COMP 360C: Computer-Checked Programs and Proofs

Dan Licata


LUXURY
BRAND
Sunkist Oranges
REG. U. S. PAT. OFF.
GROWN IN U.S.A.
CALIFORNIA FRUITGROWERS EXCHANGE
BOX
The
I Promised You
from California.

Kepler Conjecture (1611)

No way to pack equally-sized spheres in space has higher density than

Hales' proof (1998)

- Reduces Kepler Conjecture to proving that a function is no smaller than some value on 5,000 different configurations of spheres
- This requires solving 100,000 equations
- 1998 submission:
  300 pages of math
  + 50,000 LOC (revised 2006: 15,000 LOC)

Journal refereeing

In 2003, after 4 years' work,
12 referees had checked lots of lemmas,
but gave up on verifying the proof

Journal refereeing

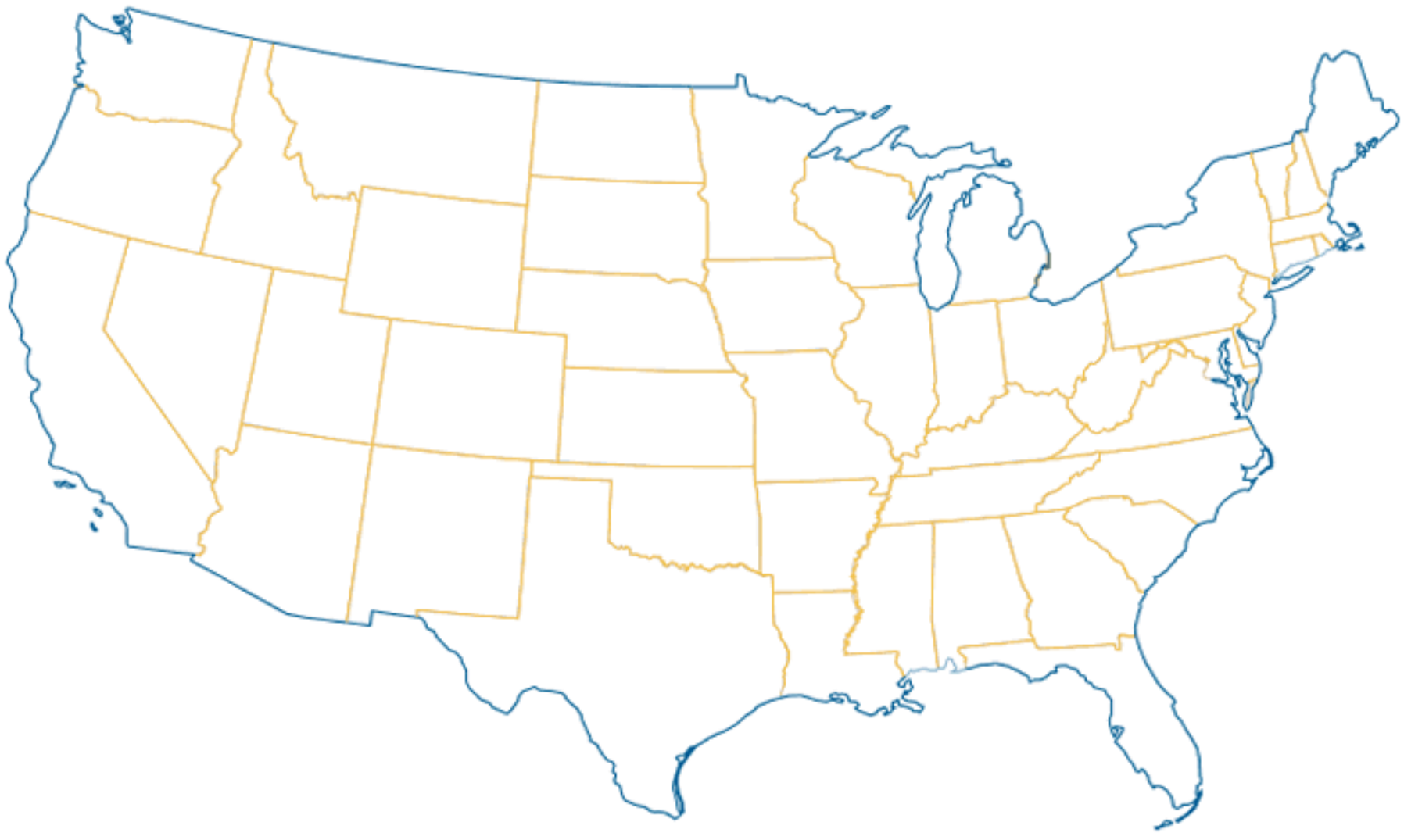
In 2003, after 4 years' work,
12 referees had checked lots of lemmas,
but gave up on verifying the proof

"This paper has brought about a change in the journal's policy on computer proof. ***It will no longer attempt to check the correctness of computer code.***"

