Satisfiability Checking

01 Overview

Prof. Dr. Erika Ábrahám

RWTH Aachen University
Informatik 2
LuFG Theory of Hybrid Systems

WS 23/24

01 Overview

Organizational

- Language: English
- Lecture (V3+Ü1):
  Monday 08:30-10:00 (AH III) and
  Friday 08:30-10:00 (AH II)
- Exercise (Ü1):
  Tuesday 10:30-11:15 (AH VI)
- Room or schedule changes:
  communicated via Moodle
- Assistants:
  Jasper Nalbach
  Valentin Promies
- Contact:
  `teaching@ths.rwth-aachen.de`

Organizational

- Weekly exercise sheets. Not mandatory (no submission) but strongly recommended.
- 3 mandatory eTests in Moodle: at least 8 out of $3x5 = 15$ eTest points are needed for exam admission.
- Exam: written, 120 minutes, 120 exam points
- During lectures, you can earn up to 12 bonus exam points! We pose 36 questions during the lectures over the whole semester, which you can answer in Moodle. Each correct answer brings you 1/3 exam point.

  Submitting answers is possible only during the lecture!

Question board in Moodle

Learning materials

- Daniel Kroening and Ofer Strichman.
  Decision Procedures: An Algorithmic Point of View.
  Springer-Verlag, Berlin, 2008.
- Slides (grateful for parts from
  `www.decision-procedures.org/slides/`)
- Selected papers and other materials (especially recordings from a previous year's lecture) in Moodle

What is this lecture about?

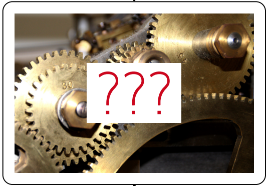
Quantifier-free
logical
formula

Solver

Satisfiability of the
input formula

What is this lecture about?

Quantifier-free logical formula



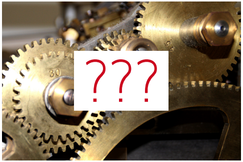
Solver

Satisfiability of the input formula

What is this lecture about?

???

Quantifier-free logical formula



Solver

???

Satisfiability of the input formula

The Boolean satisfiability problem... SAT

命题逻辑

Satisfiability problem for propositional logic

Given a formula combining some atomic propositions using the Boolean operators "and" ($\land$), "or" ($\lor$) and "not" ($\neg$), decide whether we can substitute truth values for the propositions such that the formula evaluates to true. 公式计算为真

Example

Formula:	$(a \lor \neg b) \land (\neg a \lor b \lor c)$
Satisfying assignment:	$a = true$, $b = false$, $c = true$

It is the perhaps most well-known NP-complete problem [Cook, 1971] [Levin, 1973].

...and its extension to theories SMT

SMT就是SAT（布尔表达式可满足性理论）在布尔表达式的基础上拓展到其他理论，比如加上实数理论

Satisfiability modulo theories problem (informal)

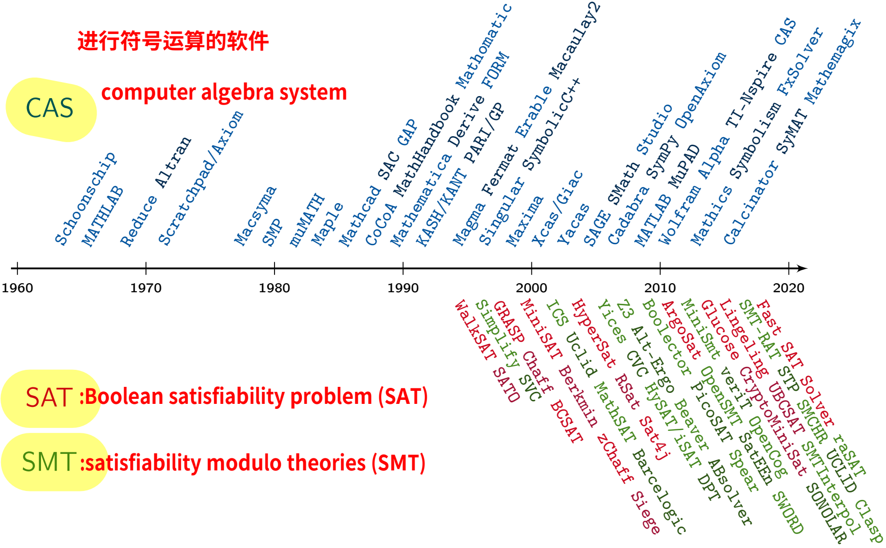
Given a Boolean combination of $\boxed{\text{constraints from some theories}}$, decide whether we can substitute (type-correct) values for the (theory) variables such that the formula evaluates to true.

