972)
```

———-

— **ignore** —

```
\bibitem[Malcolm 76]{MS76} Malcolm M A.; Simpson R B.
  title = {{Local Versus Global Strategies for Adaptive Quadrature}},
ACM Trans. Math. Softw. 1 129--146. (1976)
```

———-

— **ignore** —

```
\bibitem[Marden 66]{Mar66} Marden M.
  title = {{Geometry of Polynomials}},
Mathematical Surveys. 3 Am. Math. Soc., Providence, RI. (1966)
```

———-

— **axiom.bib** —

```
@incollection{Mark62,
  author = "Markov, Andrei .A.",
  title = {{On Constructive Mathematics}},
  booktitle = "Problems of the Constructive Direction in Mathematics:
               Part 2 -- Constructive Mathematical Analysis",
  publisher = "Academy of Science USSR",
  comment = "In Russian",
  link = "\url{http://www.mathnet.ru/links/4fe363fcbbf9aeaad8dc9baed1c7d1c8/tm1756.pdf}",
  year = "1962"
}
```

———-

— **axiom.bib** —

```
@misc{Mars07,
  author = "Marshak, U.",
  title = {{HT-AJAX - AJAX framework for Hunchentoot}},
  year = "2007",
  link = "\url{http://common-lisp.net/project/ht-ajax/ht-ajax.html}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Matt10,
  author = "Matthews, David C.J. and Wenzel, Makarius",
  title = {{Efficient Parallel Programming in Poly/ML and Isabelle/ML}},
  booktitle = "Proc. 5th ACM SIGPLAN workshop on Declarative aspects of
               multicore programming",
  pages = "53-62",
  year = "2010",
  publisher = "ACM",
  isbn = "978-1-60558-859-9",
  abstract =
    "The ML family of languages and LCF-style interactive theorem proving
    have been closely related from their beginnings about 30 years
    ago. Here we report on a recent project to adapt both the Poly/ML
    compiler and the Isabelle theorem prover to current multicore
    hardware. Checking theories and proofs in typical Isabelle application
    takes minutes or hours, and users expect to make efficient use of
    ‘‘home machines’’ with 2-8 cores, or more.
```

```
    Poly/ML and Isabelle are big and complex software systems that have
    evolved over more than two decades. Faced with the requirement to
    deliver a stable and efficient parallel programming environment, many
    infrastructure layers had to be reworked: from low-level system
    threads to high-level principles of value-oriented programming. At
    each stage we carefully selected from the many existing concepts for
    parallelism, and integrated them in a way that fits smoothly into the
    idea of purely functional ML with the addition of synchronous
    exceptions and asynchronous interrupts.

    From the Isabelle/ML perspective, the main concept to manage parallel
    evaluation is that of ''future values''. Scheduling is implicit, but it
    is also possible to specify dependencies and priorities. In addition,
    block-structured groups of futures with propagation of exceptions
    allow for alternative functional evaluation (such as parallel search),
    without requiring user code to tackle concurrency. Our library also
    provides the usual parallel combinators for functions on lists, and
    analogous versions on prover tactics.

    Despite substantial reorganization in the background, only minimal
    changes are occasionally required in user ML code, and none at the
    Isabelle application level (where parallel theory and proof processing
    is fully implicit). The present implementation is able to address more
    than 8 cores effectively, while the earlier version of the official
    Isabelle2009 release works best for 2-4 cores. Scalability beyond 16
    cores still poses some extra challenges, and will require further
    improvements of the Poly/ML runtime system (heap management and
    garbage collection), and additional parallelization of Isabelle
    application logic.",
  paper = "Matt10.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Maxi16a,
  author = "Maxima",
  title = {{Symbolic Integration: The Algorithms}},
  link = "\url{http://maxima.sourceforge.net/docs/tutorial/en/gaertner-tutorial-revision/Pages/SI001.htm}"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Mayr09,
  author = "Mayrhofer, Gunther",
  title = {{Symbolic COmputation Prover with Induction}},
  year = "2009",
  link =
```

```
    "\url{http://www.risc.jku.at/publications/download/risc_3910/Thesis.pdf}",
  school = "RISC Linz",
  abstract =
    "An important task in automated theorem proving is computing with
    ‘‘arbitrary but fixed’’ constants. This kind of highschool proving
    occurs in the main part of most proofs. The current master’s thesis
    presents an automated prover that focuses on such computations with
    symbols. The prover uses equalities and equivalences in the knowledge
    base to rewrite a goal formula. In all formulae there may be universal
    quantifiers and some selected logical connectives. Special syntax
    elements support case distinctions and sequence variables. The prover
    uses a specialized method for proving equalities and an important
    feature is proving by cases. An extension allows induction over some
    predefined domains. Additionally to the prover implementation in
    Mathematica, there is a tracer that prints a protocol of the proof
    flow. Since the code for this tracer is separated from the prover,
    there may be more than one tracer with different output. But more
    important is that a user can inspect the code of prover without being
    confused by technical details for generating some nice output. The
    main motivation for this prover is a new extension of the Theorema
    system. The aim is an environment for defining new prover in the same
    language as theorems. The advantage is clear, existing prover may
    prove facts of a new one, for example the correctness. Using this it
    is possible to build up a hierarchy of formally checked provers. For
    such reasoning about reasoners a starting point needs induction on the
    structure of terms and formulae. A first prover in the hierarchy will
    need computations with symbols in many proof branches. This may be
    done by the current Symbolic Computation Prover.",
  paper = "Mayr09.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Maye17,
  author = "Mayero, Micaela and Delahaye, David",
  title = {{A Maple Mode for Coq}},
  link = "\url{https://github.com/coq-contribs/maple-mode}",
  year = "2017",
  abstract =
    "This contribution is an interface between Coq and Maple. In
    particular, this exports the functions simplify/factor/expand/normal
    giving the corresponding tactics Simplify/Factor/Expand/Normal. The
    manipulations carried out by these tactics are proved thanks to the
    tactic Field. These tactics can be also used as functions by means of
    the Eval ... In command. "
}
```

———-

— **axiom.bib** —

```
@article{Maza95,
  author = "Maza, Marc Moreno and Rioboo, Renaud",
  title = {{Polynomial Gcd Computations over Towers of Algebraic Extensions}},
  year = "1995",
  journal = "Proceedings of AAECC11",
  algebra =
    "\newline\refto{category TSETCAT TriangularSetCategory}
     \newline\refto{category RSETCAT RegularTriangularSetCategory}
     \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
     \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
     \newline\refto{package LEXTRIPK LexTriangularPackage}
     \newline\refto{package RSDCMPK RegularSetDecompositionPackage}",
  abstract =
    "Some methods for polynomial system solving require efficient
    techniques for computing univariate polynomial gcd over algebraic
    extensions of a field. Currently used techniques compute {\sl generic}
    univariate polynomial gcd before {\sl specializing} the result using
    algebraic relations in the ring of coefficients. This strategy
    generates very big intermediate data and fails for many problems. We
    present here a new approach which takes permanently into account those
    algebraic relations. It is based on a property of subresultant
    remainder sequences and leads to a great increase of the speed of
    computations and thus the size of accessible systems.",
  paper = "Maza95.pdf",
  keywords = "axiomref"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Maza97,
  author = "Maza, Marc Moreno",
  title = {{Calculs de pgcd au-dessus des tours d'extensions simples et
           resolution des systemes d'equations algebriques}},
  school = "Universite P.etM. Curie",
  year = "1997",
  algebra =
    "\newline\refto{category TSETCAT TriangularSetCategory}
     \newline\refto{category RSETCAT RegularTriangularSetCategory}
     \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
     \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
     \newline\refto{package RSDCMPK RegularSetDecompositionPackage}",
  link = "\url{http://www.csd.uwo.ca/~moreno//Publications/MorenoMaza-Thesis-1997.ps.gz}",
  abstract =
    "This thesis is dedicated to polynomial system solving by means of
    triangular sets. A first prt presents two algorithms to compute
    polynomial gcds over tower of simple extensions. The first one was
    designed by Renaud Rioboo and applies to algebraic towers. The second
    one is a generalization of the previous one to the most general case
    of seperable towers. These algorithms lead to an efficient
    implementation of two methods suggested by Daniel Lazard to solve
    polynomial systems by means of triangular sets. These programs solved
```

```
    problems that were previously unreachable. The second method was only
    sketched by its author. So, a second part of this thesis presents the
    necessary developements to describe a right implementation. Moreover,
    a theorecal and unified presentation, together with an experimental
    comparison with similar methods due to Wu Wen-Tsun, Dongming Wang and
    Michael Kalkbrener were realized by Philippe Aubry and are reported in
    a third part of this document.",
  paper = "Maza97.pdf",
  keyword = "axiomref"
}
```

———-

— **axiom.bib** —

```
@techreport{Maza98,
  author = "Maza, M. Moreno",
  title = {{A new algorithm for computing triangular decomposition of
           algebraic varieties}},
  institution = "Numerical Algorithms Group (NAG)",
  algebra =
    "\newline\refto{category TSETCAT TriangularSetCategory}
     \newline\refto{category RSETCAT RegularTriangularSetCategory}
     \newline\refto{category NTSCAT NormalizedTriangularSetCategory}
     \newline\refto{category SFRTCAT SquareFreeRegularTriangularSetCategory}
     \newline\refto{package RSDCMPK RegularSetDecompositionPackage}",
  year = "1998"
}
```

———-

— **axiom.bib** —

```
@article{Melh02,
  author = "Melham, Thomas F.",
  title = {{PROSPER An Investigation into Software Architecture for Embedded
           Proof Engines}},
  journal = "LNCS",
  volume = "2309",
  pages = "193-206",
  year = "2002",
  abstract =
    "Prosper is a recently-completed ESPRIT Framework IV research project
    that investigated software architectures for component-based, embedded
    formal verification tools. The aim of the project was to make
    mechanized formal analysis more accessible in practice by providing a
    framework for integrating formal proof tools inside other software
    applications. This paper is an extended abstract of an invited
    presentation on Prosper given at FroCoS 2002. It describes the vision
    of the Prosper project and provides a summary of the technical
    approach taken and some of the lessons learned.",
  paper = "Melh02.pdf",
```

```
  keywords = "CAS-Proof, printed"
}
```

———-

— **axiom.bib** —

```
@book{Meye92,
  author = "Meyer, Bertrand",
  title = {{Eiffel: The Language}},
  publisher = "Prentice Hall",
  year = "1992",
  isbn = "0-13-247925-7"
}
```

———-

— **ignore** —

```
\bibitem[Mignotte 82]{Mig82} Mignotte, Maurice
  title = {{Some Useful Bounds}},
Computing, Suppl. 4, 259-263 (1982), Springer-Verlag
```

———-

— **ignore** —

```
\bibitem[McCarthy 83]{McC83} McCarthy G J.
  title = {{Investigation into the Multigrid Code MGD1}},
Report AERE-R 10889. Harwell. (1983)
```

———-

— **ignore** —

```
\bibitem[Mie97]{Mie97} Mielenz, Klaus D.
  title = {{Computation of Fresnel Integrals}},
J. Res. Natl. Inst. Stand. Technol. (NIST) V102 No3 May-June 1997 pp363-365
```

———-

— **ignore** —

```
\bibitem[Mie00]{Mie00} Mielenz, Klaus D.
  title = {{Computation of Fresnel Integrals II}},
J. Res. Natl. Inst. Stand. Technol. (NIST) V105 No4 July-Aug 2000 pp589-590
```

———-

— **axiom.bib** —

```
@article{Mill68,
  author = "Millen, J. K.",
  title = {{CHARYBDIS: A LISP program to display mathematical expressions
           on typewriter-like devices}},
  year = "1968",
  journal = "Interactive Systems for Experimental and Applied Mathematics",
  publisher = "M. Klerer and J. Reinfelds, eds., Academic Press, New York",
  pages = "79--90",
  abstract = "
    CHARYBDIS (from CHARacter-composed sYmBolic DISplay) is a LISP program
    to display mathematical expressions on typewriter-like devices such as
    line printers, teletypes, and scopes which display lines of characters,
    as well as typewriters.",
  paper = "Mill68.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Minc 79]{Min79} Henryk Minc
  title = {{Evaluation of Permanents}},
Proc. of the Edinburgh Math. Soc.(1979), 22/1 pp 27-32.
```

———-

— **ignore** —

```
\bibitem[More 74]{MGH74} More J J.; Garbow B S.;  Hillstrom K E.
  title = {{User Guide for Minpack-1}},
ANL-80-74 Argonne National Laboratory. (1974)
```

———-

— **ignore** —

```
\bibitem[Mikhlin 67]{MS67} Mikhlin S G.; Smolitsky K L.
  title = {{Approximate Methods for the Solution of Differential and
            Integral Equations}},
Elsevier.  (1967)
```

———-

— **axiom.bib** —

```
@book{Mine88,
  author = "Mines, Ray and Richman, Fred and Ruitenburg, Wim",
```

```
  title = {{A Course in Constructive Algebra}},
  year = "1988",
  publisher = "Universitext",
  abstract =
    "The constructive approach to mathematics has enjoyed a renaissance,
    caused in large part by the appearance of Errett Bishop's book
    Foundations of constructive analysis in 1967, and by the subtle
    influences of the proliferation of powerful computers. Bishop
    demonstrated that pure mathematics can be developed from a
    constructive point of view while maintaining a continuity with
    classical terminology and spirit; much more of classical mathematics
    was preserved than had been thought possible, and no classically false
    theorems resulted, as had been the case in other constructive schools
    such as intuitionism and Russian constructivism. The computers created
    a widespread awareness of the intuitive notion of an effecti ve
    procedure, and of computation in principle, in addi tion to
    stimulating the study of constructive algebra for actual
    implementation, and from the point of view of recursive function
    theory. In analysis, constructive problems arise instantly because we
    must start with the real numbers, and there is no finite procedure for
    deciding whether two given real numbers are equal or not (the real
    numbers are not discrete) . The main thrust of constructive
    mathematics was in the direction of analysis, although several
    mathematicians, including Kronecker and van der waerden, made
    important contributions to construc tive algebra. Heyting, working in
    intuitionistic algebra, concentrated on issues raised by considering
    algebraic structures over the real numbers, and so developed a
    handmaiden'of analysis rather than a theory of discrete algebraic
    structures.",
  paper = "Mine88.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Mitchell 80]{MG80} Mitchell A R.; Griffiths D F.
  title = {{The Finite Difference Method in Partial Differential Equations}},
Wiley. (1980)
```

———-

— **ignore** —

```
\bibitem[Moler 73]{MS73} Moler C B.;  Stewart G W.
  title = {{An Algorithm for Generalized Matrix Eigenproblems}},
SIAM J. Numer. Anal. 10 241--256. 1973
```

———-

— **axiom.bib** —

```
@article{Moss93,
  author = "Mosses, Peter D.",
  title = {{The Use of Sorts in Algebraic Specification}},
  journal = "Lecture Notes in Computer Science",
  volume = "655",
  pages = "66-91",
  year = "1993",
  abstract =
    "Algebraic specification frameworks exploit a variety of sort
    disciplines. The treatment of sorts has a considerable influence on
    the ease with which such features as partiality and polymorphism can
    be specified. This survey gives an accessible overview of various
    frameworks, focusing on their sort disciplines and assessing their
    strengths and weaknesses for practical applications. Familiarity with
    the basic notions of algebraic specification is assumed.",
  paper = "Moss93.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Muld97,
  author = "Mulders, Thom",
  title = {{A Note on Subresultants and the Lazard/Rioboo/Trager Formula in
           Rational Function Integration}},
  journal = "Journal of Symbolic Computation",
  year = "1997",
  volume = "24",
  number = "1",
  month = "July",
  pages = "45-50",
  abstract = "
    An ambiguity in a formula of Lazard, Rioboo and Trager, connecting
    subresultants and rational function integration, is indicated and
    examples of incorrect interpretations are given.",
  paper = "Muld97.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Munksgaard 80]{Mun80} Munksgaard N.
  title = {{Solving Sparse Symmetric Sets of Linear Equations by
            Pre-conditioned Conjugate Gradients}},
ACM Trans. Math. Softw. 6 206--219. (1980)
```

———-

— **ignore** —

```
\bibitem[Murray 72]{Mur72} Murray W, (ed)
  title = {{Numerical Methods for Unconstrained Optimization}},
Academic Press. (1972)
```

———-

— **ignore** —

```
\bibitem[Murtagh 83]{MS83} Murtagh B A.; Saunders M A
  title = {{MINOS 5.0 User's Guide}},
Report SOL 83-20. Department of Operations Research, Stanford University 1983
```

———-

— **ignore** —

```
\bibitem[Musser 78]{Mus78} Musser, David R.
  title = {{On the Efficiency of a Polynomial Irreducibility Test}},
Journal of the ACM, Vol. 25, No. 2, April 1978, pp. 271-282
```

———-

1.41.14 N

— **axiom.bib** —

```
@article{Nara99,
  author = "Naraschewski, Wolfgang and Nipkow, Tobias",
  title = {{Type Inference Verified: Algorithm W in Isabelle/HOL}},
  journal = "LNCS",
  volume = "1512",
  year = "1999",
  pages = "317-332",
  paper = "Nara99.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Naud98,
  author = "Naudin, Patrice and Quitte, Claude",
  title = {{Univariate polynomial factorization over finite fields}},
  journal = "Theor. Comput. Sci.",
  volume = "191",
  number = "1-2",
  pages = "1-36",
```

```
  year = "1998",
  abstract =
    "This paper is a tutorial introduction to univariate polynomial
    factorization over finite fields. The authors recall the classical
    methods that induced most factorization algorithms (Berlekamps and
    the Cantor-Zassenhaus ones) and some refinements which can be applied
    to these methods. Explicit algorithms are presented in a form suitable
    for almost immediate implementation. They give a detailed description
    of an efficient implementation of the Cantor-Zassenhaus algorithm used
    in the release 2 of the Axiom computer algebra system.",
  paper = "Naud98.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Neup12,
  author = "Neuper, Walther",
  title = {{Automated Generation of User Guidance by Combining
            Computation and Deduction}},
  year = "2012",
  publisher = "Open Publishing Association",
  pages = "82-101",
  abstract =
    "Herewith, a fairly old concept is published for the first time and
    named Lucas Interpretation. This has been implemented in a
    prototype, which has been proved useful in educational practice and
    has gained academic relevance with an emerging generation of
    educational mathematics assistants (EMA) based on Computer Theorem
    Proving (CTP).

    Automated Theorem Proving (ATP), i.e.  deduction, is
    the most reliable technology used to check user input. However ATP is
    inherently weak in automatically generating solutions for arbitrary
    problems in applied mathematics. This weakness is crucial for EMAs:
    when ATP checks user input as incorrect and the learner gets stuck
    then the system should be able to suggest possible next steps.