Map Coloring

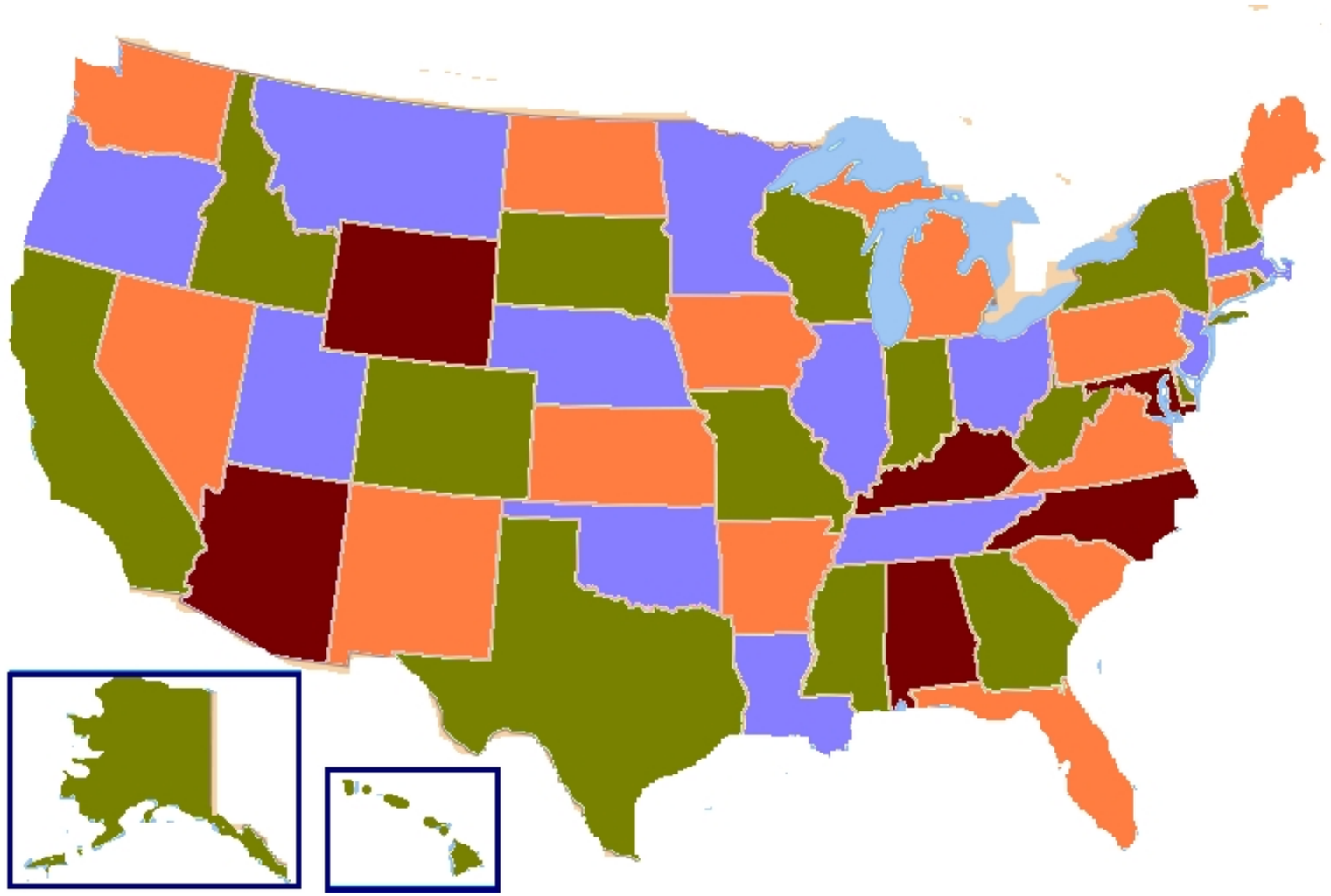
Two adjacent countries must have different colors

Map Coloring



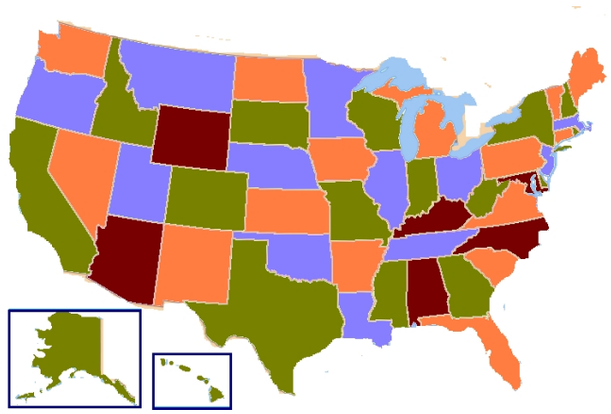
Two adjacent (non-corner) states must have different colors

Map Coloring



Two adjacent (non-corner) states must have different colors

4 Color Theorem

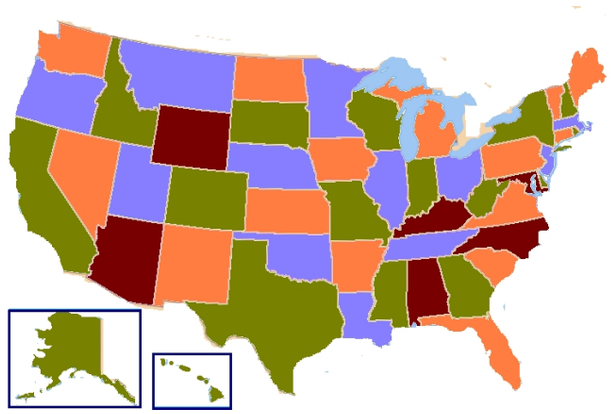


Every map can be colored with at most 4 colors

Proof (Appel&Haken,1976):

- There is a set of 1,936 maps, each of which cannot be part of the smallest counterexample
- Any counterexample has one of these maps as part of it

4 Color Theorem



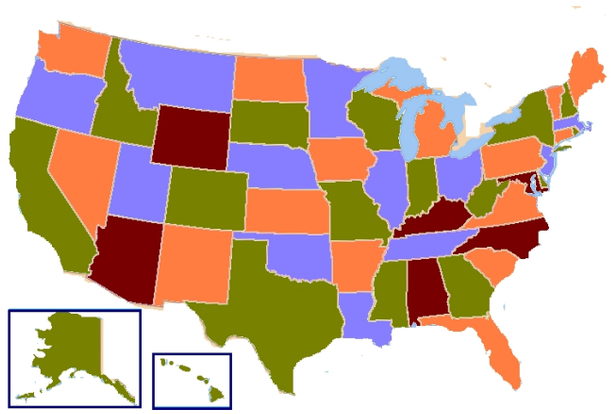
Every map can be colored with at most 4 colors

Proof (Appel&Haken,1976):

- There is a set of 1,936 maps, each of which cannot be part of the smallest counterexample
- Any counterexample has one of these maps as part of it

found by computer

4 Color Theorem

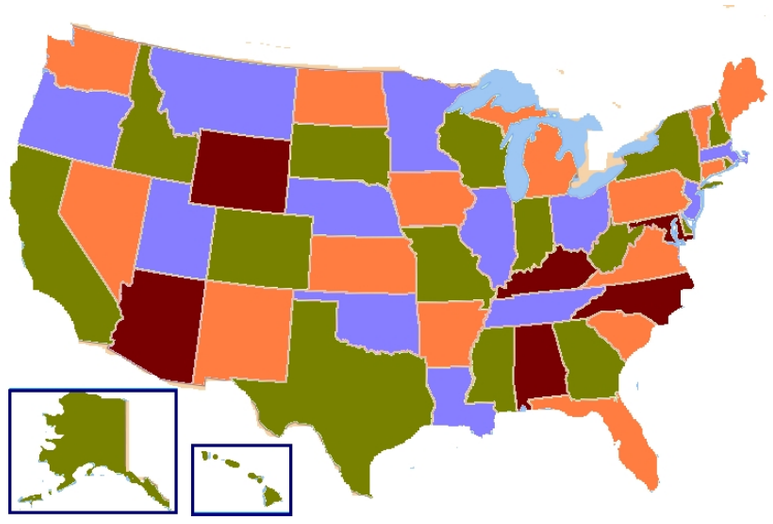


Every map can be colored with at most 4 colors

Proof (Appel&Haken,1976):