A non-linear real arithmetic example

Formula: $(x - 2y > 0 \vee x^2 - 2 = 0) \wedge x^4 y + 2x^2 - 4 > 0$

Satisfying assignment: $x = \sqrt{2}, \quad y = 2$

Hard problems... non-linear integer arithmetic is even undecidable.


进行符号运算的软件
CAS
computer algebra system
Schoonschip
MATHLAB
Reduce
Altran
Scratchpad/Axiom
Macsyma
SMP
muMATH
Maple
Mathcad
SAC
GAP
CoCoA
MathHandbook
Mathematic
Mathematica
Derive
FORM
KASH/KANT
PARI/GP
Magma
Fermat
Erable
Macaulay2
Singular
SymbolicC++
Maxima
Xcas/Giac
Yacas
SAGE
SMath Studio
Cadabra
SymPy
OpenAxiom
MATLAB
MuPAD
Wolfram Alpha
TI-Nspire CAS
Mathics
Symbolism
FxSolver
Calcinator
SyMAT
Mathemagix
1960
1970
1980
1990
2000
2010
2020
WalkSAT
SATO
Simplify
SVC
GRASP
Chaff
BCSAT
MiniSAT
Berkmin
zChaff
Siege
ICS
Uclid
MathSAT
Barcelogic
HyperSat
RSat
Sat4j
Yices
CVC
HySAT/iSAT
DPT
Z3
Alt-Ergo
Beaver
ABsolver
Boolector
PicoSAT
Spear
ArgoSat
OpenSMT
SatEEn
MiniSmt
veriT
OpenCog
SWORD
Glucose
CryptoMiniSat
SONOLAR
Lingeling
UBCSAT
SMTInterpol
SMT-RAT
STP
SMCHR
UCLID
Fast SAT Solver
raSAT
Clasp
SAT :Boolean satisfiability problem (SAT)
SMT :satisfiability modulo theories (SMT)

Satisfiability checking for propositional logic

Success story: SAT-solving

- Practical problems with millions of variables are solvable.
- Frequently used in different research areas for, e.g., analysis, synthesis and optimisation.
- Also massively used in industry for, e.g., digital circuit design and verification.

Satisfiability checking for propositional logic

Success story: SAT-solving

- Practical problems with millions of variables are solvable.
- Frequently used in different research areas for, e.g., analysis, synthesis and optimisation.
- Also massively used in industry for, e.g., digital circuit design and verification.

Community support:

- Standardised input language, lots of benchmarks available.
- Competitions since 2002.

  2021: 4 tracks, 45 versions of 18 solvers in main track

  SAT Live! forum as community platform, dedicated conferences, journals, etc.