    The key idea of Lucas Interpretation is to compute the steps of a
    calculation following a program written in a novel CTP-based
    programming language, i.e. computation provides the next steps.  User
    guidance is generated by combining deduction and computation: the
    latter is performed by a specific language interpreter, which works
    like a debugger and hands over control to the learner at breakpoints,
    i.e.  tactics generating the steps of calculation.  The interpreter
    also builds up logical contexts providing ATP with the data required
    for checking user input, thus combining computation and deduction.

    The paper describes the concepts underlying Lucas Interpretation so
    that open questions can adequately be addressed, and prerequisites for
    further work are provided.",
  paper = "Neup12.pdf"
```

```
}
```

———-

— **axiom.bib** —

```
@article{Neup13a,
  author = "Neuper, Walther",
  title = {{On the Emergence of TP-based Educational Math
            Assistants}},
  journal = "The Electronic Journal of Mathematics and Technology",
  volume = "1",
  number = "1",
  year = "2013",
  link = "\url{http://www.ist.tugraz.at/projects/isac/publ/newgen.pdf}",
  abstract =
    "Presently Computer Algebra Systems, Dynamic Geometry Systems and
    Spreadsheets cover most of e-learning in high-school mathematics and
    as well are used for education in formal parts of science.  Recently
    and largely unnoticed in public, the academic discipline of
    interactive and automated Theorem Proving (TP) has become of major
    importance for mathematics and computer science.

    This paper considers the promises of TP technology for education in
    science, technology, engineering and mathematics at the full range of
    levels from high-school to university.

    Starting from prototypes of TP-based educational mathematics systems,
    conceptual foundations are considered: TP-based software which
    implements reasoning as an essential part of mathematical thinking
    technology.  Educational objectives for the development of TP-based
    systems are discussed and concluded with some predictions on possible
    impact of TP-based educational mathematics assistants.

    The final conclusion suggests to announce the emergence of a new,
    TP-based generation of educational mathematics software.",
  paper = "Neup13a.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Nijenhuis 78]{NW78} Nijenhuis and Wilf
  title = {{Combinatorical Algorithms}},
Academic Press, New York 1978.
```

———-

— **ignore** —

```
\bibitem[Nikolai 79]{Nik79} Nikolai P J.
  title = {{Algorithm 538: Eigenvectors and eigenvalues of real generalized
           symmetric matrices by simultaneous iteration}},
ACM Trans. Math. Softw. 5 118--125. (1979)
```

———-

— **axiom.bib** —

```
@book{Nils76,
  author = "Nilsson, Nils J.",
  title = {{Principles of Artificial Intelligence}},
  publisher = "Morgan Kaufmann",
  year = "1976",
  abstract =
    "A classic introduction to artificial intelligence intended to bridge
    the gap between theory and practice, Principles of Artificial
    Intelligence describes fundamental AI ideas that underlie applications
    such as natural language processing, automatic programming, robotics,
    machine vision, automatic theorem proving, and intelligent data
    retrieval. Rather than focusing on the subject matter of the
    applications, the book is organized around general computational
    concepts involving the kinds of data structures used, the types of
    operations performed on the data structures, and the properties of the
    control strategies used. Principles of Artificial Intelligenceevolved
    from the author's courses and seminars at Stanford University and
    University of Massachusetts, Amherst, and is suitable for text use in
    a senior or graduate AI course, or for individual study."
}
```

———-

— **axiom.bib** —

```
@article{Nipk02,
  author = "Nipkow, Tobias",
  title = {{Structured Proofs in Isar/HOL}},
  journal = "LNCS",
  volume = "2646",
  year = "2002",
  pages = "259-278",
  abstract =
    "Isar is an extension of the theorem prover Isabelle with a language
    for writing human-readable structured proofs. This paper is an
    introduction to the basic constructs of this language.",
  paper = "Nipk02.pdf",
  keywords = "printed"
}
```

———-

1.41.15 O

— **axiom.bib** —

```
@misc{OCAM14,
  author = "unknown",
  title = {{The OCAML website}},
  link = "\url{http://ocaml.org}"
}
```

———-

— **ignore** —

```
\bibitem[Ollagnier 94]{Olla94} Ollagnier, Jean Moulin
  title = {{Algorithms and methods in differential algebra}},
  link = "\url{http://www.lix.polytechnique.fr/~moulin/papiers/atelier.pdf}",
  paper = "Olla94.pdf"
```

———-

— **ignore** —

```
\bibitem[Olver 10]{NIST10} Olver, Frank W.; Lozier, Daniel W.;
Boisvert, Ronald F.; Clark, Charles W. (ed)
  title = {{NIST Handbook of Mathematical Functions}},
(2010) Cambridge University Press ISBN 978-0-521-19225-5
```

———-

— **ignore** —

```
\bibitem[OpenM]{OpenM}.
  title = {{OpenMath Technical Overview}},
  link = "\url{http://www.openmath.org/overview/technical.html}",
```

———-

— **ignore** —

```
\bibitem[Ortega 70]{OR70} Ortega J M.; Rheinboldt W C.
  title = {{Iterative Solution of Nonlinear Equations in Several Variables}},
Academic Press. (1970)
```

———-

— **axiom.bib** —

```
@misc{Ostr1845,
```

```
  author = "Ostrogradsky. M.W.",
  title = {{De l'int\'{e}gration des fractions rationelles.}},
  journal = "Bulletin de la Classe Physico-Math\'{e}matiques de
    l'Acae\'{e}mie Imp\'{e}riale des Sciences de St. P\'{e}tersbourg,",
  volume = "IV",
  pages = "145-167,286-300",
  year = "1845"
}
```

———-

1.41.16 P

— **axiom.bib** —

```
@misc{Paga17,
  author = "Pagani, Kurt",
  title = {{trigonometric simplification}},
  year = "2017",
  link = "\url{https://groups.google.com/forum/#topic/fricas-devel/kteSoeR_Iyg}"
}
```

———-

— **ignore** —

```
\bibitem[Paige 75]{PS75} Paige C C.; Saunders M A.
  title = {{Solution of Sparse Indefinite Systems of Linear Equations}},
SIAM J. Numer. Anal. 12 617--629. (1975)
```

———-

— **ignore** —

```
\bibitem[Paige 82a]{PS82a} Paige C C.; Saunders M A.
  title = {{LSQR: An Algorithm for Sparse Linear Equations and
            Sparse Least-squares}},
ACM Trans. Math. Softw. 8 43--71. (1982)
```

———-

— **ignore** —

```
\bibitem[Paige 82b]{PS82b} Paige C C.; Saunders M A.
  title = {{ALGORITHM 583 LSQR: Sparse Linear Equations and
            Least-squares Problems}},
ACM Trans. Math. Softw. 8 195--209. (1982)
```

———-

— **ignore** —

```
\bibitem[Parlett 80]{Par80} Parlett B N.
  title = {{The Symmetric Eigenvalue Problem}},
Prentice-Hall. 1980
```

———-

— **ignore** —

```
\bibitem[Parnas 10]{PJ10} Parnas, David Lorge; Jin, Ying
  title = {{Defining the meaning of tabular mathematical expressions}},
Science of Computer Programming V75 No.11 Nov 2010 pp980-1000 Elesevier
```

———-

— **ignore** —

```
\bibitem[Parnas 95]{PM95} Parnas, David Lorge; Madey, Jan
  title = {{Functional Documents for Computer Systems}},
Science of Computer Programming V25 No.1 Oct 1995 pp41-61 Elesevier
```

———-

— **axiom.bib** —

```
@book{Paul81,
  author = "Paul, Richard",
  title = {{Robot Manipulators}},
  year = "1981",
  publisher = "MIT Press",
  isbn = "0-262-16082-X"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Paul89,
  author = "Paulson, Lawrence C. and Smith, Andrew W.",
  title = {{Logic Programming, Functional Programming, and
            Inductive Definitions}},
  booktitle = "Int. Workshop on Extensions of Logic Programming",
  publisher = "Springer",
  year = "1989",
  pages = "283-309",
  abstract =
    "The unification of logic and functional programming, like the Holy
    Grail, is sought by countless people [6, 14]. In reporting our
    attempt, we first discuss the motivation. We argue that logic
```

```
    programming is still immature, compared with functional programming,
    because few logic programs are both useful and pure.  Functions can
    help to purify logic programming, for they can eliminate certain uses
    of the cut and can express certMn negations positively.

    More generally, we suggest that the traditional paradigm -- logic
    programming as first-order logic -- is seriously out of step with
    practice. We offer an alternative paradigm. We view the logic program
    as an inductive definition of sets and relations. This view explains
    certain uses of Negation as Failure, and explains why most attempts to
    extend PROLO G to larger fragments of first-order logic have not been
    successful. It suggests a logic language with functions,
    incorporating equational unification.

    We have implemented a prototype of this language. It is very slow,
    but complete, and appear to be faster than some earlier
    implementations of narrowing.  Our experiments illustrate the
    programming style and shed light on the further development of such
    languages.",
  paper = "Paul89.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Paul90,
  author = "Paulson, Lawrence C.",
  title = {{A Formulation of the Simple Theory of Types (for
            Isabelle)}},
  journal = "LNCS",
  volume = "417",
  pages = "246-274",
  year = "1990",
  abstract =
    "Simple type theory is formulated for use with the generic theorem
    prover Isabelle. This requires explicit type inference rules. There
    are function, product, and subset types, which may be
    empty. Descriptions (the eta-operator) introduce the Axiom of
    Choice. Higher-order logic is obtained through reflection between
    formulae and terms of type bool. Recursive types and functions can be
    formally constructed. Isabelle proof procedures are described. The
    logic appears suitable for general mathematics as well as
    computational problems.",
  paper = "Paul90.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Pear56,
  author = "Pearcey, T.",
  title = {{Table of the Fresnel Integral}},
  publisher = "Cambridge University Press",
  year = "1956",
  algebra = "\newline\refto{package DFSFUN DoubleFloatSpecialFunctions}"
}
```

———-

— **ignore** —

```
\bibitem[Pereyra 79]{Per79} Pereyra V.
  title = {{PASVA3: An Adaptive Finite-Difference Fortran Program for First
           Order Nonlinear, Ordinary Boundary Problems}},
Codes for Boundary Value Problems in Ordinary Differential Equations.
Lecture Notes in Computer Science.
(ed B Childs, M Scott, J W Daniel, E Denman and P Nelson) 76
Springer-Verlag.  (1979)
```

———-

— **ignore** —

```
\bibitem[Peters 67a]{Pet67a} Peters G.
  title = {{NPL Algorithms Library}},
Document No. F2/03/A. (1967)
```

———-

— **ignore** —

```
\bibitem[Peters 67b]{Pet67b} Peters G.
  title = {{NPL Algorithms Library}},
Document No.F1/04/A (1967)
```

———-

— **ignore** —

```
\bibitem[Peters 70]{PW70} Peters G.; Wilkinson J H.
  title = {{The Least-squares Problem and Pseudo-inverses}},
Comput. J. 13 309--316. (1970)
```

———-

— **ignore** —

```
\bibitem[Peters 71]{PW71} Peters G.; Wilkinson J H.
  title = {{Practical Problems Arising in the Solution of Polynomial
```

```
          Equations}},
J. Inst. Maths Applics. 8 16--35. (1971)
```

———-

— **axiom.bib** —

```
@misc{Pfei12,
  author = "Pfeil, Greg",
  title = {{Common Lisp Type Hierarchy}},
  year = "2012",
  link = "\url{http://sellout.github.io/2012/03/03/common-lisp-type-hierarchy}"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Pfen89a,
  author = "Pfenning, Frank",
  title = {{Elf: A Language for Logic Definition and Verified
           Metaprogramming}},
  booktitle = "4th Symp. on Logic in Computer Science",
  pages = "313-322",
  isbn = "0-8186-1954-6",
  year = "1989",
  abstract =
    "We describe Elf, a metalanguage for proof manipulation environments
    that are independent of any particular logic system. Elf is
    intended for meta-programs such as theorem provers, proof
    transformers, or type inference programs for programming languages
    with complex type systems. Elf unifies logic definition (in the
    style of LF, the Edinburgh Logical Framework) with logic
    programming (in the style of $\lambda$Prolog). It achieves this
    unification by giving {\sl types} an operational interpretation,
    much the same way that Prolog gives certain formulas
    (Horn-clauses) an operational interpretation. Novel features of
    Elf include: (1) the Elf search process automatically constructs
    terms that can represent object-logic proofs, and thus a program
    need not construct them explicitly, (2( the partial correectness
    of meta-programs with respect to a given logic can be expressed
    and proved in Elf itself, and (3) Elf exploits Elliott's
    unification algorithm for a $\lambda$-calculus with dependent
    types.",
  paper = "Pfen89a.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Pfen92b,
  author = "Pfenning, Frank and Rohwedder, Ekkehard",
  title = {{Implementing the Meta-Theory of Deductive Systems}},
  journal = "Lecture Notes in Computer Science",
  volume = "607",
  pages = "771-775",
  year = "1992",
  abstract =
    "We exhibit a methodology for formulating and verifying meta-theorems
    about deductive systems in the Elf language, an implementation of the
    LF Logical Framework with an operational semantics in the spirit of
    logic programming.  It is based on the mechanical verification of
    properties of transformations between deductions, which relies on type
    reconstruction and schema-checking.  The latter is justified by
    induction principles for closed LF objects, which can be constructed
    over a given signature.  We illustrate our technique through several
    examples, the most extensive of which is an interpretation of
    classical logic in minimal logic through a continuation-passing-style
    transformation on proofs.",
  paper = "Pfen92b.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Pfen97,
  author = "Pfenning, Frank",
  title = {{Computation and Deduction}},
  year = "1997",
  link = "\url{http://www.cs.cmu.edu/~twelf/notes/cd.pdf}",
  paper = "Pfen97.pdf"
}
```

———-

— **axiom.bib** —

```
@phdthesis{Phil02,
  author = "Philippe, M. Trebuchet",
  title = {{Toward a fast and numerically stable algebraic equation solving}},
  comment = "Vers une resolution stable et rapide des equations algebriques",
  school = "l'Universite de Paris 6",
  year = "2002",
  month = "December",
  abstract =
    "Polynomial systems can be found in many industrial applications. They
    are also in the heart of effective algebraic geometry. A fundamental
    tool for studying them is the Groebner bases. The knowledge of this
    paricular base of the ideal generated by the polynomials composing the
    system allows us to compute in $A=K[x_1,\ldots,x_n]/I$, the quotient
    algebra, and this is necessary when we try to solve. Nevertheless,
```

```
    Groebner bases computations rely heavily on the introduction of
    monomial ordering. This introduces a certain rigidity in the
    computation and thus numerical instability. We propose a new algorithm
    that tries to remedy that problem. It generalises Groebner bases
    computation and is much less numerically instable. To do this, we
    decrease the requirement of monomial ordering, and use a new normal
    form criterion. We then give an algorithm and prove its termination
    and correctness when the input polynomial system is
    0-dimensional. After, we compare it with the previously known methods
    and show how it can be seen as a generalisation of them. Next, we
    detail how we implemented it in C++ using the Synaps library. We also
    describe the sparse matrix elimination algorithm we used in or
    program. Finally, we present some of the experiments we have done with
    our program in domains like computer vision, algorithmic geometry,
    robotics, or pharmacology.",
  paper = "Phil02.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Pierce 82]{Pie82} R.S. Pierce
  title = {{Associative Algebras}},
Graduate Texts in Mathematics 88
Springer-Verlag,  Heidelberg, 1982, ISBN 0-387-90693-2
```

———-

— **axiom.bib** —

```
@misc{Pier17,
  author = "Pierce, Benjamin",
  title = {{DeepSpec Summer School, Coq Intensive, Part 1 (July 13,2017)}},
  year = "2017",
  link = "\url{https://www.youtube.com/watch?v=jG61w5pOc2A}"
}
```

———-

— **ignore** —

```
\bibitem[Piessens 73]{Pie73} Piessens R.
  title = {{An Algorithm for Automatic Integration}},
Angewandte Informatik. 15 399--401. (1973)
```

———-

— **ignore** —

```
\bibitem[Piessens 74]{PMB74} Piessens R.; Mertens I.; Branders M.
```

```
  title = {{Integration of Functions having End-point Singularities}},
Angewandte Informatik. 16 65--68. (1974)
```

———-

— **ignore** —

```
\bibitem[Piessens 75]{PB75} Piessens R.; Branders M.
  title = {{Algorithm 002. Computation of Oscillating Integrals}},
J. Comput. Appl. Math. 1 153--164. (1975)
```

———-

— **ignore** —

```
\bibitem[Piessens 76]{PVRBM76} Piessens R.; Van Roy-Branders M.; Mertens I.
  title = {{The Automatic Evaluation of Cauchy Principal Value Integrals}},
Angewandte Informatik. 18 31--35. (1976)
```

———-

— **ignore** —

```
\bibitem[Piessens 83]{PDUK83} Piessens R.; De Doncker-Kapenga E.;
Uberhuber C.; Kahaner D.
  title = {{QUADPACK, A Subroutine Package for Automatic Integration}},
Springer-Verlag.(1983)
```

———-

— **axiom.bib** —

```
@book{Prie12,
  author = "Priest, Graham",
  title = {{An Introduction to Non-Classical Logic}},
  year = "2012",
  publisher = "Cambridge University Press"
}
```

———-

— **ignore** —

```
\bibitem[Polya 37]{Pol37} Polya, G.
  title = {{Kombinatorische Anzahlbestimmungen fur Gruppe}},
Graphen und chemische Verbindungen''
Acta Math. 68 (1937) 145-254.
```

———-

— **axiom.bib** —

```
@article{Pome80,
  author = "Pomerance, C. and Slefridge, J.L. and Wagstaff Jr., S.S.",
  title = {{The Pseudoprimes up to $25\cdot 10^9$}},
  journal = "Math. Comp.",
  volume = "35",
  pages = "1003-1026",
  year = "1980"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Popo09,
  author = "Popov, Nikolaj and Jebelean, Tudor",
  title = {{Functional Program Verification in Theorema Soundness and
           Completeness}},
  booktitle = "Proc. 15th Biennial Workshop on Programmiersprachen und
               Grundlagen der Programmierung KPS'09",
  year = "2009",
  pages = "221-229",
  link =
    "\url{http://www.risc.jku.at/publications/download/risc_3913/PopJeb.pdf}",
  abstract =
    "We present a method for verifying recursive functional programs. We
    define a Verification Condition Generator (VCG) which covers the most
    frequent types of recursive programs. These programs may operate on
    arbitrary domains. Soundness and Completeness of the VCG are proven on
    the meta level, and this provides a warranty that any system based on
    our results will be sound.",
  paper = "Popo09.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Popo09a,
  author = "Popov, Nikolaj and Jebelean, Tudor",
  title = {{A Complete Method for Algorithm Validation}},
  booktitle =
    "Proc. Workshop on Automated Math Theory Exploration AUTOMATHEO'09",
  pages = "21-25",
  year = "2009",
  link = "\url{http://www.risc.jku.at/publications/download/risc_3915/PopJeb-AUTOMATHEO.pdf}",
  abstract =
    "We present some novel ideas for proving total correctness of
    recursive functional programs and we discuss how they may be used for
    algorithm validation.  As usual, correctness (validation) is
    transformed into a set of first-order predicate logic
```