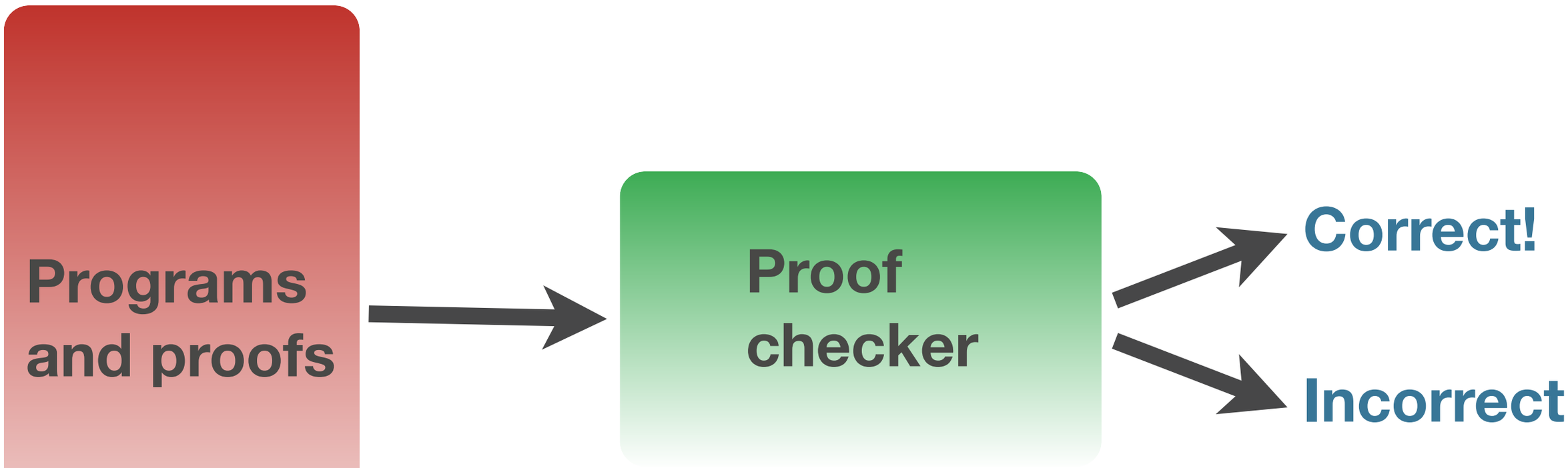
- There is a set of 1,936 maps, each of which cannot be part of the smallest counterexample — **found by computer**
- Any counterexample has one of these maps as part of it — **traditional proof**

“Cyborg” Proofs



- Proofs have both a traditional mathematical part and a computational part
- Need to check the math
- Need to check correctness of the code

Computer-checked math



Kepler Conjecture

flyspeck
The Flyspeck Project

Project Home | Downloads | Wiki | Issues | Source

Summary | People

Project Information

Starred by 21 users
Project feeds

Code license
MIT License

Labels
theoremproving,
keplerconjecture

Members
TCHa...@gmail.com,
seanmcl@gmail.com,
solovyev...@gmail.com,
jrh...@gmail.com
22 contributors

Featured

Wiki pages
FlyspeckFactSheet
Show all »

Links

Groups
Flyspeck disucssion

The Flyspeck Project

Introduction

The purpose of the flyspeck project is to produce a formal proof of the Kepler Conjecture. The name `flyspeck' comes from matching the pattern `/f.*p.*k/` against an English dictionary. FPK in turn is an acronym for "The Formal Proof of Kepler."

Book

The formalization project is based on the book "Dense Sphere Packings: a formal blueprint", published by Cambridge University Press, and available here.

Internal Links

- Flyspeck Wiki
- The text part of the formalization is complete!
- What's left in the computer part of the formalization.
- Computer resources for the verification of Fejes Toth's Conjecture on Packings with Kissing Number Twelve.

External Links

- Hol-light source code repository
- 2009 Hanoi workshop
- McLaughlin's revision of the kepler code
- Flyspeck I: Tame Graphs G. Bauer and T. Nipkow
- Flyspeck II: Linear Programs S. Obua
- QED Manifesto
- Nonlinear inequality proving R. Zumkeller
- Learning-assisted Automated Reasoning with Flyspeck C. Kaliszyk and J. Urban
- Formal Verification of Nonlinear Inequalities with Taylor Interval Approximations T. Hales and A. Solovyev
- Formal Computations and Methods A. Solovyev

T. Hales acknowledges the support of the NSF through grant 0503447 on the "Formal Foundations of Discrete Geometry" and grant 0804189 on the "Formal Proof of the Kepler Conjecture". He gratefully acknowledges support from the Benter Foundation.

Four Color Theorem

A computer-checked proof of the Four Colour Theorem

Georges Gonthier
Microsoft Research Cambridge

This report gives an account of a successful formalization of the proof of the Four Colour Theorem, which was fully checked by the Coq v7.3.1 proof assistant [13]. This proof is largely based on the mixed mathematics/computer proof [26] of Robertson et *al*, but contains original contributions as well. This document is organized as follows: section 1 gives a historical introduction to the problem and positions our work in this setting; section 2 defines more precisely what was proved; section 3 explains the broad outline of the proof; section 4 explains how we exploited the features of the Coq assistant to conduct the proof, and gives a brief description of the tactic shell that we used to write our proof scripts; section 5 is a detailed account of the formal proof (for even more details the actual scripts can be consulted); section 6 is a chronological account of how the formal proof was developed; finally, we draw some general conclusions in section 7.