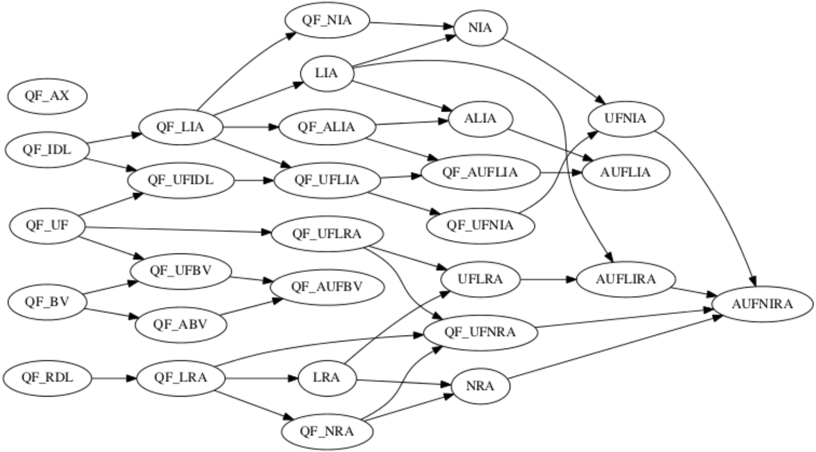
Satisfiability modulo theories solving

- Propositional logic is sometimes too weak for modelling.
- We need more expressive logics and decision procedures for them.
- Logics:
  quantifier-free fragments of first-order logic over various theories.
- Our focus: SAT-modulo-theories (SMT) solving.

Satisfiability modulo theories solving

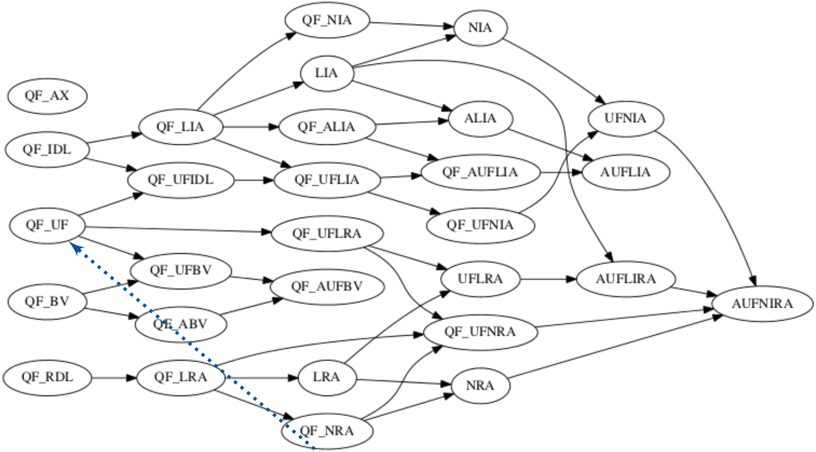
- Propositional logic is sometimes too weak for modelling.
- We need more expressive logics and decision procedures for them.
- Logics:
  quantifier-free fragments of first-order logic over various theories.
- Our focus: SAT-modulo-theories (SMT) solving.
- SMT-LIB as standard input language since 2004.
- Competitions since 2005.
- 2021 SMT-COMP competition: 25 solvers

SMT-LIB theories



Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

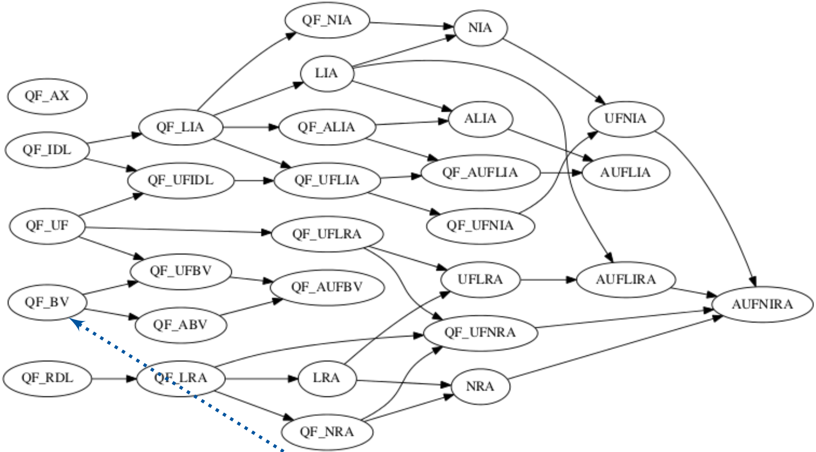


Quantifier-free equality logic with uninterpreted functions

$$(a = c \wedge b = d) \rightarrow f(a, b) = f(c, d)$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