```
    formulaeverification conditions.  As a distinctive feature of our
    method, these formulae are not only sufficient, but also necessary
    for the correctness. We demonstrate our method on the Nevilles
    algorithm for polynomial interpolation and show how it may be
    validated automatically. In fact, even if a small part of the
    specification is missingin the literature this is often a case --
    the correctness cannot be proven. Furthermore, a relevant
    counterexample may be constructed automatically.",
  paper = "Popo99a.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Powell 70]{Pow70} Powell M J D.
  title = {{A Hybrid Method for Nonlinear Algebraic Equations}},
Numerical Methods for Nonlinear Algebraic Equations.
(ed P Rabinowitz) Gordon and Breach. (1970)
```

———-

— **ignore** —

```
\bibitem[Powell 74]{Pow74} Powell M J D.
  title = {{Introduction to Constrained Optimization}},
Numerical Methods for Constrained Optimization.
(ed P E Gill and W Murray) Academic Press. pp1-28. 1974
```

———-

— **ignore** —

```
\bibitem[Powell 83]{Pow83} Powell M J D.
  title = {{Variable Metric Methods in Constrained Optimization}},
Mathematical Programming: The State of the Art.
(ed A Bachem, M Groetschel and B Korte) Springer-Verlag. pp288--311. 1983
```

———-

— **axiom.bib** —

```
@inproceedings{Prat73,
  author = "Pratt, Vaughan R.",
  title = {{Top Down Operator Precedence}},
  booktitle = "Proc. 1st annual ACM SIGACT-SIGPLAN Symposium on Principles
               of Programming Languages",
  series = "POPL'73",
  pages = "41-51",
  year = "1973",
  link = "\url{http://hall.org.ua/halls/wizzard/pdf/Vaughan.Pratt.TDOP.pdf}",
```

```
  paper = "Prat73.pdf",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@book{Pres07,
  author = "Press, William H. and Teukolsky, Saul A. and
            Vetterling, William T. and Flannery, Brian P.",
  title = {{Numerical Recipes (3rd Edition)}},
  year = "2007",
  publisher = "Cambridge University Press",
  isbn = "978-0-521-88068-8",
}
```

———-

— **ignore** —

```
\bibitem[Pryce 77]{PH77} Pryce J D.; Hargrave B A.
  title = {{The Scale Pruefer Method for one-parameter and multi-parameter
            eigenvalue problems in ODEs}},
Inst. Math. Appl., Numerical Analysis Newsletter. 1(3)  (1977)
```

———-

— **ignore** —

```
\bibitem[Pryce 81]{Pry81} Pryce J D.
  title = {{Two codes for Sturm-Liouville problems}},
Technical Report CS-81-01. Dept of Computer Science, Bristol University (1981)
```

———-

— **ignore** —

```
\bibitem[Pryce 86]{Pry86} Pryce J D.
  title = {{Error Estimation for Phase-function Shooting Methods fo}},
Sturm-Liouville Problems''
J. Num. Anal. 6 103--123. (1986)
```

———-

— **axiom.bib** —

```
@misc{Puff09,
  author = "Puffinware LLC",
  title = {{Singular Value Decomposition (SVD) Tutorial}},
```

```
  link = "\url{http://www.puffinwarellc.com/p3a.htm}"
}
```

———-

1.41.17 Q

— **ignore** —

```
\bibitem[Quintana-Orti 06]{QG06} Quintana-Orti, Gregorio;
van de Geijn, Robert
  title = {{Improving the performance of reduction to Hessenberg form}},
ACM Transactions on Mathematical Software, 32(2):180-194, June 2006.
```

———-

1.41.18 R

— **axiom.bib** —

```
@article{Rabe13,
  author = "Rabe,Florian and Kohlhase, Michael",
  title = {{A Scalable Module System}},
  journal = "Information and Computation",
  volume = "230",
  pages = "1-54",
  year = "2013",
  abstract =
   "Symbolic and logic computation systems ranging from computer algebra
   systems to theorem provers are finding their way into science,
   technology, mathematics and engineering. But such systems rely on
   explicitly or implicitly represented mathematical knowledge that needs
   to be managed to use such systems effectively.

   While mathematical knowledge management (MKM) ‘‘in the small’’ is
   well-studied, scaling up to large, highly interconnected corpora
   remains difficult. We hold that in order to realize MKM in the
   large, we need representation languages and software architectures
   that are designed systematically with large-scale processing in mind.

   Therefore, we have designed and implemented the Mmt language  a
   module system for mathematical theories. Mmt is designed as the
   simplest possible language that combines a module system, a
   foundationally uncommitted formal semantics, and web-scalable
   implementations. Due to a careful choice of representational
   primitives, Mmt allows us to integrate existing representation
   languages for formal mathematical knowledge in a simple, scalable
   formalism. In particular, Mmt abstracts from the underlying
   mathematical and logical foundations so that it can serve as a
   standardized representation format for a formal digital
```

```
   library. Moreover, Mmt systematically separates logic-dependent and
   logic-independent concerns so that it can serve as an interface layer
   between computation systems and MKM systems.",
  paper = "Rabe13.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Rabe13a,
  author = "Rabe, Florian",
  title = {{The MMT API: A Generic MKM System}},
  journal = "LNCS",
  volume = "7961",
  pages = "339-343",
  year = "2013",
  abstract =
   "The Mmt language has been developed as a scalable representation and
   interchange language for formal mathematical knowledge. It permits
   natural representations of the syntax and semantics of virtually all
   declarative languages while making Mmt-based MKM services easy to
   implement. It is foundationally unconstrained and can be instantiated
   with specific formal languages.

   The Mmt API implements the Mmt language along with multiple backends
   for persistent storage and frontends for machine and user
   access. Moreover, it implements a wide variety of Mmt-based knowledge
   management services. The API and all services are generic and can be
   applied to any language represented in Mmt. A plugin interface permits
   injecting syntactic and semantic idiosyncrasies of individual formal
   languages.",
  paper = "Rabe13a.pdf",
  keywrods = "printed"
}
```

———-

— **axiom.bib** —

```
@article{Rabe16,
  author = "Rabe, Florian",
  title = {{The Future of Logic: Foundation-Independence}},
  journal = "Logica Universalis",
  volume = "10",
  number = "1",
  pages = "1-20",
  year = "2016",
  abstract =
   "Throughout the twentieth century, the automation of formal logics in
   computers has created unprecedented potential for practical
   applications of logicmost prominently the mechanical verification of
```

```
   mathematics and software. But the high cost of these applications
   makes them infeasible but for a few flagship projects, and even those
   are negligible compared to the ever-rising needs for verification. One
   of the biggest challenges in the future of logic will be to enable
   applications at much larger scales and simultaneously at much lower
   costs. This will require a far more efficient allocation of
   resources. Wherever possible, theoretical and practical results must
   be formulated generically so that they can be instantiated to
   arbitrary logics; this will allow reusing results in the face of
   todays multitude of application-oriented and therefore diverging
   logical systems. Moreover, the software engineering problems
   concerning automation support must be decoupled from the theoretical
   problems of designing logics and calculi; this will allow researchers
   outside or at the fringe of logic to contribute scalable
   logic-independent tools. Anticipating these needs, the author has
   developed the Mmt framework. It offers a modern approach towards
   defining, analyzing, implementing, and applying logics that focuses on
   modular design and logic-independent results. This paper summarizes
   the ideas behind and the results about Mmt. It focuses on showing how
   Mmt. provides a theoretical and practical framework for the future of
   logic.",
  paper = "Rabe16.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Rabe17,
  author = "Rabe, Florian",
  title = {{How to Identify, Translate, and Combine Logics?}},
  journal = "J. of Logic and Computation",
  volume = "27",
  number = "6",
  pages = "1753-1798",
  year = "2017",
  abstract =
   "We give general definitions of logical frameworks and
   logics. Examples include the logical frameworks LF and Isabelle and
   the logics represented in them. We apply this to give general
   definitions for equivalence of logics, translation between logics and
   combination of logics. We also establish general criteria for the
   soundness and completeness of these. Our key messages are that the
   syntax and proof systems of logics are theories; that both semantics
   and translations are theory morphisms; and that combinations are
   colimits. Our approach is based on the Mmt language, which lets us
   combine formalist declarative representations (and thus the associated
   tool support) with abstract categorical conceptualizations.",
  paper = "Rabe17.pdf",
  keywords = "printed, DONE"
}
```

———-

— **axiom.bib** —

```
@article{Rabi80,
  author = "Rabin, M.O.",
  title = {{Probabilistic Algorithm for Testing Primality}},
  journal = "J. Number Theory",
  volume = "12",
  pages = "128-138",
  year = "1980"
}
```

———-

— **ignore** —

```
\bibitem[Rabinowitz 70]{Rab70} Rabinowitz P.
  title = {{Numerical Methods for Nonlinear Algebraic Equations}},
Gordon and Breach. (1970)
```

———-

— **ignore** —

```
\bibitem[Ralston 65]{Ral65} Ralston A.
  title = {{A First Course in Numerical Analysis}},
McGraw-Hill. 87--90. (1965)
```

———-

— **ignore** —

```
\bibitem[Ramakrishnan 03]{Ram03} Ramakrishnan, Maya
  title = {{A Gentle Introduction to Lyapunov Functions}},
ORSUM August 2003
  link = "\url{http://www.or.ms.unimelb.edu.au/handouts/lyaptalk.1.pdf}",
```

———-

— **ignore** —

```
\bibitem[Ramsey 03]{Ra03} Ramsey, Norman
  title = {{Noweb--A Simple, Extensible Tool for Literate Programming}},
  link = "\url{http://www.eecs.harvard.edu/~nr/noweb}",
```

———-

— **axiom.bib** —

```
@phdthesis{Rave91,
  author = "Ravenscroft, Robert Andrews Jr.",
  title = {{Generating Function Algorithms for Symbolic Computation}},
  school = "Brown University",
  year = "1991",
  file = "Rave91.pdf",
  abstract =
    "Various algebraic models have been used to implement symbolic
    procedures for solving summations and recurrences. Surprisingly, one
    model that has received little attention in the development of
    symbolic algorithms is generating functions. Generating function
    methods for discrete mathematics provide an ad hoc collection of
    techniques for solving summations and recurrences. In this work we
    consider their systematic application to symbolic computation.

    To demonstrate the feasibility and capabilities of generating
    functions as an algebraic model of computation, we develop and
    implement algorithms for manipulating generating functions and the
    sequences that they encode. In particular, we concentrate on
    algorithms for solving summations and recurrences. Techniques are
    developed to handle special functions and hybrid terms, which are
    terms involving two or more factors.

    Given various classes of summations and recurrences we develop a
    theory to characterize the form of their solution. This allows us to
    demonstrate the ability of our algorithms to solve problems of these
    classes. This knowledge also assists us in developing algorithms that
    know where to look for a solution rather than doing an ad hoc search
    for a solution."

}
```

———-

— **axiom.bib** —

```
@book{Redf98,
  author = "Redfern, D.",
  title = {{The Maple Handbook: Maple V Release 5}},
  publisher = "Springer",
  year = "1998"
}
```

———-

— **axiom.bib** —

```
@article{Redf27,
  author = "Redfield, J. Howard",
  title = {{The Theory of Group-Reduced Distributions}},
  journal = "American J. Math.",
  volume = "49",
```

```
  number = "3",
  year = "1927",
  pages = "433-455"
}
```

———-

— **ignore** —

```
\bibitem[Reinsch 67]{Rei67} Reinsch C H.
  title = {{Smoothing by Spline Functions}},
Num. Math. 10 177--183. (1967)
```

———-

— **ignore** —

```
\bibitem[Renka 84]{Ren84} Renka R L.
  title = {{Algorithm 624: Triangulation and Interpolation of Arbitrarily
            Distributed Points in the Plane}},
ACM Trans. Math. Softw. 10 440--442. (1984)
```

———-

— **ignore** —

```
\bibitem[Renka 84]{RC84} Renka R L.; Cline A K.
  title = {{A Triangle-based C Interpolation Method}},
Rocky Mountain J. Math. 14 223--237. (1984)
```

———-

— **ignore** —

```
\bibitem[Reutenauer 93]{Re93} Reutenauer, Christophe
  title = {{Free Lie Algebras}},
Oxford University Press, June 1993 ISBN 0198536798
```

———-

— **ignore** —

```
\bibitem[Reznick 93]{Rezn93} Reznick, Bruce
  title = {{An Inequality for Products of Polynomials}},
Proc. AMS Vol 117 No 4 April 1993
  paper = "Rezn93.pdf"
```

———-

— **ignore** —

```
\bibitem[Rich xx]{Rixx} Rich, A.D.; Jeffrey, D.J.
  title = {{Crafting a Repository of Knowledge Based on Transformation}},
  link = "\url{http://www.apmaths.uwo.ca/~djeffrey/Offprints/IntegrationRules.pdf}",
  abstract = "
    We describe the development of a repository of mathematical knowledge
    based on transformation rules. The specific mathematical problem is
    indefinite integration. It is important that the repository be not
    confused with a look-up table. The database of transformation rules is
    at present encoded in Mathematica, but this is only one convenient
    form of the repository, and it could be readily translated into other
    formats. The principles upon which the set of rules is compiled is
    described. One important principle is minimality. The benefits of the
    approach are illustrated with examples, and with the results of
    comparisons with other approaches.",
  paper = "Rixx.pdf"
```

———-

— **ignore** —

```
\bibitem[Rich 10]{Ri10} Rich, Albert D.
  title = {{Rule-based Mathematics}},
  link = "\url{http://www.apmaths.uwo.ca/~arich}",
```

———-

— **axiom.bib** —

```
@article{Rich69,
  author = "Richardson, Daniel",
  title = {{Some Undecidable Problems involving Elementar Functions of
            a Real Variable}},
  journal = "J. Symbolic Logic",
  volume = "33",
  number = "4",
  year = "1969",
  pages = "514-520",
  abstract =
    "Let $E$ be a set of expressions representing real, single valued,
    partially defined functions of one real variable. $E^*$ will be the
    set of functions represented by expressions in $E$. If $A$ is an
    expression in $E$, $A(x)$ is the function denoted by $AA$. It is
    assumed that $D^*$ contains the identity function and the rational
    numbers as constant functions and that $E^*$ is closed under addition,
    subtraction, multiplication, and composition"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Rich94,
  author = "Richardson, Dan and Fitch, John P.",
  title = {{The identity problem for elementary functions and constants}},
  booktitle = "ACM Proc. of ISSAC 94",
  pages = "285-290",
  isbn = "0-89791-638-7",
  year = "1994",
  abstract =
    "A solution for a version of the identify problem is proposed for a
    class of functions including the elementary functions. Given f(x),
    g(x), defined at some point $\beta$ we decide whether or not
    $f(x) \equiv g(x)$
    in some neighbourhood of $\beta$. This problem is first reduced to a
    problem about zero equivalence of elementary constants. Then a semi
    algorithm is given to solve the elementary constant problem. This semi
    algorithm is guaranteed to give the correct answer whenever it
    terminates, and it terminates unless the problem being considered
    contains a counterexample to Schanuel's conjecture.",
  paper = "Rich94.pdf"

}
```

————-

— **ignore** —

```
\bibitem[Richtmyer 67]{RM67} Richtmyer R D.; Morton K W.
  title = {{Difference Methods for Initial-value Problems}},
Interscience (2nd Edition).  (1967)
```

————-

— **ignore** —

```
\bibitem[Rioboo 92]{REF-Rio92} Rioboo, R.
  title = {{Real algebraic closure of an ordered field, implementation in
           Axiom}},
In Wang [Wan92], pp206-215 ISBN 0-89791-489-9 (soft cover)
In proceedings of the ISSAC'92 Conference, Berkeley 1992 pp. 206-215.
0-89791-490-2 (hard cover) LCCN QA76.95.I59 1992
```

————-

— **ignore** —

```
\bibitem[Rioboo 96]{Rio96} Rioboo, R.
  title = {{Generic computation of the real closure of an ordered field}},
In Mathematics and Computers in Simulation Volume 42, Issue 4-6,
November 1996.
```

———-

— **axiom.bib** —

```
@article{Ritt50,
  author = "Ritt, Joseph Fels",
  title = {{Differential Algebra}},
  journal = "AMS Colloquium Publications",
  year = "1950",
  volume = "33",
  isbn = "978-0-8218-4638-4",
  algebra = "\newline\refto{category DVARCAT DifferentialVariableCategory}",
  paper = "Ritt50.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Rote 01]{Rote01} Rote, G\"unter
  title = {{Division-free algorithms for the determinant and the Pfaffian}},
in Computational Discrete Mathematics ISBN 3-540-42775-9 pp119-135
  link = "\url{http://page.mi.fu-berlin.de/rote/Papers/pdf/Division-free+algorithms.pdf}",
```

———-

— **axiom.bib** —

```
@InProceedings{Rube06,
  author = "Rubey, Martin",
  title = {{Extented rate, more GFUN}},
  booktitle = "4th Colloquium on mathematics and computer science",
  series = "DMTCS",
  year = "2006",
  location = "Nancy, France",
  pages = "431-434",
  link = "\url{http://mathinfo06.iecn.u-nancy.fr/papers/dmAG431-434.pdf}",
  algebra =
    "\newline\refto{package GUESS Guess}
     \newline\refto{package GUESSAN GuessAlgebraicNumber}
     \newline\refto{package GUESSF GuessFinite}
     \newline\refto{package GUESSF1 GuessFiniteFunctions}
     \newline\refto{package GUESSINT GuessInteger}
     \newline\refto{package GUESSP GuessPolynomial}
     \newline\refto{package GUESSUP GuessUnivariatePolynomial}",
  abstract =
    "We present a software package that guesses formulas from sequences
    of, for example, rational numbers or rational functions, given the
    first few terms. Thereby we extend and complement C. Krattenthaler's
    program RATE [RATE: a Mathematics guessing machine] and the relevant
```

```
    parts of B. Salvy and P. Zimmermann's GFUN.",
  paper = "Rube06.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Rubey 07]{Rub07} Rubey, Martin
  title = {{Formula Guessing with Axiom}},
April 2007
```

———-

— **axiom.bib** —

```
@article{Rump88,
  author = "Rump, Siegfried M.",
  title = {{Algebraic Computation, Numerical Computation and Verified
            Inclusions}},
  journal = "LNCS",
  volume = "296",
  pages = "177-197",
  year = "1988",
  abstract =
    "The three different types of computation -- the algebraic
    manipulation, the numerical computation and the computation of
    verified results -- are aiming on different problems and deliver
    qualitatively different results, each method having its specific
    advantages for specific classes of problems. The following remarks
    give some thoughts on possible combinations of all three methods to
    obtain algorithms benefitting from the specific strength of either
    method.",
  paper = "Rump88.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Rush00,
  author = "Rushby, John",
  title = {{Disappearing Formal Methods}},
  booktitle = "High Assurance Systems Engineering, 5th Int. Symp.",
  pages = "95-96",
  year = "2000",
  publisher = "ACM",
  paper = "Rush00.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Rutishauser 69]{Rut69} Rutishauser H.
  title = {{Computational aspects of F L Bauer's simultaneous iteration
           method}},
Num. Math. 13 4--13.  (1969)
```

———-

— **ignore** —

```
\bibitem[Rutishauser 70]{Rut70} Rutishauser H.
  title = {{Simultaneous iteration method for symmetric matrices}},
Num. Math. 16 205--223. (1970)
```

———-

1.41.19 S

— **axiom.bib** —

```
@book{Saun79,
  author = "MacLane, Saunders and Birkhoff, Garrett",
  title = {{Algebra, Second Edition}},
  publisher = "MacMillan",
  year = "1979",
  algebra = "\newline\refto{category GRMOD GradedModule}"
}
```

———-

— **axiom.bib** —

```
@misc{Salo16,
  author = "Salomone, Matthew",
  title = {{Exploring Abstract Algebra II}},
  year = "2016",
  link = "\url{https://www.youtube.com/watch?v=RNpdUG_yH_s\&list=PLLOATV5XYF8DTGAPKRPtYa4E8rOLcw88y\&index=1}"
}
```

———-

— **axiom.bib** —

```
@article{Scha61,
  author = "Schafer, R.D.",
  title = {{An Introduction to Nonassociative Algebras}},
```

```
  year = "1961",
  algebra = "\newline\refto{category NARNG NonAssociativeRng}",
  link = "\url{http://www.gutenberg.org/ebooks/25156}",
  journal = "Advanced Subject Matter Institute",
  abstract =
    "These are notes for my lectures in July, 1961, at the Advanced
    Subject Matter Institute in Algebra which was held at Oklahoma State
    University in the summer of 1961.