Odd Order Theorem

Theorem Proof Gains Acclaim

By Rob Knies
October 11, 2012 9:30 AM PT

At 5:46 p.m. on Sept. 20, Georges Gonthier, principal researcher at Microsoft Research Cambridge, sent a brief email to his colleagues at the Microsoft Research-Inria Joint Centre in Paris. It read, in full:

"This is really the End."

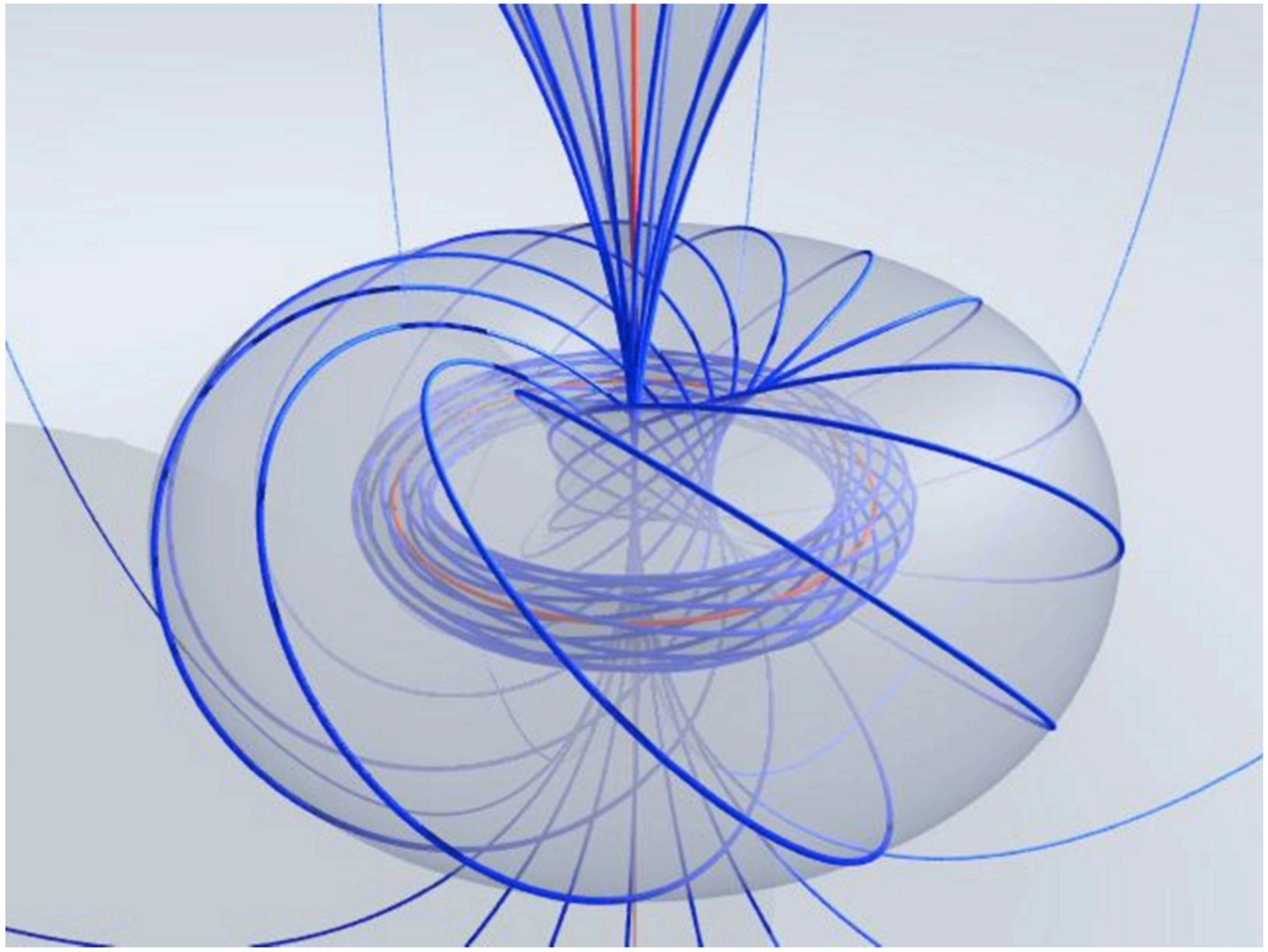
Georges Gonthier

Those five innocuous words heralded the culmination of a project that had consumed more than six years and resulted in the formal proof of the Feit-Thompson Theorem, the first major step of the classification of finite simple groups.

The theorem, first proved by Walter Feit and John Griggs Thompson in 1963 and also known as the Odd-Order Theorem, states that in mathematical group theory, every finite group of odd order is solvable.

That might seem a deceptively simple definition to non-mathematicians, but Gonthier and his collaborators on the Mathematical Components project at the Microsoft Research-INRIA Joint Centre are anything but. Their achievement in completing a computer-assisted proof by the Coq proof assistant, developed at Inria, the French National Research Institute for Computer Science and Applied Mathematics, was acclaimed widely as news spread.

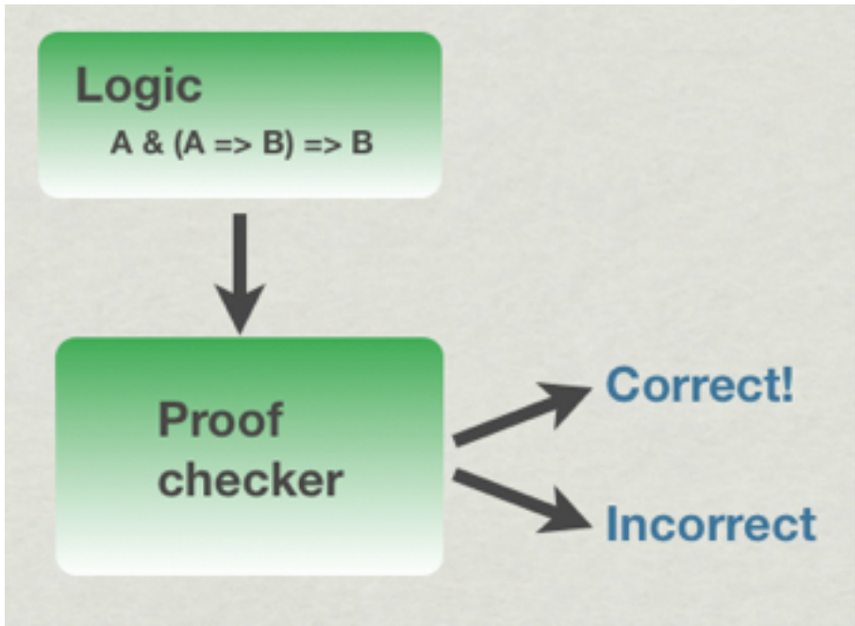
Homotopy theory



Computer-assisted proofs

Proof assistant

- Interactive proof editor
- Automated proofs
- Libraries



Computer-assisted proofs

- are much easier to believe:
  computer does the journal reviewing
- can use computational methods
  and still be fully rigorous
- broaden access:
  computer as gifted&talented teacher
- are easier to write?