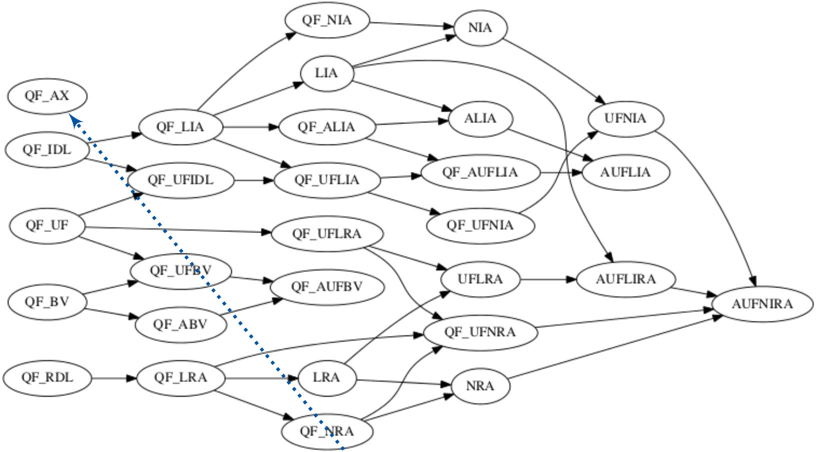


Quantifier-free bit-vector arithmetic

$$a + b \geq 0 \wedge (a|b) \leq (a\&b)$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

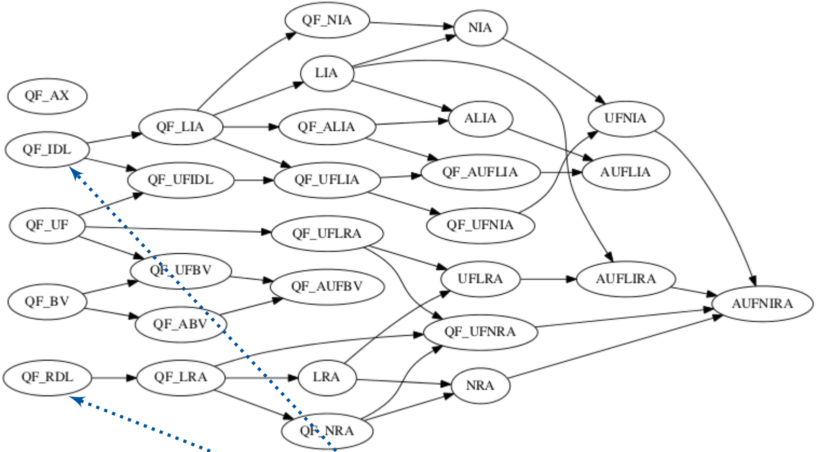


Quantifier-free array theory

$$i = j \rightarrow read(write(a, i, v), j) = v$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

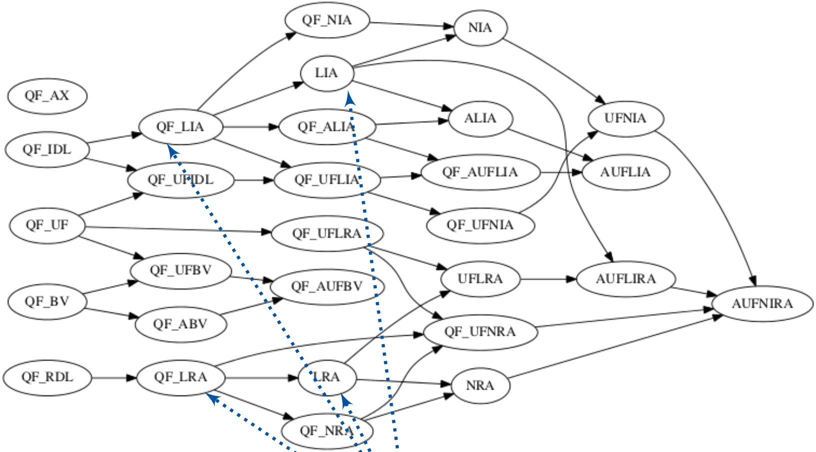


Quantifier-free integer/rational difference logic

$$x - y \geq 0 \vee x - z < 0$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