    Students at the Institute were provided with reprints of my paper,
    {\sl Structure and representation of nonassociate algebras} (Bulletin
    of the American Mathematical Society, vol. 61 (1955), pp469-484),
    together with copies of a selective bibliography of more recent papers
    on non-associative algebras. These notes supplement the 1955 Bulletin
    article, bringing the statements there up to date and providing
    detailed proofs of a selected group of theorems. The proofs illustrate
    a number of important techniques used in the study of nonassociative
    algebras.",
  paper = "Scha61.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Scha86,
  author = "Schaffert, C. and Cooper, T.",
  title = {{An Introduction to Trellis/Owl}},
  journal = "SIGPLAN Notices",
  volume = "21",
  number = "11",
  publisher = "ACM",
  year = "1986",
  pages = "9-16"
}
```

———-

— **axiom.bib** —

```
@book{Scha66,
  author = "Schafer, R.D.",
  title = {{An Introduction to Nonassociative Algebras}},
  year = "1966",
  publisher = "Academic Press, New York",
  algebra =
    "\newline\refto{category NARNG NonAssociativeRng}
     \newline\refto{category NASRING NonAssociativeRing}
     \newline\refto{domain ALGSC AlgebraGivenByStructuralConstants}"
}
```

———-

— **axiom.bib** —

```
@book{Scha10,
  author = "Schafer, R.D.",
  title = {{An Introduction to Nonassociative Algebras}},
  year = "2010",
  publisher = "Benediction Classics",
  algebra = "\newline\refto{category NARNG NonAssociativeRng}",
  isbn = "978-1849025904",
  abstract =
    "Concise study presents in a short space some of the important ideas
    and results in the theory of non-associative algebras, with particular
    emphasis on alternative and (commutative) Jordan algebras. Written as
    an introduction for graduate students and other mathematicians meeting
    the subject for the first time."
}
```

———-

— **axiom.bib** —

```
@book{Sche01,
  author = "Schelter, William F.",
  title = {{Maxima Manual Version 5.41.0}},
  year = "2001",
  publisher = "Sourceforge",
  paper = "Sche01.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Schr09,
  author = "Schreiner, Wolfgang",
  title = {{How to Write Postconditions with Nultiple Cases}},
  year = "2009",
  institution = "RISC Linz",
  abstract =
    "We investigate and compare the two major styles of writing program
    function postconditions with multiple cases: as conjunctions of
    implications or as disjunctions of conjunctions. We show that both
    styles not only have different syntax but also different semantics and
    pragmatics and give recommendations for their use.",
  paper = "Schr09.pdf"
}
```

———-

— **axiom.bib** —

```
@inproceedings{Schr00,
  author = "Schreiner, Wolfgang and Danielczyk-Landerl, Werner and
            Marin, Mircea and Stocher, Wolfgang",
  title = {{A Generic Programming Environment for High-Performance
           Mathematical Libraries}},
  booktitle = "Int. Seminar on Generic Programming",
  publisher = "Springer-Verlag",
  year = "2000",
  pages = "256-268",
  isbn = "3-540-41090-2",
  abstract =
    "We report on a programming environment for the development of
    generic mathematical libraries based on functors (parameterized
    modules) that have rigorously specified but very abstract
    interfaces. We focus on the combination of the functor-based
    programming style with software engineering principles in large
    development projects. The generated target code is highly efficient
    and can be easily embedded into foreign application environments.",
  paper = "Schr00.pdf"
}
```

———-

— **axiom.bib** —

```
@techreport{Schr14,
  author = "Schreiner, Wolfgang",
  title = {{Some Lessons Learned on Writing Predicate Logic Proofs in
           Isabelle/Isar}},
  year = "2014",
  institution = "RISC Linz",
  abstract =
    "We describe our experience with the use of the proving assistant
    Isabelle and its proof development language Isar for formulating and
    proving formal mathematical statements. Our focus is on how to use
    classical predicate logic and well established proof principles for
    this purpose, bypassing Isabelles meta-logic and related technical
    aspects as much as possible.  By a small experiment on the proof of
    (part of a) verification condition for a program, we were able to
    identify a number of important patterns that arise in such proofs
    yielding to a workflow with which we feel personally comfortable; the
    resulting guidelines may serve as a starting point for a the
    application of Isabelle / Isar for the average mathematical user
    (i.e, a mathematical user who is not interested in Isabelle / Isar per
    se but just wants to use it as a tool for computer-supported formal
    theory development).",
  paper = "Schr14.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Schoenberg 53]{SW53} Schoenberg I J.; Whitney A.
  title = {{On Polya Frequency Functions III}},
Trans. Amer. Math. Soc. 74  246--259. (1953)
```

———-

— **ignore** —

```
\bibitem[Schoenhage 82]{Sch82} Schoenhage, A.
  title = {{The fundamental theorem of algebra in terms of computational
           complexity}},
preliminary report, Univ. Tuebingen, 1982
```

———-

— **ignore** —

```
\bibitem[Schonfelder 76]{Sch76} Schonfelder J L.
  title = {{The Production of Special Function Routines for a
           Multi-Machine Library}},
Software Practice and Experience. 6(1) (1976)
```

———-

— **axiom.bib** —

```
@article{Schw85,
  author = "Schwarz, Fritz",
  title = {{An Algorithm for Determining Polynomial First Integrals of
            Autonomous Systems of Ordinary Differential Equations}},
  journal = "J. Symbolic Computation",
  volume = "1",
  year = "1985",
  pages = "229-233",
  paper = "Schw85.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Schw88,
  author = "Schwarz, Fritz",
  title = {{Symmetries of Differential Equations: From Sophus Lie to
            Computer Algebra}},
  journal = "SIAM Review",
  volume = "30",
  number = "3",
  year = "1988",
  abstract =
    "The topic of this article is the symmetry analysis of
```

```
    differential equations and the applications of computer algebra to
    th extensive analytical calculations which are usually involved in
    it. The whole area naturally decomposes into two parts depending
    on whether ordinary or partial differential equations are
    considered. We show how a symmetry may be applied to lower the
    order of an ordinary differential equation or to obtain similarity
    solutions of partial differential equations. The computer algebra
    packages SODE and SPDE, respectively, which have been developed to
    perform almost all algebraic manipulations necessary to determine
    the symmetry group of a given differential equation, are
    presented. Futhermore it is argued that the application of
    computer algebra systems has qualitatively changed this area of
    applied mathematics",
  keywords = "axiomref, printed"
}
```

———-

— **axiom.bib** —

```
@article{SCSCP,
  author = "The SCSCP development team",
  title = {{Symbolic Computation Software Composability Protocol}},
  journal = "Communications in Computer Algebra",
  volume = "44",
  number = "4",
  link = "\url{https://gap-packages.github.io/scscp/}",
  year = "2010",
  paper = "SCSCP.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@book{Segg93,
  author = "{von Seggern}, David Henry",
  title = {{CRC Standard Curves and Surfaces}},
  publisher = "CRC Press",
  year = "1993",
  isbn = "0-8493-0196-3"
}
```

———-

— **ignore** —

```
\bibitem[Seiler 95a]{Sei95a} Seiler, W.M.; Calmet, J.
  title = {{JET -- An Axiom Environment for Geometric Computations with
           Differential Equations}},
```

```
  paper = "Sei95a.pdf",
  keywords = "axiomref"
```

———-

— **ignore** —

```
\bibitem[Shepard 68]{She68} Shepard D.
  title = {{A Two-dimensional Interpolation Function for Irregularly
           Spaced Data}},
Proc. 23rd Nat. Conf. ACM. Brandon/Systems Press Inc.,
Princeton. 517--523. 1968
```

———-

— **ignore** —

```
\bibitem[Shirayanagi 96]{Shir96} Shirayanagi, Kiyoshi
  title = {{Floating point Gr\"obner bases}},
Mathematics and Computers in Simulation 42 pp 509-528 (1996)
  abstract = "
    Bracket coefficients for polynomials are introduced. These are like
    specific precision floating point numbers together with error
    terms. Working in terms of bracket coefficients, an algorithm that
    computes a Gr{\"o}bner basis with floating point coefficients is
    presented, and a new criterion for determining whether a bracket
    coefficient is zero is proposed. Given a finite set $F$ of polynomials
    with real coefficients, let $G_\mu$ be the result of the algorithm for
    $F$ and a precision $\mu$, and $G$ be a true Gr{\"o}bner basis of
    $F$. Then, as $\mu$ approaches infinity, $G_\mu$ converges to $G$
    coefficientwise. Moreover, there is a precision $M$ such that if
    $\mu \ge M$, then the sets of monomials with non-zero coefficients of
    $G_\mu$ and $G$ are exactly the same. The practical usefulness of the
    algorithm is suggested by experimental results.",
  paper = "Shir96.pdf"
```

———-

— **axiom.bib** —

```
@misc{Simo97,
  author = "Simon, Barry",
  title = {{The PC Is Now Axiomatic}},
  publisher = "PC Mag",
  year = "1997",
  month = "March",
  day = "25",
  keywords = "axiomref"
}
```

———-

— **ignore** —

```
\bibitem[Singer 89]{Sing89} Singer, M.F.
‘‘Formal Solutions of Differential Equations’’
J. Symbolic COmputation 10, No.1 59-94 (1990)
  abstract = "
    We give a survey of some methods for finding formal solutions of
    differential equations. These include methods for finding power series
    solutions, elementary and liouvillian solutions, first integrals, Lie
    theoretic methods, transform methods, asymptotic methods. A brief
    discussion of difference equations is also included.",
  paper = "Sing89.pdf",
  keywords = "survey"
```

———-

— **ignore** —

```
\bibitem[Sit 92]{REF-Sit92} Sit, William
  title = {{An Algorithm for Parametric Linear Systems}},
J. Sym. Comp., April 1992
```

———-

— **ignore** —

```
\bibitem[Smith 67]{Smi67} Smith B T.
  title = {{ZERPOL: A Zero Finding Algorithm for Polynomials Using
           Laguerre’s Method}},
Technical Report. Department of Computer Science, University of Toronto,
Canada.  (1967)
```

———-

— **ignore** —

```
\bibitem[Smith 85]{Smi85} Smith G D.
  title = {{Numerical Solution of Partial Differential Equations:
           Finite Difference Methods}},
Oxford University Press (3rd Edition). (1985)
```

———-

— **ignore** —

```
\bibitem[Sobol 74]{Sob74} Sobol I M.
  title = {{The Monte Carlo Method}},
The University of Chicago Press. 1974
```

———-

— **axiom.bib** —

```
@misc{Sorg96,
  author = "Sorge, Volker",
  title = {{Integration eines Computeralgebrasystems in eine logische
            Beweisumgebung }},
  school = "Univ. des Saarlandes",
  year = "1996",
  comment = "Master's thesis"
}
```

———-

— **axiom.bib** —

```
@article{Sorg00,
  author = "Sorge, Volker",
  title = {{Non-trivial Symbolic Computations in Proof Planning}},
  journal = "LNCS",
  volume = "1794",
  pages = "121-135",
  year = "2000",
  abstract =
    "We discuss a pragmatic approach to integrate computer algebra into
    proof planning. It is based on the idea to separate computation and
    verification and can thereby exploit the fact that many elaborate
    symbolic computations are trivially checked. In proof planning the
    separation is realized by using a powerful computer algebra system
    during the planning process to do non-trivial symbolic
    computations. Results of these computations are checked during the
    refinement of a proof plan to a calculus level proof using a small,
    self-implemented system that gives us protocol information on its
    calculation. This protocol can be easily expanded into a checkable
    low-level calculus proof ensuring the correctness of the
    computation. We demonstrate our approach with the concrete
    implementation in the $\Omega$MEGA system.",
  paper = "Sorg00.pdf",
  keywords = "CAS-Proof, printed, DONE"
}
```

———-

— **axiom.bib** —

```
@book{Stee90,
  author = "Steele, Guy L.",
  title = {{Common Lisp, The Language}},
  year = "1990",
  link = "\url{http://daly.axiom-developer.org/clm.pdf}",
  publisher = "Digital Equipment Corporation",
  isbn = "1-555558-041-6",
  algebra = "\newline\refto{package DFSFUN DoubleFloatSpecialFunctions}",
```

```
  paper = "Stee90.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Stur93,
  author = "Sturmfels, Bernd.",
  title = {{Algorithms in Invariant Theory}},
  year = "1993",
  publisher = "Springer",
  paper = "Stur93.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Stic93,
  author = "Stichtenoth, H.",
  title = {{Algebraic function fields and codes}},
  publisher = "Springer-Verlag",
  year = "1993"
}
```

———-

— **ignore** —

```
\bibitem[Stroud 66]{SS66} Stroud A H.; Secrest D.
  title = {{Gaussian Quadrature Formulas}},
Prentice-Hall. (1966)
```

———-

— **ignore** —

```
\bibitem[Stroud 71]{Str71} Stroud A H.
  title = {{Approximate Calculation of Multiple Integrals}},
Prentice-Hall 1971
```

———-

— **axiom.bib** —

```
@article{Supp77,
  author = "Suppes, Patrick and Smith, Robert and Beard, Marian",
  title = {{University-Level Computer-Assisted Instruction at Stanford: 1975}},
  journal = "Instructional Science",
```

```
  volume = "6",
  year = "1977",
  pages = "151-185",
  abstract =
    "This article provides an overview of current work on university-level
    computer-assisted instruction at Stanford University.  Brief
    descriptions are given of the main areas of current interest.  The
    main emphasis is on the courses now being used: Introduction to Logic,
    Axiomatic Set Theory, Old Church Slavonic, History of the Russian
    Literary Language, Introduction to Bulgarian, Introduction to BASIC,
    Introduction to LISP, and various courses in music.",
  paper = "Supp77.pdf"
}
```

———-

— axiom.bib —

```
@article{Supp89,
  author = "Suppes, P. and Takahashi, S.",
  title = {{An Interactive Calculus Theorem-prover for Continuity
            Properites}},
  journal = "J. Symbolic Computation",
  volume = "7",
  number = "6",
  year = "1989",
  pages = "573-590",
  abstract =
    "he work reported concerns the development of an interactive
    theorem-prover for the foundationsof the differential and integral
    calculus. The main tools are a resolution theorem-prover VERIFY,
    previously developed for interactive proofs in set theory, and the
    symbolic computation program REDUCE. The use of REDUCE in a
    theorem-proving context is described in detail. Sample proofs are
    given with data on computation time per step on an IBM-4381.",
  paper = "Supp89.pdf",
  keywords = "printed"
}
```

———-

— ignore —

```
\bibitem[Swarztrauber 79]{SS79} Swarztrauber P N.; Sweet R A.
  title = {{Efficient Fortran Subprograms for the Solution of Separable
            Elliptic Partial Differential Equations}},
ACM Trans. Math. Softw. 5 352--364. (1979)
```

———-

— ignore —

```
\bibitem[Swarztrauber 84]{SS84} Swarztrauber P N.
  title = {{Fast Poisson Solvers}},
Studies in Numerical Analysis. (ed G H Golub)
Mathematical Association of America. (1984)
```

———-

— **axiom.bib** —

```
@techreport{Swee86,
  author = "Sweedler, Moss E.",
  title = {{Typing in Scratchpad II}},
  institution = "IBM Research",
  year = "1986",
  month = "January",
  type = "Scratchpad II Newsletter",
  volume = "1",
  number = "2",
}
```

———-

1.41.20 T

— **axiom.bib** —

```
@book{Tait1890,
  author = "Tait, P.G.",
  title = {{An Elementary Treatise on Quaternions}},
  publisher = "C.J. Clay and Sons, Cambridge University Press Warehouse,
               Ave Maria Lane",
  year = "1890"
}
```

———-

— **ignore** —

```
\bibitem[Taivalsaari 96]{Tai96} Taivalsaari, Antero
  title = {{On the Notion of Inheritance}},
ACM Computing Surveys, Vol 28 No 3 Sept 1996 pp438-479
```

———-

— **ignore** —