Kepler proof

Informal

- 300 pages of math + 15,000 lines of code
- 15 hours to run

Computer-checked

- 350,000 lines of math + code
- >2 years to run

Kepler proof

Informal

- 300 pages of math + 15,000 lines of code
- 15 hours to run

Computer-checked

- 350,000 lines of math + code
- >2 years to run

~5-10x longer

Kepler proof

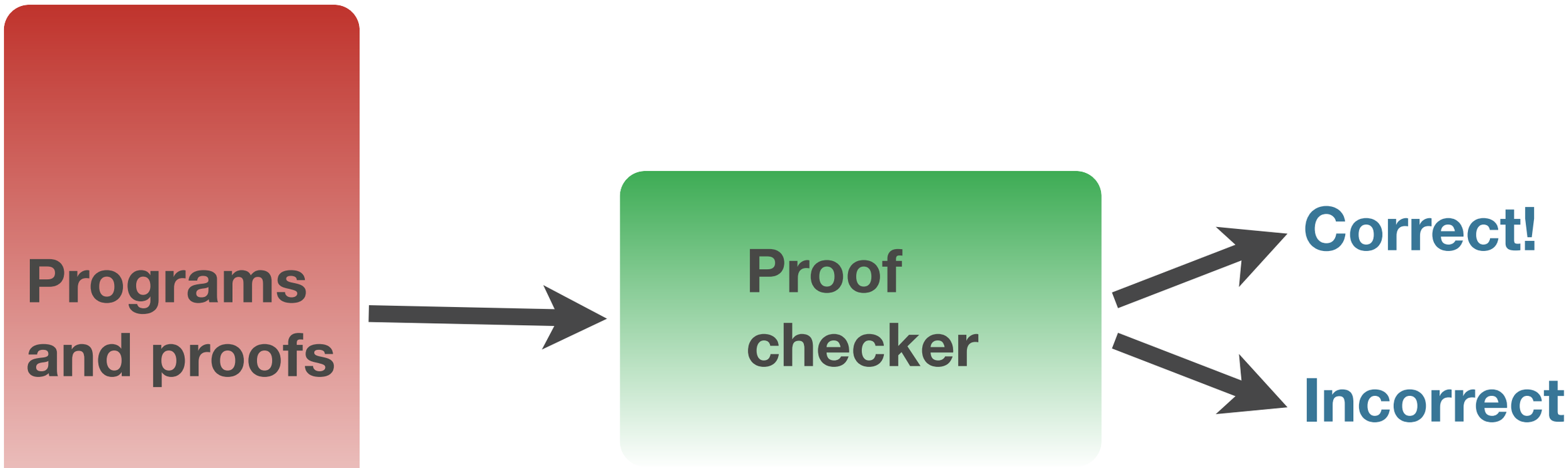
Informal

- 300 pages of math + 15,000 lines of code
- 15 hours to run

Computer-checked

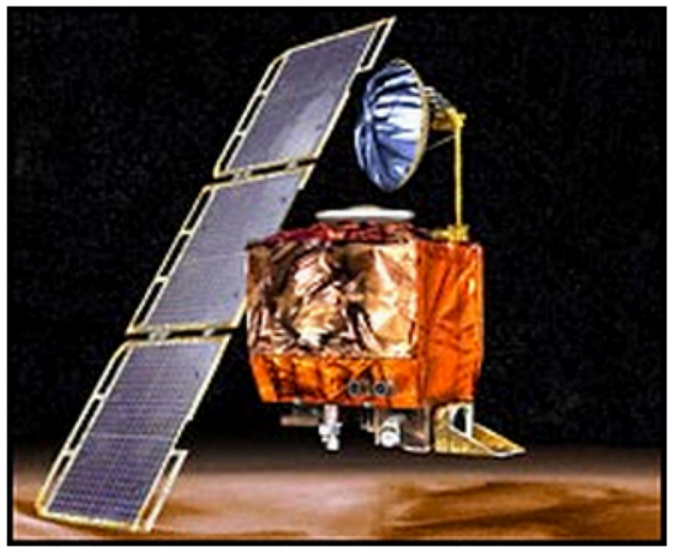
- 350,000 lines of math + code **~5-10x longer**
- >2 years to run **~2000x slower**

Computer-checked code



Mystery of Orbiter Crash Solved

By Kathy Sawyer
Washington Post Staff Writer
Friday, October 1, 1999; Page A1


Scientists do not yet know what caused the Mars Orbiter to crash. (AP)

NASA's Mars Climate Orbiter was lost in space last week because engineers failed to make a simple conversion from English units to metric, an embarrassing lapse that sent the $125 million craft fatally close to the Martian surface, investigators said yesterday.

Officials are scrambling to determine whether a similar error is buried in the computer files of two other spacecraft currently cruising through space: the Mars Polar Lander, scheduled to hit the Martian surface on Dec. 3, and the Stardust craft bound for a comet.

It now appears the error had affected the orbiter mission from its launching almost 10 months and 416 million miles before its Sept. 23 failure. And yet the problem was never caught and corrected by the system of checks and balances at the Jet Propulsion Laboratory (JPL) in California, which manages this and numerous other interplanetary missions for NASA.