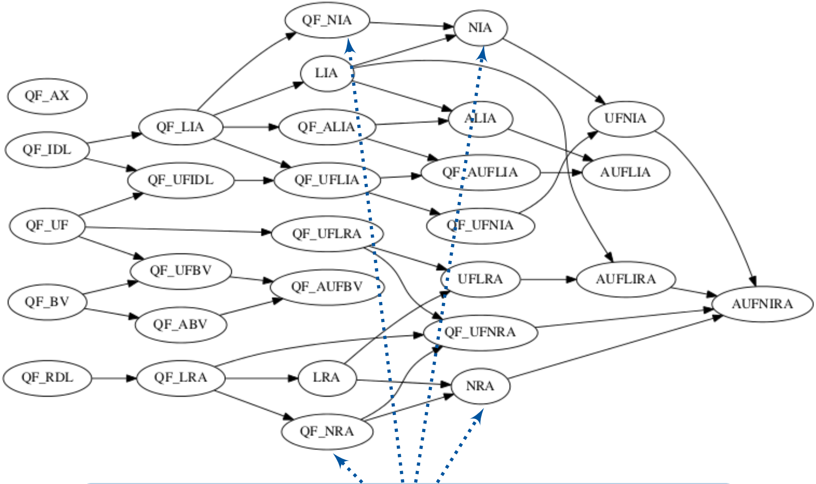


(Quantifier-free) real/integer linear arithmetic

$4x + 7y = 8 \wedge (y = 0 \vee x > y)$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

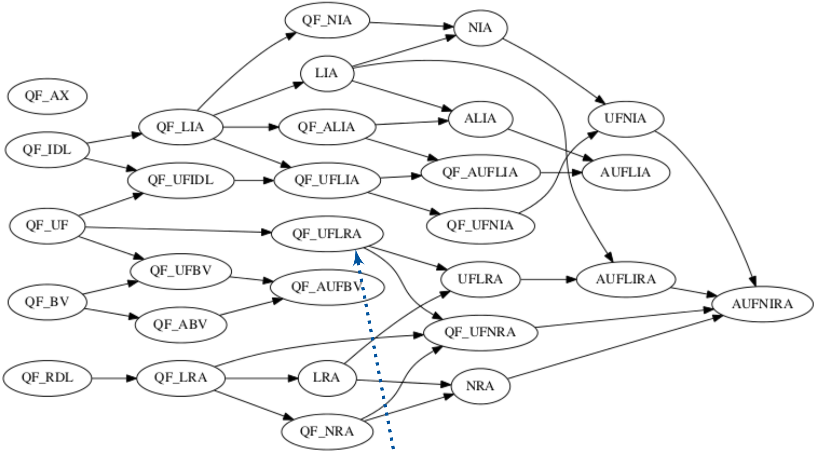


(Quantifier-free) real/integer non-linear arithmetic

$$x^2 + 2xy + y^2 > 0 \vee (x \geq 1 \wedge xz + yz^2 = 0)$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