```
\bibitem[Temme 87]{Tem87} Temme N M.
  title = {{On the Computation of the Incomplete Gamma Functions for Large
            Values of the Parameters}},
Algorithms for Approximation. (ed J C Mason and M G Cox)
```

```
Oxford University Press. (1987)
```

———-

— **ignore** —

```
\bibitem[Temperton 83a]{Tem83a} Temperton C.
  title = {{Self-sorting Mixed-radix Fast Fourier Transforms}},
J. Comput. Phys. 52 1--23. (1983)
```

———-

— **ignore** —

```
\bibitem[Temperton 83b]{Tem83b} Temperton C.
  title = {{Fast Mixed-Radix Real Fourier Transforms}},
J. Comput. Phys. 52 340--350. (1983)
```

———-

— **axiom.bib** —

```
@article{That82,
  author = "Thatcher, J.W. and Wagner, E.G. and Wright, J.B.",
  title = {{Data Type Specification: Parameterization and the Power of
           Specification Techniques}},
  journal = "ACM TOPLAS",
  volume = "4",
  number = "4",
  pages = "711-732",
  year = "1982",
  abstract =
    "Our earlier work on abstract data types is extended by the answers to
    a number of questions on the power and limitations of algebraic
    specification techniques and by an algebraic treatment of
    parameterized data types like {\bf sets-of()} and
    {\bf stacks-of-()}. The ``hidden function'' problem (the need to include
    operations in specifications which are wanted hidden from the user) is
    investigated; the relative power of conditional specifications and
    equational specifications is investigated; the relative power of
    conditional specifications and equational specifications is
    investigated; and it is shown that parameterized specifications must
    contain ``side conditions'' (e.g. that {\bf finite-sets-of-d} requires
    an equality predicate on {\bf d}).",
  paper = "That82.pdf"
}
```

———-

— **axiom.bib** —

```
@article{Thur94,
  author = "Thurston, William P.",
  title = {{On Proof and Progress in Mathematics}},
  journal = "Bulletin AMS",
  volume = "30",
  number = "2",
  month = "April",
  year = "1994",
  link = "\url{https://arxiv.org/pdf/math/9404236.pdf}",
  paper = "Thur94.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Tray10,
  author = "Traytel, Bmytro",
  title = {{Extensions of a Type Inference Algorithm with Coerce Subtyping}},
  school = "Der Technischen Universitat Munchen",
  link = "\url{https://www21.in.tum.de/~traytel/bscthesis.pdf}",
  year = "2010",
  abstract =
    "Subtyping with coercion semantics allows a type inference system to
    correct some ill-typed programs by the automatic insertion of
    implicit type conversions at run-time. This simplifies programmers
    life but has its price: the general typability problem for given base
    type subtype dependencies is NP-complete.  Nevertheless, if the given
    coercions define an order on types with certain properties, the
    problem behaves in a sane way in terms of complexity.  This thesis
    presents an algorithm that can be used to extend Hindley-Milner type
    inference with coercive subtyping assuming a given partial order on
    base types. Especially, we discuss restrictions on the subtype
    dependencies that are necessary to achieve an efficient
    implementation. Examples of problems that occur if these restrictions
    are not met are given. The result of these considerations is that the
    algorithm is complete if the given base type poset is a disjoint union
    of lattices. Furthermore, the interaction of subtyping with type
    classes is addressed.  The algorithm that is extended to deal with
    type classes requires even a stronger restriction to assure
    completeness.  An ML-implementation of the presented algorithm is used
    in the generic proof assistant Isabelle.",
  paper = "Tray10.pdf",
  keywords = "printed"
}
```

———-

— **axiom.bib** —

```
@misc{Tryb02,
  author = "Trybulec, Andrzej",
```

```
  title = {{Towards Practical Formalizaiton of Mathematics}},
  comment = "abstract",
  link =
  "\url{http://www.macs.hw.ac.uk/~fairouz/forest//events/automath2002/abstracts/Andrzej.abst.html}",
  abstract =
    "In sixties the opinion that mathematics cannot be practically
    formalized was well established. So the main achievement of de Bruijn,
    among many others, is the decision that the problem must be reconsider
    and probably positively solved. And I believe that 1967 will be
    treated by future historians of mathematics as a turning point.

    It is not easy to say precisely what we learnt in the meantime, if we
    learnt anything at all. I believe that the most important results are:
    \begin{enumerate}
    \item we need experiments with much more advanced mathematics than
    already done
    \item a system for practical formalization of mathematics probably will
    not be a simple system based on small number of primitive notions
    \item integration with a computer algebra systems may be necessary or at
    least a feasible system must have bigger computational power.
    \end{enumerate}

    Still we should me more concerned with the original ideas of de
    Bruijn, the most important that eventually we have to be able to
    formalize new mathematical results, published in mathematical
    newspapers in this century."
}
```

———-

1.41.21 U

— **ignore** —

```
\bibitem[Unknown 61]{Unk61} Unknown
  title = {{Chebyshev-series}},
Modern Computing Methods
Chapter 8. NPL Notes on Applied Science (2nd Edition). 16 HMSO. 1961
```

———-

— **axiom.bib** —

```
@InProceedings{Mart99,
  author = "Martin, Ursula",
  title = {{Computers, reasoning and mathematical practice}},
  booktitle = "Computational Logic",
  publisher = "Springer",
  year = "1999",
  location = "Marktoberdorf, Germany",
  pages = "301-346",
```

```
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.50.2061}",
  abstract =
    "We identify three main objectives for computer aided reasoning
    enhancing the techniques available for mathematical experimentation,
    developing community standards for experiment and modelling and
    developing methods which will make computation more acceptable as part
    of a proof. We discuss three areas of research which address these:
    the use of theorem proving techniques to enhance or extend
    mathematical software systems, support for formal methods techniques
    to increase the reliability of such systems, and the use of computer
    aided formal reasoning in support of mathematical practice. This last
    includes activities such as formalizing systems for computational
    mathematics or visualization so that they can still be used informally
    but generate a formal development, and developing techniques to
    provide assistance in the initial stages of developing a new theory.

    By mathematics here we mean the activities of working research
    mathematicians, producing new results in pure or applied mathematics,
    although we touch briefly on some questions concerning the
    applications of computational mathematics and simulation in research
    science and engineering. We have left out several other related areas
    entirely: logical questions of decidability, soundness or
    completeness, theoretical computer science issues of semantics,
    computability or complexity, foundational issues such as
    constructivity, computer aided proofs about software and hardware, and
    the use of computers in mathematical education at all levels and in
    heuristic discovery. In particular foundational questions about
    computation have transformed mathematical logic, computation has made
    constructive proof feasible, and effective notions of practice for
    proofs about hardware and software are by no means well understood.
    However these matters fall outside the scope of this paper.",
  paper = "Mart99.pdf",
  keywords = "axiomref"
}
```

———-

1.41.22 V

— **ignore** —

```
\bibitem[Van Dooren 76]{vDDR76} Van Dooren P.;  De Ridder L.
  title = {{An Adaptive Algorithm for Numerical Integration over an
            N-dimensional Cube}},
J. Comput. Appl. Math. 2 207--217. (1976)
```

———-

— **ignore** —

```
\bibitem[van Hoeij 94]{REF-vH94} van Hoeij, M.
```

```
  title = {{An algorithm for computing an integer}},
basis in an algebraic function field''
{\sl J. Symbolic Computation}
18(4):353-364, October 1994
```

———-

— **ignore** —

```
\bibitem[Van Loan 92]{Van92} Van Loan, C.
  title = {{Computational Frameworks for the Fast Fourier Transform}},
SIAM Philadelphia. (1992)
```

———-

— **axiom.bib** —

```
@misc{Vogl07,
  author = "Vogler, Doctor",
  title = {{Genus of a Plane Curve}},
  year = "2007",
  link = "\url{http://mathforum.org/library/drmath/view/71229.html}",
  algebra =
    "\newline\refto{package GPAFF GeneralPackageForAlgebraicFunctionField}
\newline\refto{package PAFFFF PackageForAlgebraicFunctionFieldOverFiniteField}
     \newline\refto{package PAFF PackageForAlgebraicFunctionField}"
}
```

———-

— **axiom.bib** —

```
@techreport{Vuil87,
  author = "Vuillemin, J.",
  title = {{Exact Real Computer Arithmetic with Continued Fractions}},
  institution = "Institut National de Recherche en Informatique et en
                 Automatique",
  year = "1987",
  number = "760",
  abstract =
    "We discuss a representation of the {\sl computable real numbers} by
    continued fractions. This deals with the subtle points of undecidable
    and integer division, as well as representing the infinite
    $\infty=1/0$ and undefined $\bot=0/0$ numbers. Two general algorithms
    for performing arithmetic operations are introduced. The
    {\sl algebraic algorithm}, which computes sums and products of
    continued fractions as a special case, basically operates in a
    positional manner, producing one term of output for each term of
    input. The {\sl transcendental algorithm} uses a general formula of
    Gauss to compute the continued fractions of exponentials, logarithms,
    trigonometric functions, as well as a wide class of special
```

```
    functions. This work has been implemented in Le\_Lisp and the
    performance of these algorithms appears to be quite good; however, no
    competing system has been available for comparison",
  paper = "Vuil87.pdf"
}
```

———-

1.41.23 W

— **ignore** —

```
\bibitem[Wait 85]{WM85} Wait R.; Mitchell A R.
  title = {{Finite Element Analysis and Application}},
Wiley. (1985)
```

———-

— **axiom.bib** —

```
@book{Walk78,
  author = "Walker, Robert J.",
  title = {{Algebraic Curves}},
  year = "1978",
  publisher = "Princeton University Press",
  isbn = "978-0-387-90361-3",
  algebra =
    "\newline\refto{package GPAFF GeneralPackageForAlgebraicFunctionField}
\newline\refto{package PAFFFF PackageForAlgebraicFunctionFieldOverFiniteField}
     \newline\refto{package PAFF PackageForAlgebraicFunctionField}"
}
```

———-

— **ignore** —

```
\bibitem[Wang 92]{Wang92} Wang, D.M.
  title = {{An implementation of the characteristic set method in Maple}},
Proc. DISCO'92 Bath, England
```

———-

— **ignore** —

```
\bibitem[Ward 75]{War75} Ward, R C.
  title = {{The Combination Shift QZ Algorithm}},
SIAM J. Numer. Anal. 12 835--853. 1975
```

———-

— **axiom.bib** —

```
@misc{Watt03,
  author = "Watt, Stephen",
  title = {{Aldor}},
  link = "\url{http://www.aldor.org}",
  year = "2003"
}
```

———-

— **axiom.bib** —

```
@article{Wegb74,
  author = "Wegbreit, Ben",
  title = {{The Treatment of Data Types in EL1}},
  journal = "Communications of the ACM",
  volume = "17",
  number = "5",
  year = "1974",
  pages = "251-264",
  abstract =
    "In constructing a general purpose programming language, a key issue
    is providing a sufficient set of data types and associated operations
    in a manner that permits both natural problem-oriented notation and
    efficient implementation. The EL1 language contains a number of
    features specifically designed to simultaneously satisfy both
    requirements. The resulting treatment of data types includes provision
    for programmer-defined data types and generaic routines, programmer
    control over type conversion, and very flexible data type behavior, in
    a context that allows efficient compiled code and compact data
    representation.",
  paper = "Wegb74.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Weil71,
  author = "Weil, Andr\'{e}",
  title = {{Courbes alg\'{e}briques et vari\'{e}t\'{e}s Abeliennes}},
  year = "1971"
}
```

———-

— **ignore** —

```
\bibitem[Weisstein]{Wein} Weisstein, Eric W.
  title = {{Hypergeometric Function}},
```

```
MathWorld - A Wolfram Web Resource
  link = "\url{http://mathworld.wolfram.com/HypergeometricFunction.html}",
```

———-

— **axiom.bib** —

```
@misc{Weit03,
  author = "Weitz, E.",
  title = {{CL-WHO -Yet another Lisp markup language}},
  year = "2003",
  link = "\url{http://www.weitz.de/cl-who/}"
}
```

———-

— **axiom.bib** —

```
@misc{Weit06,
  author = "Weitz, E.",
  title = {{HUNCHENTOOT - The Common Lisp web server formerly known as TBNL}},
  year = "2006",
  link = "\url{http://www.weitz.de/hunchentoot}"
}
```

———-

— **ignore** —

```
\bibitem[Wesseling 82a]{Wes82a} Wesseling, P.
  title = {{MGD1 - A Robust and Efficient Multigrid Method}},
Multigrid Methods. Lecture Notes in Mathematics. 960
Springer-Verlag. 614--630. (1982)
```

———-

— **ignore** —

```
\bibitem[Wesseling 82b]{Wes82b} Wesseling, P.
  title = {{Theoretical Aspects of a Multigrid Method}},
SIAM J. Sci. Statist. Comput. 3 387--407. (1982)
```

———-

— **ignore** —

```
\bibitem[Wicks 89]{Wic89} Wicks, Mark; Carlisle, David, Rahtz, Sebastian
  title = {{dvipdfm.def}},
  link = "\url{http://web.mit.edu/texsrc/source/latex/graphics/dvipdfm.def}",
```

———-

— **axiom.bib** —

```
@article{Wied03,
  author = "Wiedijk, Freek",
  title = {{Formal Proof Sketches}},
  journal = "LNCS",
  volume = "3085",
  year = "2003",
  pages = "378-393",
  abstract =
    "Formalized mathematics currently does not look much like informal
    mathematics. Also, formalizing mathematics currently seems far too
    much work to be worth the time of the working mathematician. To
    address both of these problems we introduce the notion of a formal
    proof sketch . This is a proof representation that is in between a
    fully checkable formal proof and a statement without any proof at
    all. Although a formal proof sketch is too high level to be checkable
    by computer, it has a precise notion of correctness (hence the
    adjective formal ).  We will show through examples that formal proof
    sketches can closely mimic already existing mathematical
    proofs. Therefore, although a formal proof sketch contains gaps in
    the reasoning from a formal point of view (which is why we call it a
    sketch ), a mathematician probably would call such a text just a
    proof.",
  paper = "Wied03.pdf",
  keywords = "printed"
}
```

———-

— **ignore** —

```
\bibitem[Wiki 3]{Wiki3}.
  title = {{Givens Rotations}},
  link = "\url{http://en.wikipedia.org/wiki/Givens_rotation}",
```

———-

— **axiom.bib** —

```
@misc{Wiki14a,
  author = "ProofWiki",
  title = {{Euclidean Algorithm}},
  link = "\url{http://proofwiki.org/wiki/Euclidean_Algorithm}"
}
```

———-

— **axiom.bib** —

```
@misc{Wiki14b,
  author = "ProofWiki",
  title = {{Division Theorem}},
  link = "\url{http://proofwiki.org/wiki/Division_Theorem}"
}
```

———-

— **axiom.bib** —

```
@misc{Wiki16,
  author = "Anonymous",
  title = {{Levenshtein distance}},
  year = "2016",
  link = "\url{https://en.wikipedia.org/wiki/Levenshtein\_distance}"
}
```

———-

— **ignore** —

```
\bibitem[Williamson 85]{Wil85} Williamson, S.G.
  title = {{Combinatorics for Computer Science}},
Computer Science Press, 1985.
```

———-

— **ignore** —

```
\bibitem[Wilkinson 71]{WR71} Wilkinson J H.; Reinsch C.
  title = {{Handbook for Automatic Computation II, Linear Algebra}},
Springer-Verlag. 1971
```

———-

— **ignore** —

```
\bibitem[Wilkinson 63]{Wil63} Wilkinson J H.
  title = {{Rounding Errors in Algebraic Processes}},
 Chapter 2. HMSO. (1963)
```

———-

— **ignore** —

```
\bibitem[Wilkinson 65]{Wil65} Wilkinson J H.
  title = {{The Algebraic Eigenvalue Problem}},
 Oxford University Press. (1965)
```

———-

— **ignore** —

```
\bibitem[Wilkinson 78]{Wil78} Wilkinson J H.
  title = {{Singular Value Decomposition -- Basic Aspects}},
Numerical Software -- Needs and Availability.
(ed D A H Jacobs) Academic Press. (1978)
```

———-

— **ignore** —

```
\bibitem[Wilkinson 79]{Wil79} Wilkinson J H.
  title = {{Kronecker's Canonical Form and the QZ Algorithm}},
Linear Algebra and Appl. 28 285--303. 1979
```

———-

— **ignore** —

```
\bibitem[Wisbauer 91]{Wis91} Wisbauer, R.
  title = {{Bimodule Structure of Algebra}},
Lecture Notes Univ. Duesseldorf 1991
```

———-

— **ignore** —

```
\bibitem[Woerz-Busekros 80]{Woe80} Woerz-Busekros, A.
  title = {{Algebra in Genetics}},
Lectures Notes in Biomathematics 36, Springer-Verlag,  Heidelberg, 1980
```

———-

— **ignore** —

```
\bibitem[Wolberg 67]{Wol67} Wolberg J R.
''Prediction Analysis''
Van Nostrand. (1967)
```

———-

— **ignore** —

```
\bibitem[Wolfram 09]{Wo09} Wolfram Research
  link = "\url{http://mathworld.wolfram.com/Quaternion.html}",
```

———-

— **axiom.bib** —

```
@article{Wosx92,
  author = "Wos, Larry",
  title = {{The Impossibility of the Automation of Logical Reasoning}},
  journal = "Lecture Notes in Computer Science",
  volume = "607",
  year = "1992",
  pages = "1-3",
  paper = "Wosx92.pdf"
}
```

———-

— **axiom.bib** —

```
@book{Wuxx94,
  author = "Wu, Wen-tsun",
  title = {{Mechanical Theorem Proving in Geometries}},
  isbn = "978-3-7091-6639-0",
  publisher = "Springer",
  year = "1994"
}
```

———-

— **ignore** —

```
\bibitem[Wu 87]{WU87} Wu, W.T.
  title = {{A Zero Structure Theorem for polynomial equations solving}},
MM Research Preprints, 1987
```

———-

— **axiom.bib** —

```
@article{Wulf76,
  author = "Wulf, William A. and London, Ralph L. and Shaw, Mary",
  title = {{An Introduction to the Construction and Verification of
           Alphard Programs}},
  journal = "IEEE Tr. Software Engineering",
  volume = "SE-2",
  number = "4",
  year = "1976",
  pages = "253-265",
  abstract =
    "The programming language Alphard is designed to provide support for
    both the methodologies of ``well-structured'' programming and the
    techniques of formal program verification.  Language constructs allow
```