As a result, flight controllers believe the spacecraft plowed into the Martian atmosphere, where the stresses crippled it, aborted its insertion into Martian orbit and most likely left it hurtling on through space in an orbit around the sun.

```
[<Measure>] type kg
[<Measure>] type s
[<Measure>] type m

let gravityOnEarth = 9.81<m/s^2> // an acceleration
let heightOfMyOfficeWindow = 3.5<m> // a distance

let speedOfImpact = sqrt (2.0 * gravityOnEarth * heightOfMyOfficeWindow)
```

```
val speedOfImpact : float<m/s>

Full name: Units1.speedOfImpact
```

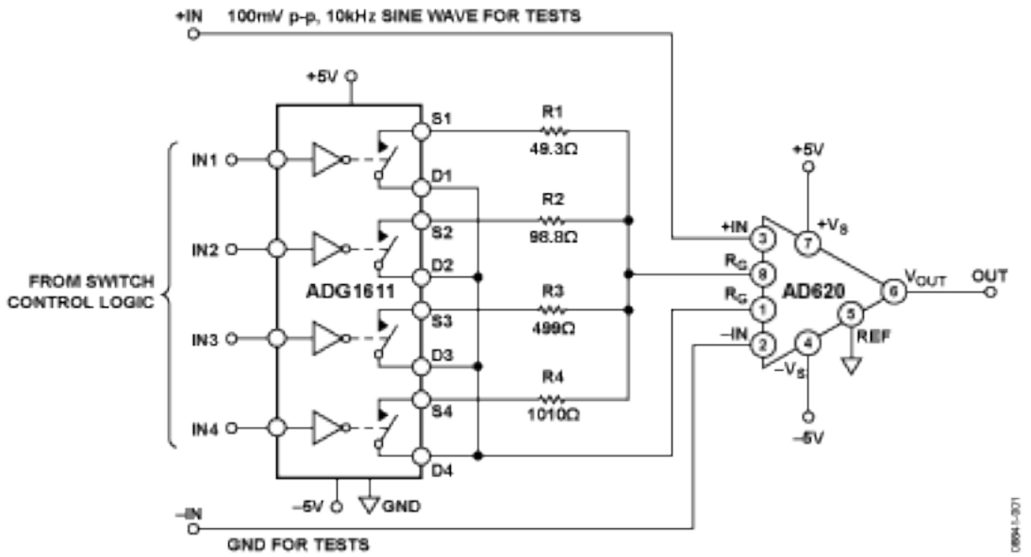
```
let speedOfImpact = sqrt (2.0 * gravityOnEarth + heightOfMyOfficeWindow)
```

```
The unit of measure 'm' does not match the unit of measure 'm/s ^ 2'
```


```
method Find(a: array<int>, key: int) returns (index: int)
   requires a != null
   ensures 0 <= index ==> index < a.Length && a[index] == key
   ensures index < 0 ==> forall k :: 0 <= k < a.Length ==> a[k] != key

   var low, high := 0, a.Length;
   while low < high
      invariant 0 <= low <= high <= a.Length
      invariant forall i ::
         0 <= i < a.Length && !(low <= i < high) ==> a[i] != value
   {
      var mid := (low + high) / 2;
      if a[mid] < value
      {
         low := mid + 1;
      }
      else if value < a[mid]
      {
         high := mid;
      }
      else
      {
         return mid;
      }
   }
   return -1;
```

ACL2



NOTES
1.R1, R2, R3 AND R4 ARE 0.1%.

The seL4 Microkernel

Security is no excuse for poor performance!

The world's first operating-system kernel with an end-to-end proof of implementation correctness and security enforcement is available as open source.