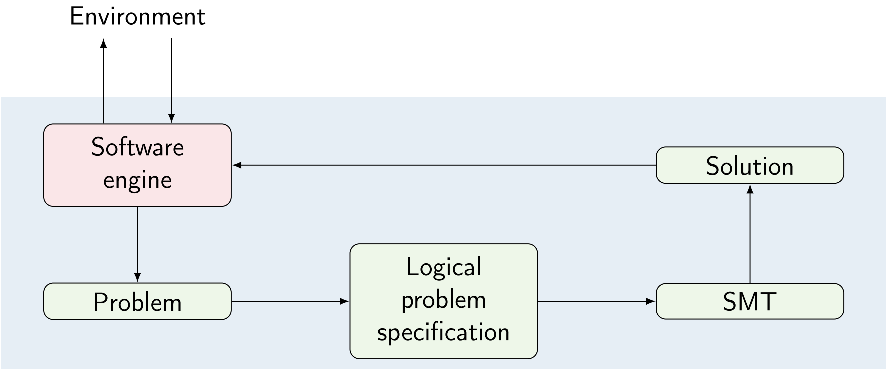


Combined theories

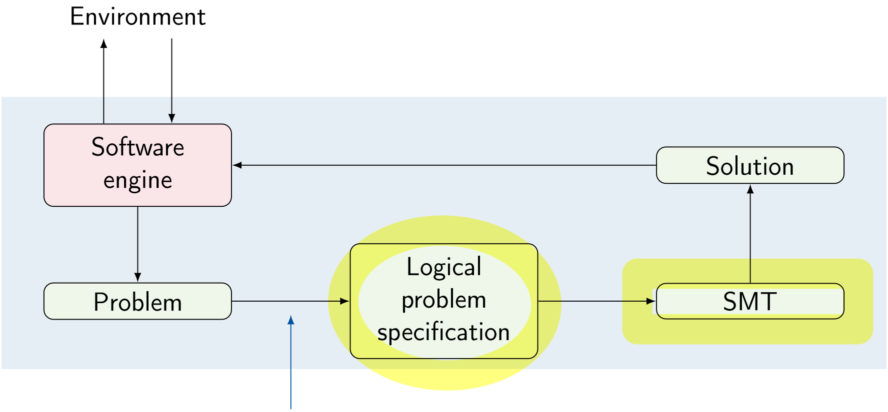
$2f(x) + 5y > 0 \wedge \neg(f(x) = y \vee x + 2y = 0)$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SAT/SMT embedding structure

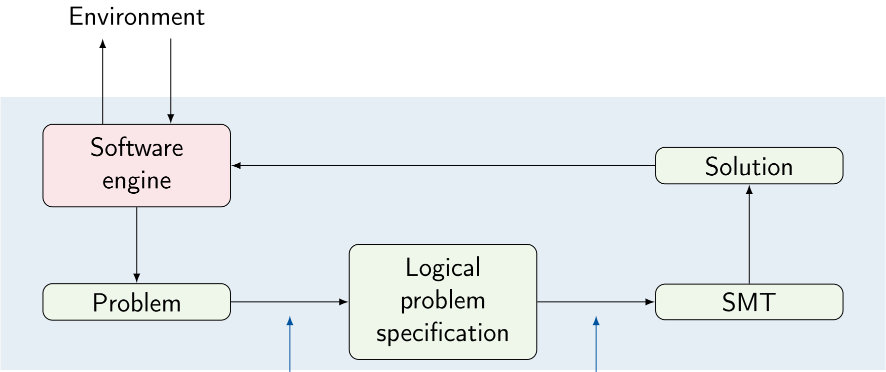


SAT/SMT embedding structure



Encoding: SAT/SMT-LIB standard
elaborate encoding is extremely important!

SAT/SMT embedding structure



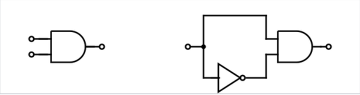
Encoding: SAT/SMT-LIB standard
elaborate encoding is extremely important!

standard input language → free solver choice

Circuit satisfiability problem

From Wikipedia, the free encyclopedia

In theoretical computer science, the **circuit satisfiability problem** (also known as **CIRCUIT-SAT**, **CircuitSAT**, **CSAT**, etc.) is the decision problem of determining whether a given Boolean circuit has an assignment of its inputs that makes the output true.[1] In other words, it asks whether the inputs to a given Boolean circuit can be consistently set to **1** or **0** such that the circuit outputs **1**. If that is the case, the circuit is called *satisfiable*. Otherwise, the circuit is called *unsatisfiable.* In the figure to the right, the left circuit can be satisfied by setting both inputs to be **1**, but the right circuit is unsatisfiable.