```
    a programmer to isolate an abstraction, specifying its behavior
    publicly while localizing.  knowledge about its implementation.  The
    verification of such an abstraction consists of showing that its
    implementation behaves in accordance with its public specifications;
    the abstraction can then be used with confidence in constructing
    other programs, and the verification of that use employs only the
    public specifications.  This paper introduces Alphard by developing
    and verifying a data structure definition and a program that uses it.
    It shows how each language construct contributes to the development of
    the abstraction and discusses the way the language design and the
    verification methodology wete tailored to each other.  It serves not
    only as an introduction to Alphard, but also as an example of the
    symbiosis between verification and methodology in language design.
    The strategy of program structuring, illustrated for Alphard, is
    also applicable to most of the ``data abstraction'' mechanisms now
    appearing.",
  paper = "Wulf76.pdf"
}
```

———-

— **ignore** —

```
\bibitem[Wynn 56]{Wynn56} Wynn P.
  title = {{On a Device for Computing the $e_m(S_n )$ Transformation}},
Math. Tables Aids Comput. 10 91--96. (1956)
```

———-

1.41.24 X

1.41.25 Y

1.41.26 Z

— **ignore** —

```
\bibitem[Zakrajsek 02]{Zak02} Zakrajsek, Helena
  title = {{Applications of Hermite transform in computer algebra}},
  link = "\url{http://www.imfm.si/preprinti/PDF/00835.pdf}",
  abstract = "
    let $L$ be a linear differential operator with polynomial
    coefficients.  We show that there is an isomorphism of differential
    operators ${\bf D_\alpha}$ and an integral transform ${\bf H_\alpha}$
    (called the Hermite transform) on functions for which $({\bf
    D_\alpha}{\bf L})f(x)=0$ implies ${\bf L}{\bf H_alpha}(f)(x)=0$. We
    present an algorithm that computes the Hermite transform of a rational
    function and use it to find $n+1$ linearly independent solutions of
    ${\bf L}y=0$ when $({\bf D_\alpha}{\bf L})f(x)=0$ has a rational
    solution with $n$ distinct finite poles.",
  paper = "Zak02.pdf"
```

———-

— **axiom.bib** —

```
@misc{Zdan14,
  author = "Zdancewic, Steve and Martin, Milo M.K.",
  title = {{Vellvm: Verifying the LLVM}},
  link = "\url{http://www.cis.upenn.edu/~stevez/vellvm}"
}
```

———-

— **ignore** —

```
\bibitem[Zhi 97]{Zhi97}
  author = "Zhi, Lihong",
  title = {{Optimal Algorithm for Algebraic Factoring}},
  year = "1997",
  link = "\url{http://www.mmrc.iss.ac.cn/~lzhi/Publications/zopfac.pdf}",
  abstract = "
    This paper presents an optimized method for factoring multivariate
    polynomials over algebraic extension fields which defined by an
    irreducible ascending set. The basic idea is to convert multivariate
    polynomials to univariate polynomials and algebraic extensions fields
    to algebraic number fields by suitable integer substitutions, then
    factorize the univariate polynomials over the algebraic number fields.
    Finally, construct multivariate factors of the original polynomial by
    Hensel lemma and TRUEFACTOR test. Some examples with timing are
    included.",
  paper = "Zhi97.pdf"
```

———-

— **axiom.bib** —

```
@phdthesis{Zinn04,
  author = "Zinn, Claus",
  title = {{Understanding Informal Mathematical Discourse}},
  school = "Universitat Erlangen Nurnberg",
  year = "2004",
  abstract =
    "Automated reasoning is one of the most established disciplines in
    informatics and artificial intelligence, and formal methods become
    increasingly employed in practical applications. However, for the most
    part, such applications seem to be limited to informatics-specific
    areas (e.g., the verification of correctness properties of software
    and hardware specifications) and areas close to informatics such as
    computational linguistics (e.g., the computation of consistence and
    informativeness properties of semantic representations). Naturally,
    there is also a potential for practical applications in the the area
```

of mathematics: the generation of proofs for mathematically interesting and motivated theorems and, quite associated, the computer-supported formalisation of (parts of elementary) mathematics. It is a matter of fact, however, that mathematicians rarely use computer-support for the construction and verification of proofs. This is mainly caused by the unnaturalness of the language and the reasoning that such proof engines support.

In the past, researchers in the area of automated reasoning have focused their work on formalisms and algorithms that allow the construction and verification of proofs that are written in a formal-logical language and that only use a limited number of inference rules. For the computer scientist, such formal proofs have the advantage of a simple and ambiguous-free syntax, which can thus be easily processed. Moreover, the limited number of inference rules has a direct impact on the complexity of the search space that needs to be conquered during the process of constructing proofs. The verification of given formal proofs is greatly facilitated by the complete explicitness of their logical argumentation where no reasoning step is left out. For mathematicians, however, such formal proofs are usually hard to understand. For them, they are written in an unfamiliar and artificial language and much too detailed. Moreover, the sheer number of inference steps, while logically relevant, describe only trivial mathematical details and make it difficult to follow a proofs main underlying argumentation line. In practise, thus, mathematicians use proof generation engines rather seldom, if at all. The same is true with regard to proof verification tools. The amount of work that is required to verify a mathematical proof with such tools is considerable, if not prohibitive. Since proof verification systems only accept formal proofs as input, the mathematicians first task is to manually translate the mathematical proof into the formal language that is accepted by the verifier. This in turn includes the translation of the proofs underlying mathematical argumentation into inference rules that are supported by the proof engine. Such translations and refinements are usually very time consuming, tedious, and prone to error themselves. Hence, how the proof verifier then judges the result of proof translation and proof refinement is only of limited relevance to the original mathematical proof. From the mathematicians point of view, there is thus a need for a proof verification system that is capable of processing mathematical proofs automatically, at least with regard to translating the mathematicians expert language into the systems artificial formal language. Such a system would have an enormous potential in the community of mathematics, and this potential has been recognised early. In the beginning of the 1960s, John McCarthy, one of the pioneers of artificial intelligence, remarked that ‘‘[c]hecking mathematical proofs is potentially one of the most interesting and useful applications of automatic computers’’ [111]. More than forty years thereafter, a tool that supports mathematicians with the verification of mathematical proofs is more science fiction than reality. Its realisation is associated with research questions within the disciplines of automated reasoning and computational linguistics that are still only partially answered.

```
    This thesis aims at contributing towards the realisation of a
    verifier for mathematical proofs. It attempts to provide a general
    framework as well as an implementation for such a proof
    engine. The dissertations objects of study are short and simple
    proofs that were taken from textbooks on elementary number
    theory. Fig. 1 depicts a proof of the mathematical truth that
    every positive integer greater than 1 can be represented as a
    product of one or more primes. The proof consists of only a few
    lines and little mathematical knowledge is necessary to follow the
    proof authors argumentation. It is this kind of short proof that
    we attempt to check automatically.",
  paper = "Zinn04.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{OCon08,
  author = "O'Connor, Christine",
  title = {{Christine Jeanne O'Connor Obituary}},
  year = "2008",
  link = "\url{http://www.cargainfuneralhomes.com/home/index.cfm/obituaries/view/fh_id/10350/id/183842}"
}
```

———-

— **axiom.bib** —

```
@misc{Hale13,
  author = "Hales, Thomas C.",
  title = {{Mathematics in the Age of the Turing Machine}},
  year = "2013",
  link = "\url{http://arxiv.org/pdf/1302.2898v1}",
  abstract = "
    This article gives a survey of mathematical proofs that rely on
    computer calculations and formal proofs",
  paper = "Hale13.pdf"
}
```

———-

— **axiom.bib** —

```
@misc{Martxx,
  author = "Martin, W.A. and Fateman, R.J.",
  title = {{The Macsyma System}},
  link = "\url{http://groups.csail.mit.edu/mac/classes/symbolic/spring13/readings/simplification/martin-fateman-
  abstract = "
    MACSYMA is a system for symbolic manipulation of algebraic expressions
    which is being developed at Project MAC, M.I.T. This paper discusses
```

```
    its philosophy, goals, and current achievements.

    Drawing on the past work of Maring, Moses, and Engelman, it extends
    the capabilities of automated algebraic manipulation systems in
    several areas, including

    a) limit calculations
    b) symbolic integration
    c) solution of equations
    d) canonical simplification
    e) user-level pattern matching
    f) user-specified expression manipulation
    g) programming and bookkeeping assistance

    MACSYMA makes extensive use of the power of its rational function
    subsystem. The facilities derived from this are discussed in
    considerable detail.

    An appendix briefly notes some highlights of the overall system.",
  paper = "Martxx.pdf"

}
```

———-

— **axiom.bib** —

```
@misc{Andr00,
  author = "Andrews, George E. and Knopfmacher, Arnold and Paule, Peter
            and Zimmermann, Burkhard",
  title = {{Engel Expansions of q-Series by Computer Algebra}},
  year = "2000",
  link = "\url{http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.207}",
  abstract = "
    The $q$-Engle Expansion is an algorithm that leads to unique series
    expansions of $q$-series. Various examples related to classical
    partition theorems, including the Rogers-Ramanujan identities together
    with the elegant generalization found by Garrett, Ismail and Stanton,
    have been described recently. The object of this paper is to present
    the computer algebra package Engel, written in Mathematics, that has
    already played a signiicant role in this work. The package now is made
    freely available via the web and should help to intensify research in
    this new branch of $q$-series theory. Among various illustrative
    examples we present a new infinite Rogers-Ramanujan type family that
    has been discovered by using the package.",
  paper = "Andr00.pdf"

}
```

———-

— **axiom.bib** —

```
@misc{Abra99,
  author = "Abramov, Sergei A. and van Hoeij, Mark",
  title = {{Integration of Solutions of Linear Functional Equations}},
  year = "1999",
  link = "\url{http://www.math.fsu.edu/~hoeij/papers/itsf99/ab_final.pdf}",
  abstract = "
    We introduce the notion of the adjoint Ore ring and give a definition
    of adjoint polynomial, operator and equation. We apply this for
    integrating solutions of Ore equations.",
  paper = "Abra99.pdf"

}
```

———-

— **axiom.bib** —

```
@misc{Bail97,
  author = "Bailey, David and Borwein, Peter and Plouffe, Simon",
  title = {{On the Rapid Computation of Various Polylogarithmic Constants}},
  year = "1997",
  link = "\url{http://www.ams.org/journals/mcom/1997-66-218/S0025-5718-97-00856-9/S0025-5718-97-00856-9.pdf}",
  abstract = "
    We give algorithms for the computation of the $d$-th digit of certain
    transcendental numbers in various bases. These algorithms can be
    easily implemented (multiple precision arithmetic is not needed),
    require virtually no memeory, and feature run times that scale nearly
    linearly with the order of the digit desired. They make it feasible to
    compute, for example, the billionth binary digit of log(2) or $\pi$ on
    a modest work station in a few hours run time.

    We demonstrate this technique by computing the ten billionth
    hexadecimal digit of $\pi$, the billionth hexadecimal digits of
    $\pi^2$, log(2), and log${}^2$(2), and the ten billionth decimal digit
    of log(9/10).

    These calculations rest on the observation that very special types of
    identities exist for certain numbers like $\pi$, $\pi^2$, log(2) and
    log${}^2$. These are essentially polylogarithmic ladders in an integer
    base. A number of these identities that we derive in this work appear
    to be new, for example the critical identity for $\pi$:

    \[\pi=\sum_{i=0}^\infty{\frac{1}{16^i}\left(
    \frac{4}{8i+1}-\frac{2}{8i+4}-\frac{1}{8i+5}-\frac{1}{8i+6}\right)}\]",
  paper = "Bail97.pdf"

}
```

———-

— **axiom.bib** —

```
@misc{Thie15,
  author = "Thiery, Nicolas M.",
  title = {{Open Digital Research Environment Toolkit for the
            Advancement of Mathematics}},
  year = "2015",
  link = "\url{http://opendreamkit.org}",
  abstract =
   "OpenDreamKit will deliver a flexible toolkit enabling research groups
    to set up Virtual Research Environments, customised to meet the varied
    needs of research projects in pure mathematics and applications, and
    supporting the full research life-cycle from exploration, through
    proof and publication, to archival and sharing of data and code.

    OpenDreamKit will be built out of a sustainable ecosystem of
    community-developed open software, databases, and services,
    including popular tools such as LINBOX, MPIR, SAGE (sagemath.org),
    GAP, PARI/GP, LMFDB, and SINGULAR. We will extend the JUPYTER Notebook
    environment to provide a flexible user interface. By improving and
    unifying existing building blocks, OpenDreamKit will maximise both
    sustainability and impact, with beneficiaries extending to scientific
    computing, physics, chemistry, biology and more, and including
    researchers, teachers, and industrial practitioners.

    We will define a novel component-based VRE architecture and adapt
    existing mathematical software, databases, and user interface
    components to work well within it on varied platforms. Interfaces to
    standard HPC and grid services will be built in. Our architecture will
    be informed by recent research into the sociology of mathematical
    collaboration, so as to properly support actual research practice. The
    ease of set up, adaptability and global impact will be demonstrated in
    a variety of demonstrator VREs.

    We will ourselves study the social challenges associated with
    large-scale open source code development and publications based on
    executable documents, to ensure sustainability.

    OpenDreamKit will be conducted by a Europe-wide steered by demand
    collaboration, including leading mathematicians, computational
    researchers, and software developers with a long track record of
    delivering innovative open source software solutions for their
    respective communities. All produced code and tools will be open
    source.",
  paper = "Thie15.pdf"
}
```

———-

Chapter 2

Beebe Bibliography

— beebe.bib —

```
%%% -*-BibTeX-*-
%%% ====================================================================
%%%  BibTeX-file{
%%%     author          = "Nelson H. F. Beebe",
%%%     version         = "1.18",
%%%     date            = "15 August 2014",
%%%     time            = "18:59:18 MDT",
%%%     filename        = "axiom.bib",
%%%     address         = "University of Utah
%%%                        Department of Mathematics, 110 LCB
%%%                        155 S 1400 E RM 233
%%%                        Salt Lake City, UT 84112-0090
%%%                        USA",
%%%     telephone       = "+1 801 581 5254",
%%%     FAX             = "+1 801 581 4148",
%%%     URL             = "http://www.math.utah.edu/~beebe",
%%%     checksum        = "22023 4297 20558 200979",
%%%     email           = "beebe at math.utah.edu, beebe at acm.org,
%%%                        beebe at computer.org (Internet)",
%%%     codetable       = "ISO/ASCII",
%%%     keywords        = "bibliography; AXIOM;
%%%                       Scratchpad; symbolic algebra",
%%%     license         = "public domain",
%%%     supported       = "yes",
%%%     docstring       = "This file contains a bibliography of
%%%                        publications about the AXIOM (formerly
%%%                        known as Scratchpad) symbolic algebra
%%%                        system.  It also covers publications about
%%%                        Scratchpad and a few about MathCAD.
%%%
%%%                        At version 1.18, the year coverage looked
%%%                        like this:
%%%
%%%                             1971 (   3)    1984 (   6)    1997 (   0)
```

```
%%%                     1972 (   3)    1985 (   2)    1998 (   1)
%%%                     1973 (   0)    1986 (   4)    1999 (   2)
%%%                     1974 (   1)    1987 (   5)    2000 (   0)
%%%                     1975 (   1)    1988 (   8)    2001 (   1)
%%%                     1976 (   1)    1989 (  18)    2002 (   1)
%%%                     1977 (   1)    1990 (   7)    2003 (   3)
%%%                     1978 (   0)    1991 (  16)    2004 (   1)
%%%                     1979 (   0)    1992 (  16)    2005 (   9)
%%%                     1980 (   0)    1993 (  10)    2006 (   2)
%%%                     1981 (   0)    1994 (  11)    2007 (   6)
%%%                     1982 (   0)    1995 (   7)    2008 (   1)
%%%
%%%                     Article:         34
%%%                     Book:            17
%%%                     InCollection:     1
%%%                     InProceedings:   48
%%%                     Manual:           1
%%%                     MastersThesis:    2
%%%                     Proceedings:     28
%%%                     TechReport:      16
%%%
%%%                     Total entries:  147
%%%
%%%                 Scratchpad and AXIOM were developed at IBM
%%%                 research laboratories over many years before
%%%                 they were offered as supported products.
%%%                 Today, AXIOM is marketed for several
%%%                 platforms by the Numerical Algorithms Group,
%%%                 Inc. (NAG) (Downer's Grove, IL, USA. and
%%%                 Oxford, UK), and versions are available in
%%%                 late 1995 for Hewlett--Packard 9000 (HP-UX),
%%%                 IBM RS/6000 (AIX), and Sun (SunOS) systems,
%%%                 with ports to other systems under
%%%                 development.
%%%
%%%                 Further information on AXIOM licensing
%%%                 can be found at http://www.nag.com/ and
%%%                 http://www.nag.co.uk/1h/symbolic/AX.html.
%%%
%%%                 NAG maintains an AXIOM code and documentation
%%%                 repository at http://www.nag.co.uk/symbolic/
%%%                 AX/Upload_Readme.html.  All of the NAG
%%%                 technical reports listed in this bibliography
%%%                 can be found at that address.
%%%
%%%                 In 2006, AXIOM became free software, with a
%%%                 Web site at
%%%
%%%                     http://www.axiom-developer.org
%%%
%%%                 This bibliography has been collected from the
%%%                 author's personal bibliography files, from
%%%                 the very large computer science bibliography
%%%                 collection on ftp.ira.uka.de in
```

```
%%%                         /pub/bibliography to which many people of
%%%                         have contributed, and from several
%%%                         Internet-accessible library catalogs, notably
%%%                         those of the University of California,
%%%                         Library of Congress, OCLC, plus the IEEE
%%%                         INSPEC (1989--1995) database.
%%%
%%%                         This bibliography is sorted by year, and
%%%                         within each year, by author and title key,
%%%                         with ``bibsort -byyear''.  Cross-referenced
%%%                         proceedings entries appear at the end,
%%%                         because of a restriction in the current
%%%                         BibTeX.
%%%
%%%                         The checksum field above contains a CRC-16
%%%                         checksum as the first value, followed by the
%%%                         equivalent of the standard UNIX wc (word
%%%                         count) utility output of lines, words, and
%%%                         characters.  This is produced by Robert
%%%                         Solovay's checksum utility.",
%%%  }
%%% ====================================================================

%%% Acknowledgement abbreviations:

@String{ack-nhfb = "Nelson H. F. Beebe,
                    University of Utah,
                    Department of Mathematics, 110 LCB,
                    155 S 1400 E RM 233,
                    Salt Lake City, UT 84112-0090, USA,
                    Tel: +1 801 581 5254,
                    FAX: +1 801 581 4148,
                    e-mail: \path|beebe@math.utah.edu|,
                            \path|beebe@acm.org|,
                            \path|beebe@computer.org| (Internet),
                    URL: \path|http://www.math.utah.edu/~beebe/|"}