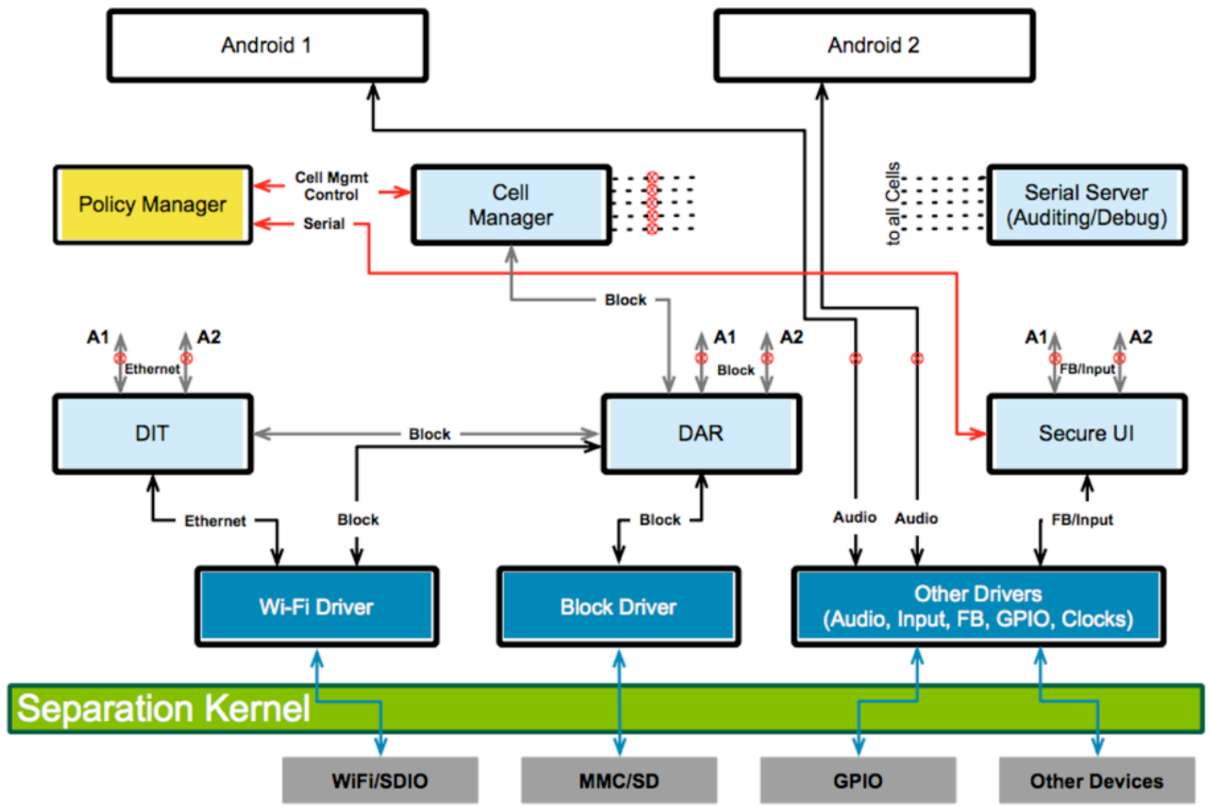
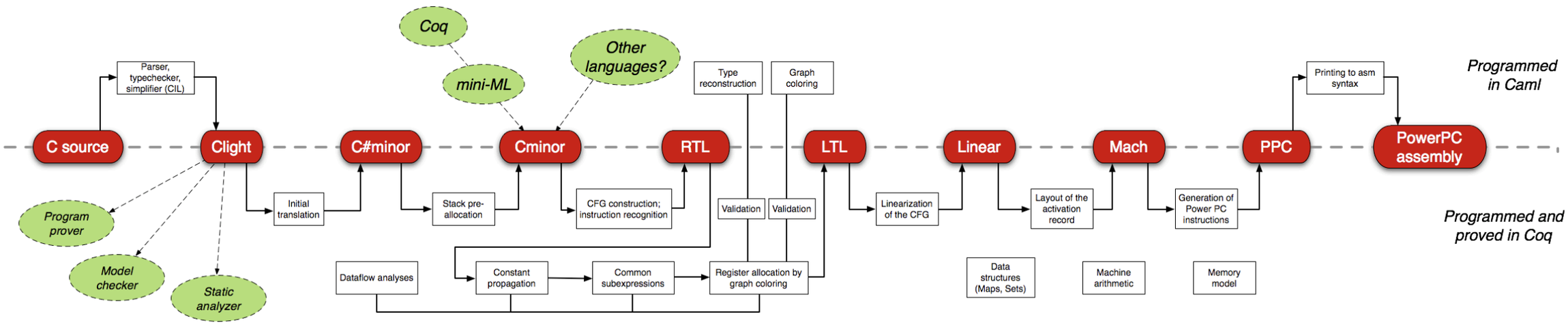
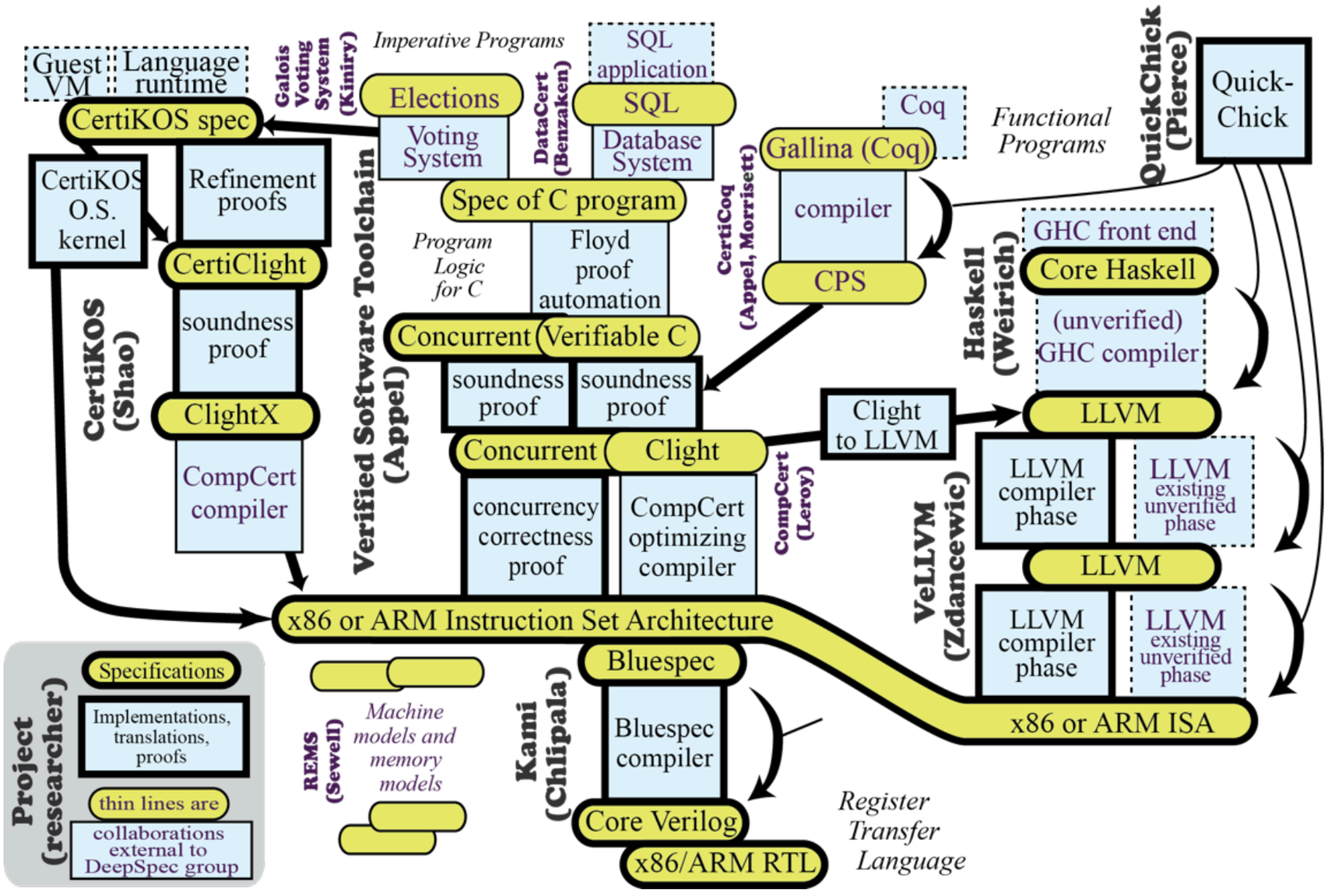