The circuit on the left is satisfiable but the circuit on the right is not.

CircuitSAT is closely related to Boolean satisfiability problem (SAT), and likewise, has been proven to be NP-complete.[2] It is a prototypical NP-complete problem; the Cook–Levin theorem is sometimes proved on CircuitSAT instead of on the SAT and then reduced to the other satisfiability problems to prove their NP-completeness.[1][3] The satisfiability of a circuit containing m arbitrary binary gates can be decided in time $O(2^{0.4058m})$.[4]

Source: Wikipedia.

Application example: Symbolic execution

Program 1.2.1 A recursion-free program with bounded loops and an SSA unfolding.

```
int Main(int x, int y)
{
  if (x < y)
     x = x + y;
  for (int i = 0; i < 3; ++i) {
      y = x + Next(y);
  }
  return x + y;
}

int Next(int x) {
  return x + 1;
}
```

```
int Main(int x0, int y0)
{
  int x1;
  if (x0 < y0)
     x1 = x0 + y0;
  else
     x1 = x0;
  int y1 = x1 + y0 + 1;
  int y2 = x1 + y1 + 1;
  int y3 = x1 + y2 + 1;
  return x1 + y3;
}
```

$$\exists x_1, y_1, y_2, y_3 \left(\begin{array}{l} (x_0 < y_0 \implies x_1 = x_0 + y_0) \ \wedge \ (\neg(x_0 < y_0) \implies x_1 = x_0) \wedge \\ y_1 = x_1 + y_0 + 1 \ \wedge \ y_2 = x_1 + y_1 + 1 \ \wedge \ y_3 = x_1 + y_2 + 1 \ \wedge \\ result = x_1 + y_3 \end{array} \right)$$

Source: Nikolaj Bjørner and Leonardo de Moura. *Applications of SMT solvers to Program Verification*.

Rough notes for SSFT 2014.

Application example: Bounded model checking

Carnegie Mellon

CBMC Homepage

Bounded Model Checking for Software

About CBMC

CBMC is a Bounded Model Checker for C and C++ programs. It supports C89, C99, most of C11 and most compiler extensions provided by gcc and Visual Studio. It also supports SystemC using Scoot. We have recently added experimental support for Java Bytecode.

CBMC verifies array bounds (buffer overflows), pointer safety, exceptions and user-specified assertions. Furthermore, it can check C and C++ for consistency with other languages, such as Verilog. The verification is performed by unwinding the loops in the program and passing the resulting equation to a decision procedure.

While CBMC is aimed for embedded software, it also supports dynamic memory allocation using `malloc` and `new`. For questions about CBMC, contact Daniel Kroening.

CBMC is available for most flavours of Linux (pre-packaged on Debian, Ubuntu and Fedora), Solaris 11, Windows and MacOS X. You should also read the CBMC license.

CBMC comes with a built-in solver for bit-vector formulas that is based on MiniSat. As an alternative, CBMC has featured support for external SMT solvers since version 3.3. The solvers we recommend are (in no particular order) Boolector, MathSAT, Yices 2 and Z3. Note that these solvers need to be installed separately and have different licensing conditions.

Source: D. Kroening. **CBMC home page.** `http://www.cprover.org/cbmc/`

Application example: Bounded model checking

Carnegie Mellon

CBMC Homepage

Bounded Model Checking for Software

Logical encoding of finite unsafe paths

About CBMC

CBMC is a Bounded Model Checker for C and C++ programs. It supports C89, C99, most of C11 and most compiler extensions provided by gcc and Visual Studio. It also supports SystemC using Scoot. We have recently added experimental support for Java Bytecode.