%%% ====================================================================
%%% Institution abbreviations:

@String{inst-NAG                = "Numerical Algorithms Group, Inc."}

@String{inst-NAG:adr            = "Downer's Grove, IL, USA and Oxford,
                                   UK"}

%%% ====================================================================
%%% Journal abbreviations:

@String{j-ACM-COMM-COMP-ALGEBRA = "ACM Communications in Computer Algebra"}

@String{j-AMER-J-PHYSICS        = "American Journal of Physics"}

@String{j-COED                  = "CoED"}
```

```
@String{j-COMP-PHYS-COMM        = "Computer Physics Communications"}

@String{j-ELECTRONIK            = "Elektronik"}

@String{j-IFIP-TRANS-A          = "IFIP Transactions. A. Computer Science and
                                  Technology"}

@String{j-J-SYMBOLIC-COMP       = "Journal of Symbolic Computation"}

@String{j-LECT-NOTES-COMP-SCI   = "Lecture Notes in Computer Science"}

@String{j-MATH-COMP-EDU         = "Mathematics and computer education"}

@String{j-SIGPLAN               = "ACM SIG{\-}PLAN Notices"}

@String{j-SIGSAM                = "SIGSAM Bulletin (ACM Special Interest
                                  Group on Symbolic and Algebraic
                                  Manipulation)"}

@String{j-STAT-COMP             = "Statistics and Computing"}

@String{j-THEOR-COMP-SCI        = "Theoretical Computer Science"}

@String{j-TOMS                  = "ACM Transactions on Mathematical Software"}

@String{j-ZEIT-ANGE-MATH-PHYS   = "Zeitschrift fur Angewandte Mathematik
                                  und Physik"}

%%% ====================================================================
%%% Publisher abbreviations:

@String{pub-ACM                 = "ACM Press"}

@String{pub-ACM:adr             = "New York, NY 10036, USA"}

@String{pub-AP                  = "Academic Press"}

@String{pub-AP:adr              = "New York, NY, USA"}

@String{pub-DEKKER              = "Marcel Dekker"}

@String{pub-DEKKER:adr          = "New York, NY, USA"}

@String{pub-SV                  = "Spring{\-}er-Ver{\-}lag"}

@String{pub-SV:adr              = "Berlin, Germany~/ Heidelberg,
                                  Germany~/ London, UK~/ etc."}

@String{pub-VIEWEG              = "Friedrich Vieweg und Sohn"}

@String{pub-VIEWEG:adr          = "Braunschweig, Germany"}

%%% ====================================================================
%%% Series abbreviations:
```

```
@String{ser-LNCS                = "Lecture Notes in Computer Science"}

%%% ====================================================================
%%% Bibliography entries:

\getchunk{Griesmer:1971:SIF}
\getchunk{Jenks:1971:MPS}
\getchunk{Griesmer:1972:EOSb}
\getchunk{Griesmer:1972:SCV}
\getchunk{Jenks:1974:SL}
\getchunk{Norman:1975:CFP}
\getchunk{Jenks:1976:PC}
\getchunk{Lueken:1977:UIF}
\getchunk{Andrews:1984:RS}
\getchunk{Davenport:1984:S}
\getchunk{Jenks:1984:NSL}
\getchunk{Jenks:1984:PKN}
\getchunk{Sutor:1985:SIC}
\getchunk{Gebauer:1986:BAS}
\getchunk{Jenks:1986:SIA}
\getchunk{Lucks:1986:FIP}
\getchunk{Purtilo:1986:ASI}
\getchunk{Burge:1987:ISS}
\getchunk{Senechaud:1987:SIP}
\getchunk{Sutor:1987:TICa}
\getchunk{Sutor:1987:TICb}
\getchunk{Andrews:1988:ASP}
\getchunk{Davenport:1988:CA}
\getchunk{Gebauer:1988:IBA}
\getchunk{Jenks:1988:SIA}
\getchunk{Schwarz:1988:PAD}
\getchunk{Shannon:1988:UGB}
\getchunk{Sutor:1988:SIA}
\getchunk{Boehm:1989:TIP}
\getchunk{Bronstein:1989:SRE}
\getchunk{Burge:1989:ISS}
\getchunk{Dicrescenzo:1989:AEA}
\getchunk{Gianni:1989:ASS}
\getchunk{Kusche:1989:IGT}
\getchunk{Mathews:1989:SCA}
\getchunk{Ollivier:1989:IRM}
\getchunk{Salvy:1989:EAA}
\getchunk{Schwarz:1989:FAL}
\getchunk{Sit:1989:GAS}
\getchunk{Wang:1989:PCL}
\getchunk{Watt:1989:FPM}
\getchunk{Davenport:1990:SVA}
\getchunk{Fateman:1990:ATD}
\getchunk{Fortenbacher:1990:ETI}
\getchunk{Fouche:1990:ILK}
\getchunk{Melachrinoudis:1990:TAT}
\getchunk{Augot:1991:MDS}
\getchunk{Bronstein:1991:RDE}
```

```
\getchunk{Burge:1991:SRI}
\getchunk{Davenport:1991:SVA}
\getchunk{Goodwin:1991:UMT}
\getchunk{Grabmeier:1991:CSA}
\getchunk{Koseleff:1991:WGF}
\getchunk{Lambe:1991:RHP}
\getchunk{LeBlanc:1991:UMT}
\getchunk{Lynch:1991:NQM}
\getchunk{Salvy:1991:EAA}
\getchunk{Schwarz:1991:MOG}
\getchunk{Wang:1991:MMC}
\getchunk{Anonymous:1992:PEH}
\getchunk{Camion:1992:PCG}
\getchunk{Dalmas:1992:PFL}
\getchunk{Davenport:1992:AS}
\getchunk{Davenport:1992:HDO}
\getchunk{Davenport:1992:SVAa}
\getchunk{Davenport:1992:SVAb}
\getchunk{Duval:1992:EPS}
\getchunk{Gil:1992:CJC}
\getchunk{Grabmeier:1992:FFA}
\getchunk{Jenks:1992:ASC}
\getchunk{Lambe:1994:NGC}
\getchunk{Rioboo:1992:RAC}
\getchunk{Sit:1992:ASP}
\getchunk{Smedley:1992:UPO}
\getchunk{Zenger:1992:GFD}
\getchunk{Bronstein:1993:FPF}
\getchunk{Davenport:1993:PTR}
\getchunk{Goodloe:1993:ADT}
\getchunk{Monagan:1993:GPD}
\getchunk{Petitot:1993:EA}
\getchunk{Weber:1993:CCA}
\getchunk{Beneke:1994:DFM}
\getchunk{Brown:1994:CSC}
\getchunk{Gruntz:1994:IG}
\getchunk{Jenks:1994:HMA}
\getchunk{Keady:1994:PAS}
\getchunk{Seiler:1994:CIA}
\getchunk{Seiler:1994:PDO}
\getchunk{vanHoeij:1994:ACI}
\getchunk{Anonymous:1995:GAM}
\getchunk{Arnault:1995:CCN}
\getchunk{Boulanger:1995:OOM}
\getchunk{Broadbery:1995:IDE}
\getchunk{Duval:1995:DEA}
\getchunk{Roesner:1995:VSP}
\getchunk{Benker:1998:ICS}
\getchunk{Doye:1999:ACA}
\getchunk{Kendall:2001:SIC}
\getchunk{Daly:2002:AOS}
\getchunk{Daly:2003:AVA}
\getchunk{Grabmeier:2003:CAH}
\getchunk{Jenks:2003:AVS}
```

```
\getchunk{Caviness:2004:MRD}
\getchunk{Daly:2005:AVAb}
\getchunk{Daly:2005:AVAc}
\getchunk{Daly:2005:AVAd}
\getchunk{Daly:2005:AVAe}
\getchunk{Daly:2005:AVAf}
\getchunk{Daly:2005:AVAg}
\getchunk{Daly:2005:AVAh}
\getchunk{Daly:2005:AVAi}
\getchunk{Daly:2005:AVAj}
\getchunk{Daly:2006:AVA}
\getchunk{Li:2006:EIP}
\getchunk{Li:2007:VGP}
\getchunk{Portes:2007:AVA}
\getchunk{Page:2007:AOS}
\getchunk{Smith:2007:ADA}
\getchunk{Joyner:2008:OSC}
\getchunk{Petrick:1971:PSS}
\getchunk{Online:1972:OCP}
\getchunk{Golden:1984:PMU}
\getchunk{Fitch:1984:E}
\getchunk{Buchberger:1985:EEC}
\getchunk{Wexelblat:1987:IIT}
\getchunk{Janssen:1988:TCA}
\getchunk{ACM:1989:PAI}
\getchunk{Huguet:1989:AAA}
\getchunk{Davenport:1989:EEC}
\getchunk{Gianni:1989:SAC}
\getchunk{Mora:1989:AAA}
\getchunk{Watanabe:1990:IPI}
\getchunk{Miola:1990:DIS}
\getchunk{Cohen:1991:EIS}
\getchunk{Watt:1991:PIS}
\getchunk{Anonymous:1991:PAC}
\getchunk{Wang:1992:ISS}
\getchunk{Bronstein:1993:IPI}
\getchunk{Fitch:1993:DIS}
\getchunk{Jacob:1993:PSI}
\getchunk{Miola:1993:DIS}
\getchunk{ACM:1994:IPI}
\getchunk{Calmet:1994:RWC}
\getchunk{Levelt:1995:IPI}
\getchunk{Dooley:1999:IJS}
\getchunk{Brown:2007:PIS}
\getchunk{Shi:2007:CSIb}
\getchunk{Proceedings}
```

———-

— **Anonymous:1992:PEH** —

```
@Article{Anonymous:1992:PEH,
  author =       "Anonymous",
  title =        {{Programming Environments for High-Level Scientific
```

```
                Problem Solving. {IFIP} {TC2}\slash {WG} 2.5 Working
                Conference}},
  journal =     j-IFIP-TRANS-A,
  volume =      "A-2",
  pages =       "??--??",
  year =        "1992",
  CODEN =       "ITATEC",
  ISSN =        "0926-5473",
  bibdate =     "Tue Sep 17 06:41:20 MDT 1996",
  bibsource =   "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =    "23--27 Sept. 1991",
  conflocation = "Karlsruhe, Germany",
  confsponsor = "IFIP",
  fjournal =    "IFIP Transactions. A. Computer Science and
                Technology",
  pubcountry =  "Netherlands",
}
```

———-

— **Anonymous:1995:GAM** —

```
@Article{Anonymous:1995:GAM,
  author =      "Anonymous",
  title =       {{GAMM 94 Annual Meeting}},
  journal =     j-ZEIT-ANGE-MATH-PHYS,
  volume =      "75",
  number =      "suppl. 2",
  pages =       "",
  year =        "1995",
  CODEN =       "ZAMMAX",
  ISSN =        "0044-2267",
  bibdate =     "Fri Dec 29 12:46:02 MST 1995",
  bibsource =   "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =    "4--8 April 1994",
  conflocation = "Braunschweig, Germany",
  pubcountry =  "Germany",
}
```

———-

— **Proceedings** —

```
%%% ====================================================================
%%% Cross-referenced entries must come last:

@Proceedings{Petrick:1971:PSS,
  editor =      "S. R. Petrick",
  booktitle =   "{Proceedings of the second symposium on Symbolic and
                Algebraic Manipulation, March 23--25, 1971, Los
```

```
                Angeles, California}",
  title =       {{Proceedings of the second symposium on Symbolic and
                Algebraic Manipulation, March 23--25, 1971, Los
                Angeles, California}},
  publisher =   pub-ACM,
  address =     pub-ACM:adr,
  pages =       "x + 464",
  year =        "1971",
  LCCN =        "QA76.5 .S94 1971",
  bibdate =     "Sat Dec 30 08:56:27 1995",
  bibsource =   "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  xxISBN =      "none",
}

@Proceedings{Online:1972:OCP,
  key =         "Online'72",
  booktitle =   "{Online 72: conference proceedings \ldots{}
                international conference on online interactive
                computing, Brunel University, Uxbridge, England, 4--7
                September 1972}",
  title =       {{Online 72: conference proceedings \ldots{}
                international conference on online interactive
                computing, Brunel University, Uxbridge, England, 4--7
                September 1972}},
  publisher =   "Online Computer Systems Ltd",
  address =     "Uxbridge, England",
  pages =       "various",
  month =       sep,
  year =        "1972",
  ISBN =        "0-903796-02-3",
  ISBN-13 =     "978-0-903796-02-6",
  LCCN =        "QA76.55 .O54 1972",
  bibdate =     "Fri Dec 29 18:31:29 1995",
  bibsource =   "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =        "Two volumes.",
  acknowledgement = ack-nhfb,
}

@Proceedings{Golden:1984:PMU,
  editor =      "V. Ellen Golden and M. A. Hussain",
  booktitle =   "{Proceedings of the 1984 MACSYMA Users' Conference:
                Schenectady, New York, July 23--25, 1984}",
  title =       {{Proceedings of the 1984 MACSYMA Users' Conference:
                Schenectady, New York, July 23--25, 1984}},
  publisher =   "General Electric",
  address =     "Schenectady, NY, USA",
  pages =       "xv + 567",
  year =        "1984",
  bibdate =     "Sat Dec 30 09:01:01 1995",
  bibsource =   "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
}
```

```
@Proceedings{Fitch:1984:E,
  editor =       "J. P. Fitch",
  booktitle =    "{EUROSAM '84: International Symposium on Symbolic and
                 Algebraic Computation, Cambridge, England, July 9--11,
                 1984}",
  title =        {{EUROSAM '84: International Symposium on Symbolic and
                 Algebraic Computation, Cambridge, England, July 9--11,
                 1984}},
  volume =       "174",
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "xi + 396",
  year =         "1984",
  ISBN =         "0-387-13350-X",
  ISBN-13 =      "978-0-387-13350-8",
  LCCN =         "QA155.7.E4 I57 1984",
  bibdate =      "Fri Dec 29 18:17:16 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  series =       "Lecture Notes In Computer Science",
  acknowledgement = ack-nhfb,
}

@Proceedings{Buchberger:1985:EEC,
  editor =       "Bruno Buchberger and Bob F. Caviness",
  booktitle =    "{EUROCAL '85: European Conference on Computer Algebra,
                 Linz, Austria, April 1--3, 1985: proceedings}",
  title =        {{EUROCAL '85: European Conference on Computer Algebra,
                 Linz, Austria, April 1--3, 1985: proceedings}},
  volume =       "204",
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "various",
  year =         "1985",
  ISBN =         "0-387-15983-5 (vol. 1), 0-387-15984-3 (vol. 2)",
  ISBN-13 =      "978-0-387-15983-6 (vol. 1), 978-0-387-15984-3 (vol.
                 2)",
  LCCN =         "QA155.7.E4 E86 1985",
  bibdate =      "Fri Dec 29 18:07:46 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "Two volumes.",
  series =       "Lecture Notes in Computer Science",
  acknowledgement = ack-nhfb,
}

@Proceedings{Wexelblat:1987:IIT,
  editor =       "Richard L. Wexelblat",
  booktitle =    "{Proceedings of the SIGPLAN '87 Symposium on
                 Interpreters and Interpretive Techniques, St. Paul,
                 Minnesota, June 24--26, 1987}",
  title =        {{Proceedings of the SIGPLAN '87 Symposium on
                 Interpreters and Interpretive Techniques, St. Paul,
                 Minnesota, June 24--26, 1987}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
```

```
  pages =        "vii + 291",
  year =         "1987",
  ISBN =         "0-89791-235-7",
  ISBN-13 =      "978-0-89791-235-8",
  LCCN =         "QA76.7 .S54 v.22:7",
  bibdate =      "Thu Jan 04 18:40:07 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "SIGPLAN Notices, vol. 22, no. 7 (July 1987).",
  price =        "US\$23.00",
  acknowledgement = ack-nhfb,
  keywords =     "design; interpreters (computer programs) ---
                 congresses; languages",
  subject =      "D.0 Software, GENERAL",
}

@Proceedings{Janssen:1988:TCA,
  editor =       "R. Jan{\ss}en",
  booktitle =    "{Trends in Computer Algebra, International Symposium
                 Bad Neuenahr, May 19--21, 1987, Proceedings}",
  title =        {{Trends in Computer Algebra, International Symposium
                 Bad Neuenahr, May 19--21, 1987, Proceedings}},
  volume =       "296",
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "??",
  year =         "1988",
  ISBN =         "3-540-18928-9, 0-387-18928-9",
  ISBN-13 =      "978-3-540-18928-2, 978-0-387-18928-4",
  LCCN =         "QA155.7.E4T74 1988",
  bibsource =    "/usr/local/src/bib/bibliography/Theory/Comp.Alg.bib;
                 http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  series =       "Lecture Notes in Computer Science",
  notes =        "{\footnotesize Die Beitr{\"a}ge in diesem Band geben
                 einen guten {\"U}berblick {\"u}ber den aktuellen Stand
                 der Forschung in verschiedenen f{\"u}r die \CA\
                 wichtigen Teilgebieten, so z.B. Faktorisierung von
                 Polynomen, konstruktive Galois Theory,
                 Termersetzungssysteme und das System Scratchpad II.
                 \hfill F. Schwarz}",
}

@Proceedings{ACM:1989:PAI,
  editor =       "{ACM}",
  booktitle =    "{Proceedings of the ACM-SIGSAM 1989 International
                 Symposium on Symbolic and Algebraic Computation, ISSAC
                 '89}",
  title =        {{Proceedings of the ACM-SIGSAM 1989 International
                 Symposium on Symbolic and Algebraic Computation, ISSAC
                 '89}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "399",
  year =         "1989",
  ISBN =         "0-89791-325-6",
```

```
  ISBN-13 =      "978-0-89791-325-6",
  LCCN =         "QA76.95.I59 1989",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "17--19 July 1989",
  conflocation = "Portland, OR, USA",
  confsponsor =  "ACM",
  pubcountry =   "USA",
}

@Proceedings{Huguet:1989:AAA,
  editor =       "L. Huguet and A. Poli",
  booktitle =    "{Applied Algebra, Algebraic Algorithms and
                 Error-Correcting Codes. 5th International Conference,
                 AAECC-5 Proceedings}",
  title =        {{Applied Algebra, Algebraic Algorithms and
                 Error-Correcting Codes. 5th International Conference,
                 AAECC-5 Proceedings}},
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "417",
  year =         "1989",
  ISBN =         "3-540-51082-6",
  ISBN-13 =      "978-3-540-51082-6",
  LCCN =         "QA268.A35 1987",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "15--19 June 1987",
  conflocation = "Menorca, Spain",
  pubcountry =   "West Germany",
}