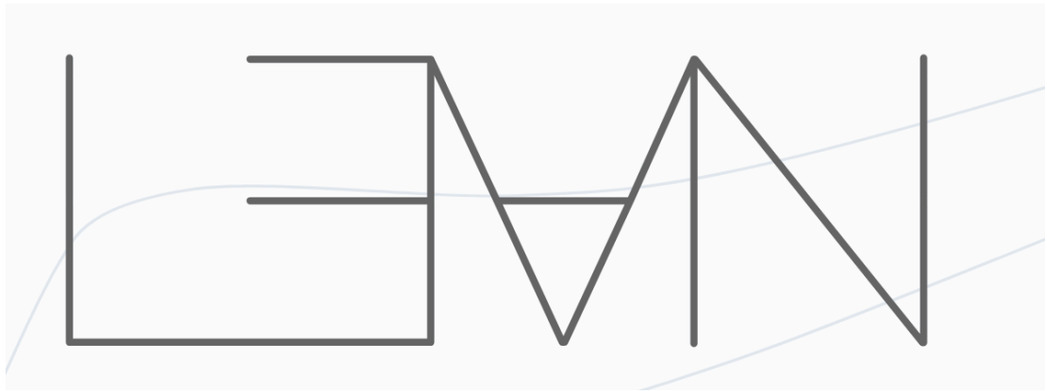
Figure 0: An example multi-persona Android platform built leveraging a separation kernel (OKL4 Microvisor) for componentization. This approach enables stronger policy controls, non-by-passable guaranteed invocation of encryption, and other high-assurance capabilities.

COMPCERT


Coq
mini-ML
Other languages?
Parser, typechecker, simplifier (CIL)
Type reconstruction
Graph coloring
Printing to asm syntax
Programmed in Caml
C source
Clight
C#minor
Cminor
RTL
LTL
Linear
Mach
PPC
PowerPC assembly
Program prover
Model checker
Static analyzer
Initial translation
Stack pre-allocation
CFG construction; instruction recognition
Validation
Validation
Linearization of the CFG
Layout of the activation record
Generation of Power PC instructions
Programmed and proved in Coq
Dataflow analyses
Constant propagation
Common subexpressions
Register allocation by graph coloring
Data structures (Maps, Sets)
Machine arithmetic
Memory model

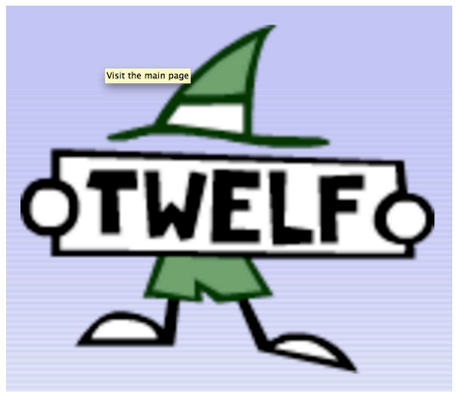
deep spec


Guest VM
Language runtime
CertiKOS spec
CertiKOS O.S. kernel
Refinement proofs
CertiClight
soundness proof
ClightX
CompCert compiler
CertiKOS (Shao)
Galois Voting System (Kiniry)
Imperative Programs
Elections
Voting System
DataCert (Benzaken)
SQL application
SQL
Database System
Spec of C program
Program Logic for C
Floyd proof automation
Concurrent Verifiable C
soundness proof
soundness proof
Concurrent Clight
concurrency correctness proof
CompCert optimizing compiler
Verified Software Toolchain (Appel)
CompCert (Leroy)
Clight to LLVM
Coq
Gallina (Coq)
compiler
CPS
CertiCoq (Appel, Morrisett)
Functional Programs
QuickChick (Pierce)
Quick-Chick
GHC front end
Core Haskell
(unverified) GHC compiler
Haskell (Weirich)
LLVM
LLVM compiler phase
LLVM existing unverified phase
LLVM
LLVM compiler phase
LLVM existing unverified phase
VeLLVM (Zdancewic)
x86 or ARM Instruction Set Architecture
x86 or ARM ISA
Bluespec
Bluespec compiler
Core Verilog
x86/ARM RTL
Kami (Chlipala)
Register Transfer Language
Machine models and memory models
REMS (Sewell)
Project (researcher)
Specifications
Implementations, translations, proofs
thin lines are
collaborations external to DeepSpec group



PRL Project "Proof/Program Refinement Logic"

implementing computational mathematics
and providing logic-based tools that help automate programming



Autoformalization

aristotle

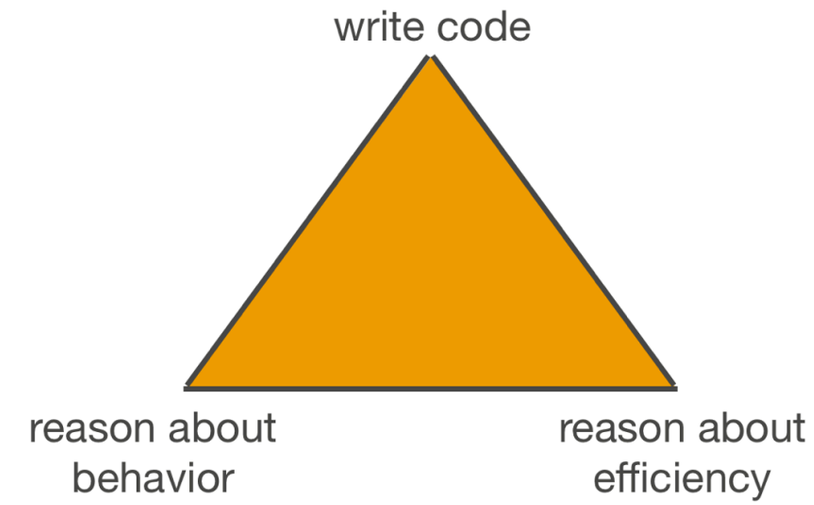
Axiom

Math, Inc.

Google DeepMind

21.3k followers https://www.deepmind.com/

Goals



- Learn to write programs and proofs in Agda
- Learn to integrate programming and specification and proving
- Understand algorithms that are hard otherwise
- Prepare to use these tools/skills in your job
- Prepare for research/grad school

Student Agda Projects

- Ben Hudson: techniques for reasoning about cost
- Joomy Korkut and Max Trifunovski: correctness of regular expression matching
- Emily Black: correctness of dictionary data structure deletion
- Phil Kaelbling: attempt at extending Agda for modal type theory
- Tess Lepeska-True: correctness of edit-distance algorithm
- Zameen Cater: version control in cubical type theory
- Adam Kleber: formalizing logic and combinatorics (Myhill Isomorphism Theorem)