@Proceedings{Davenport:1989:EEC,
  editor =       "J. H. Davenport",
  booktitle =    "{EUROCAL '87. European Conference on Computer Algebra
                 Proceedings}",
  title =        {{EUROCAL '87. European Conference on Computer Algebra
                 Proceedings}},
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "viii + 499",
  year =         "1989",
  ISBN =         "3-540-51517-8",
  ISBN-13 =      "978-3-540-51517-3",
  LCCN =         "QA155.7.E4E86 1987",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "2--5 June 1987",
  conflocation = "Leipzig, East Germany",
  confsponsor =  "Robotron; Rank Xerox",
  pubcountry =   "West Germany",
```

```
}

@Proceedings{Gianni:1989:SAC,
  editor =       "P. (Patrizia) Gianni",
  booktitle =    "{Symbolic and Algebraic Computation. International
                 Symposium ISSAC '88, Rome, Italy, July 4--8, 1988.
                 Proceedings}",
  title =        {{Symbolic and Algebraic Computation. International
                 Symposium ISSAC '88, Rome, Italy, July 4--8, 1988.
                 Proceedings}},
  volume =       "358",
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "xi + 543",
  year =         "1989",
  ISBN =         "3-540-51084-2",
  ISBN-13 =      "978-3-540-51084-0",
  LCCN =         "QA76.95 .I57 1988",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "Conference held jointly with AAECC-6.",
  series =       "Lecture Notes In Computer Science",
  acknowledgement = ack-nhfb,
  confdate =     "4--8 July 1988",
  conflocation = "Rome, Italy",
  pubcountry =   "West Germany",
}

@Proceedings{Mora:1989:AAA,
  editor =       "T. Mora",
  booktitle =    "{Applied Algebra, Algebraic Algorithms and
                 Error-Correcting Codes. 6th International Conference,
                 AAECC-6, Rome, Italy, July 4--8, 1988. Proceedings}",
  title =        {{Applied Algebra, Algebraic Algorithms and
                 Error-Correcting Codes. 6th International Conference,
                 AAECC-6, Rome, Italy, July 4--8, 1988. Proceedings}},
  volume =       "357",
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "ix + 480",
  year =         "1989",
  ISBN =         "3-540-51083-4",
  ISBN-13 =      "978-3-540-51083-3",
  LCCN =         "QA268 .A35 1988",
  bibdate =      "Tue Sep 17 06:46:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "Conference held jointly with ISSAC '88.",
  series =       "Lecture Notes in Computer Science",
  acknowledgement = ack-nhfb,
  confdate =     "4--8 July 1988",
  conflocation = "Rome, Italy",
  pubcountry =   "West Germany",
}
```

```
@Proceedings{Watanabe:1990:IPI,
  editor =       "Shunro Watanabe and Morio Nagata",
  booktitle =    "{ISSAC '90. Proceedings of the International Symposium
                 on Symbolic and Algebraic Computation}",
  title =        {{ISSAC '90. Proceedings of the International Symposium
                 on Symbolic and Algebraic Computation}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "ix + 307",
  year =         "1990",
  ISBN =         "0-89791-401-5",
  ISBN-13 =      "978-0-89791-401-7",
  LCCN =         "QA76.95 .I57 1990",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "20--24 Aug. 1990",
  conflocation = "Tokyo, Japan",
  confsponsor =  "Inf. Processing Soc. Japan; Japan Soc. Software Sci.
                 Technol.; ACM",
  pubcountry =   "USA",
}

@Proceedings{Miola:1990:DIS,
  editor =       "A. Miola",
  booktitle =    "{Design and Implementation of Symbolic Computation
                 Systems, International Symposium DISCO '90, Capri,
                 Italy, April 10--12, 1990, Proceedings}",
  title =        {{Design and Implementation of Symbolic Computation
                 Systems, International Symposium DISCO '90, Capri,
                 Italy, April 10--12, 1990, Proceedings}},
  volume =       "429",
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "xii + 283",
  year =         "1990",
  ISBN =         "0-387-52531-9 (New York), 3-540-52531-9 (Berlin)",
  ISBN-13 =      "978-0-387-52531-0 (New York), 978-3-540-52531-8
                 (Berlin)",
  LCCN =         "QA76.9.S88I576 1990",
  bibdate =      "Tue Sep 17 06:44:07 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  series =       "Lecture Notes in Computer Science",
  acknowledgement = ack-nhfb,
  confdate =     "10--12 April 1990",
  conflocation = "Capri, Italy",
  pubcountry =   "West Germany",
}

@Proceedings{Cohen:1991:EIS,
  editor =       "G. Cohen and P. Charpin",
  booktitle =    "{EUROCODE '90. International Symposium on Coding
                 Theory and Applications Proceedings}",
  title =        {{EUROCODE '90. International Symposium on Coding
```

```
                Theory and Applications Proceedings}},
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "xi + 392",
  year =         "1991",
  ISBN =         "0-387-54303-1 (New York), 3-540-54303-1 (Berlin)",
  ISBN-13 =      "978-0-387-54303-1 (New York), 978-3-540-54303-9
                 (Berlin)",
  LCCN =         "QA268.E95 1990",
  bibdate =      "Tue Sep 17 06:41:20 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "5--9 Nov. 1990",
  conflocation = "Udine, Italy",
  pubcountry =   "Germany",
}

@Proceedings{Watt:1991:PIS,
  editor =       "Stephen M. Watt",
  booktitle =    "{Proceedings of the 1991 International Symposium on
                 Symbolic and Algebraic Computation, ISSAC'91, July
                 15--17, 1991, Bonn, Germany}",
  title =        {{Proceedings of the 1991 International Symposium on
                 Symbolic and Algebraic Computation, ISSAC'91, July
                 15--17, 1991, Bonn, Germany}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "xiii + 468",
  year =         "1991",
  ISBN =         "0-89791-437-6",
  ISBN-13 =      "978-0-89791-437-6",
  LCCN =         "QA76.95.I59 1991",
  bibdate =      "Fri Dec 29 18:17:57 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
}

@Proceedings{Anonymous:1991:PAC,
  editor =       "Anonymous",
  booktitle =    "{Proceedings 1991 Annual Conference, American Society
                 for Engineering Education. Challenges of a Changing
                 World}",
  title =        {{Proceedings 1991 Annual Conference, American Society
                 for Engineering Education. Challenges of a Changing
                 World}},
  publisher =    "ASEE",
  address =      "Washington, DC, USA",
  pages =        "xxi + 2026",
  year =         "1991",
  bibdate =      "Tue Sep 17 06:37:45 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "2 vol.",
  acknowledgement = ack-nhfb,
  confdate =     "16--19 June 1991",
```

```
  conflocation = "New Orleans, LA, USA",
  pubcountry =   "USA",
}

@Proceedings{Wang:1992:ISS,
  editor =       "Paul S. Wang",
  booktitle =    "{International System Symposium on Symbolic and
                 Algebraic Computation 92}",
  title =        {{International System Symposium on Symbolic and
                 Algebraic Computation 92}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "ix + 406",
  year =         "1992",
  ISBN =         "0-89791-489-9 (soft cover), 0-89791-490-2 (hard
                 cover)",
  ISBN-13 =      "978-0-89791-489-5 (soft cover), 978-0-89791-490-1
                 (hard cover)",
  LCCN =         "QA76.95.I59 1992",
  bibdate =      "Tue Sep 17 06:35:39 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "27--29 July 1992",
  conflocation = "Berkeley, CA, USA",
  confsponsor =  "ACM",
  pubcountry =   "USA",
}

@Proceedings{Bronstein:1993:IPI,
  editor =       "Manuel Bronstein",
  booktitle =    "{ISSAC'93: proceedings of the 1993 International
                 Symposium on Symbolic and Algebraic Computation, July
                 6--8, 1993, Kiev, Ukraine}",
  title =        {{ISSAC'93: proceedings of the 1993 International
                 Symposium on Symbolic and Algebraic Computation, July
                 6--8, 1993, Kiev, Ukraine}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "viii + 321",
  year =         "1993",
  ISBN =         "0-89791-604-2",
  ISBN-13 =      "978-0-89791-604-2",
  LCCN =         "QA 76.95 I59 1993",
  bibdate =      "Thu Sep 26 05:45:15 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "ACM order number: 505930.",
  abstract =     "The following topics were dealt with: algebraic
                 solutions of equations; computer algebra systems;
                 algorithm theory and complexity; automated theorem
                 proving; polynomials; and matrix algebra.",
  acknowledgement = ack-nhfb,
  classification = "C4210 (Formal logic); C4240 (Programming and
                 algorithm theory); C7310 (Mathematics computing)",
  confdate =     "6--8 July 1993",
```

```
  conflocation = "Kiev, Ukraine",
  confsponsor =  "ACM",
  keywords =     "algebra --- data processing --- congresses; Algorithm
                 theory; Automated theorem proving; Complexity; Computer
                 algebra; mathematics --- data processing ---
                 congresses; Matrix algebra; Polynomials",
  pubcountry =   "USA",
  source =       "ISSAC '93",
  sponsor =      "Association for Computing Machinery.",
  thesaurus =    "Computational complexity; Mathematics computing;
                 Matrix algebra; Polynomials; Symbol manipulation;
                 Theorem proving",
}

@Proceedings{Fitch:1993:DIS,
  editor =       "J. Fitch",
  booktitle =    "{Design and Implementation of Symbolic Computation
                 Systems International Symposium, DISCO '92
                 Proceedings}",
  title =        {{Design and Implementation of Symbolic Computation
                 Systems International Symposium, DISCO '92
                 Proceedings}},
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "214",
  year =         "1993",
  ISBN =         "0-387-57272-4 (New York), 3-540-57272-4 (Berlin)",
  ISBN-13 =      "978-0-387-57272-7 (New York), 978-3-540-57272-5
                 (Berlin)",
  LCCN =         "QA76.9.S88I576 1992",
  bibdate =      "Tue Sep 17 06:37:45 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "13--15 April 1992",
  conflocation = "Bath, UK",
  pubcountry =   "Germany",
}

@Proceedings{Jacob:1993:PSI,
  editor =       "G. Jacob and N. E. Oussous and S. Steinberg",
  booktitle =    "{Proceedings SC 93. International IMACS Symposium on
                 Symbolic Computation. New Trends and Developments}",
  title =        {{Proceedings SC 93. International IMACS Symposium on
                 Symbolic Computation. New Trends and Developments}},
  publisher =    "LIFL Univ. Lille",
  address =      "Lille, France",
  pages =        "vii + 239",
  year =         "1993",
  bibdate =      "Tue Sep 17 06:35:39 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "14--17 June 1993",
  conflocation = "Lille, France",
  confsponsor =  "IMACS-AICA",
```

```
  pubcountry =   "France",
}

@Proceedings{Miola:1993:DIS,
  editor =       "A. Miola",
  booktitle =    "{Design and Implementation of Symbolic Computation
                 Systems International Symposium. DISCO '93 Gmunden,
                 Austria, September 15--17, 1993: Proceedings}",
  title =        {{Design and Implementation of Symbolic Computation
                 Systems International Symposium. DISCO '93 Gmunden,
                 Austria, September 15--17, 1993: Proceedings}},
  publisher =    pub-SV,
  address =      pub-SV:adr,
  pages =        "xi + 383",
  year =         "1993",
  ISBN =         "3-540-57235-X",
  ISBN-13 =      "978-3-540-57235-0",
  LCCN =         "QA76.9.S88I576 1993",
  bibdate =      "Fri Dec 29 12:46:02 MST 1995",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "15--17 Sept. 1993",
  conflocation = "Gmunden, Austria",
  pubcountry =   "Germany",
}

@Proceedings{ACM:1994:IPI,
  editor =       "{ACM}",
  booktitle =    "{ISSAC'94. Proceedings of the International Symposium
                 on Symbolic and Algebraic Computation}",
  title =        {{ISSAC'94. Proceedings of the International Symposium
                 on Symbolic and Algebraic Computation}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "ix + 359",
  year =         "1994",
  ISBN =         "0-89791-638-7",
  ISBN-13 =      "978-0-89791-638-7",
  LCCN =         "QA76.95.I59 1994",
  bibdate =      "Tue Sep 17 06:29:18 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "20--22 July 1994",
  conflocation = "Oxford, UK",
  confsponsor =  "ACM",
  pubcountry =   "USA",
}

@Proceedings{Calmet:1994:RWC,
  editor =       "J. Calmet",
  booktitle =    "{Rhine Workshop on Computer Algebra. Proceedings}",
  title =        {{Rhine Workshop on Computer Algebra. Proceedings}},
  publisher =    "Universit{\"a}t Karlsruhe",
  address =      "Karlsruhe, Germany",
```

```
  pages =        "v + 224",
  year =         "1994",
  bibdate =      "Tue Sep 17 06:32:41 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
  confdate =     "22--24 March 1994",
  conflocation = "Karlsruhe, Germany",
  confsponsor =  "Universit{\"a}t Karlsruhe",
  pubcountry =   "Germany",
}

@Proceedings{Levelt:1995:IPI,
  editor =       "A. H. M. Levelt",
  booktitle =    "{ISSAC '95: Proceedings of the 1995 International
                 Symposium on Symbolic and Algebraic Computation: July
                 10--12, 1995, Montr{\'e}al, Canada}",
  title =        {{ISSAC '95: Proceedings of the 1995 International
                 Symposium on Symbolic and Algebraic Computation: July
                 10--12, 1995, Montr{\'e}al, Canada}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "xviii + 314",
  year =         "1995",
  ISBN =         "0-89791-699-9",
  ISBN-13 =      "978-0-89791-699-8",
  LCCN =         "QA 76.95 I59 1995",
  bibdate =      "Thu Sep 26 05:34:21 MDT 1996",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  note =         "ACM order number: 505950",
  series =       "ISSAC -PROCEEDINGS- 1995",
  abstract =     "The following topics were dealt with: differential
                 equations; visualisation; algebraic numbers;
                 algorithms; systems; polynomial and differential
                 algebra; seminumerical methods; greatest common
                 divisors; and.",
  acknowledgement = ack-nhfb,
  classification = "C4100 (Numerical analysis); C4170 (Differential
                 equations); C7310 (Mathematics computing)",
  confdate =     "10--12 July 1995",
  conflocation = "Montr{\'e}al, Que., Canada",
  confsponsor =  "ACM",
  keywords =     "algebra --- data processing --- congresses; Algebraic
                 numbers; Algorithms; Differential algebra; Differential
                 equations; Greatest common divisors; mathematics ---
                 data processing --- congresses; Polynomial;
                 Seminumerical methods; Systems; Visualisation",
  pubcountry =   "USA",
  source =       "ISSAC '95",
  thesaurus =    "Data visualisation; Differential equations; Group
                 theory; Numerical analysis; Symbol manipulation",
}

@Proceedings{Dooley:1999:IJS,
  editor =       "Sam Dooley",
```

```
  booktitle =    "{ISSAC 99: July 29--31, 1999, Simon Fraser University,
                 Vancouver, BC, Canada: proceedings of the 1999
                 International Symposium on Symbolic and Algebraic
                 Computation}",
  title =        {{ISSAC 99: July 29--31, 1999, Simon Fraser University,
                 Vancouver, BC, Canada: proceedings of the 1999
                 International Symposium on Symbolic and Algebraic
                 Computation}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "xxii + 311",
  year =         "1999",
  ISBN =         "1-58113-073-2",
  ISBN-13 =      "978-1-58113-073-7",
  LCCN =         "QA76.95 .I57 1999",
  bibdate =      "Sat Mar 11 16:51:59 2000",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib",
  acknowledgement = ack-nhfb,
}

@Proceedings{Brown:2007:PIS,
  editor =       "C. W. Brown",
  booktitle =    "{Proceedings of the 2007 International Symposium on
                 Symbolic and Algebraic Computation, July 29--August 1,
                 2007, University of Waterloo, Waterloo, Ontario,
                 Canada}",
  title =        {{Proceedings of the 2007 International Symposium on
                 Symbolic and Algebraic Computation, July 29--August 1,
                 2007, University of Waterloo, Waterloo, Ontario,
                 Canada}},
  publisher =    pub-ACM,
  address =      pub-ACM:adr,
  pages =        "????",
  year =         "2007",
  ISBN =         "1-59593-743-9 (print), 1-59593-742-0 (CD-ROM)",
  ISBN-13 =      "978-1-59593-743-8 (print), 978-1-59593-742-1
                 (CD-ROM)",
  LCCN =         "QA76.5 S98 2007",
  bibdate =      "Fri Jun 20 08:53:37 2008",
  bibsource =    "http://www.math.utah.edu/pub/tex/bib/axiom.bib;
                 http://www.math.utah.edu/pub/tex/bib/issac.bib;
                 http://www.math.utah.edu/pub/tex/bib/maple-extract.bib",
  note =         "ACM order number 505070.",
  acknowledgement = ack-nhfb,
}

@Proceedings{Shi:2007:CSIb,
  editor =       "Yong Shi and Geert Dick van Albada and Jack Dongarra
                 and Peter M. A. Sloot",
  booktitle =    "{Computational Science --- ICCS 2007: 7th
                 International Conference, Beijing, China, May 27 ---
                 30, 2007, Proceedings, Part II}",
  title =        {{Computational Science --- ICCS 2007: 7th
                 International Conference, Beijing, China, May 27 ---
```

```
                30, 2007, Proceedings, Part II}},
  volume =      "4488",
  publisher =   pub-SV,
  address =     pub-SV:adr,
  pages =       "153 (est.)",
  year =        "2007",
  CODEN =       "LNCSD9",
  DOI =         "http://dx.doi.org/10.1007/978-3-540-72586-2",
  ISBN =        "3-540-72585-7 (print), 3-540-72586-5 (e-book)",
  ISBN-13 =     "978-3-540-72585-5 (print), 978-3-540-72586-2
                (e-book)",
  ISSN =        "0302-9743 (print), 1611-3349 (electronic)",
  ISSN-L =      "0302-9743",
  LCCN =        "????",
  bibdate =     "Wed Dec 19 15:19:26 MST 2012",
  bibsource =   "http://www.math.utah.edu/pub/tex/bib/axiom.bib;
                http://www.math.utah.edu/pub/tex/bib/lncs.bib;
                http://www.math.utah.edu/pub/tex/bib/magma.bib;
                http://www.math.utah.edu/pub/tex/bib/maple-extract.bib",
  series =      ser-LNCS,
  link =         "\url{http://www.springerlink.com/content/978-3-540-72586-2}",
  acknowledgement = ack-nhfb,
}
```

Index