CBMC verifies array bounds (buffer overflows), pointer safety, exceptions and user-specified assertions. Furthermore, it can check C and C++ for consistency with other languages, such as Verilog. The verification is performed by unwinding the loops in the program and passing the resulting equation to a decision procedure.

While CBMC is aimed for embedded software, it also supports dynamic memory allocation using `malloc` and `new`. For questions about CBMC, contact Daniel Kroening.

CBMC is available for most flavours of Linux (pre-packaged on Debian, Ubuntu and Fedora), Solaris 11, Windows and MacOS X. You should also read the CBMC license.

CBMC comes with a built-in solver for bit-vector formulas that is based on MiniSat. As an alternative, CBMC has featured support for external SMT solvers since version 3.3. The solvers we recommend are (in no particular order) Boolector, MathSAT, Yices 2 and Z3. Note that these solvers need to be installed separately and have different licensing conditions.

Source: D. Kroening. **CBMC home page.** `http://www.cprover.org/cbmc/`

Application example: Bounded model checking

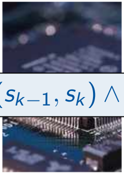
Carnegie Mellon

CBMC Homepage

Bounded Model Checking for Software

Logical encoding of finite unsafe paths

About CBMC



CBMC is a Bounded Model Checker for C and C++ programs. It supports C89, C99, most of C11 and most compiler extensions provided by gcc and Visual Studio. It also supports SystemC using Scoot. We have recently added experimental support for Java

Encoding idea: $Init(s_0) \wedge Trans(s_0, s_1) \wedge \ldots \wedge Trans(s_{k-1}, s_k) \wedge Bad(s_0, \ldots, s_k)$

tions and user-specified assertions. Furthermore, it can check C and C++ for consistency with other languages, such as Verilog. The verification is performed by unwinding the loops in the program and passing the resulting equation to a decision procedure.

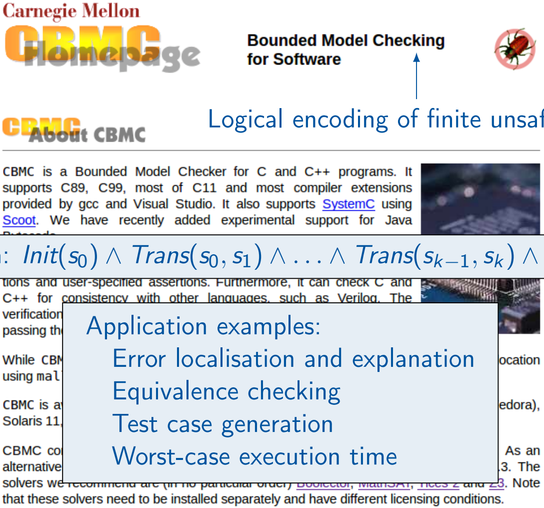
While CBMC is aimed for embedded software, it also supports dynamic memory allocation using `malloc` and `new`. For questions about CBMC, contact Daniel Kroening.

CBMC is available for most flavours of Linux (pre-packaged on Debian, Ubuntu and Fedora), Solaris 11, Windows and MacOS X. You should also read the CBMC license.

CBMC comes with a built-in solver for bit-vector formulas that is based on MiniSat. As an alternative, CBMC has featured support for external SMT solvers since version 3.3. The solvers we recommend are (in no particular order) Boolector, MathSAT, Yices 2 and Z3. Note that these solvers need to be installed separately and have different licensing conditions.

Source: D. Kroening. **CBMC home page.** `http://www.cprover.org/cbmc/`

Application example: Bounded model checking



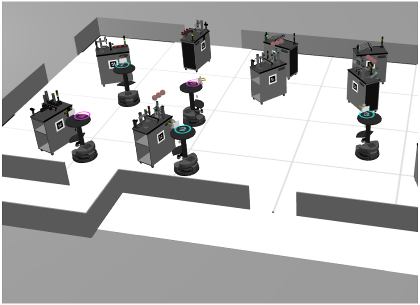
Logical encoding of finite unsafe paths

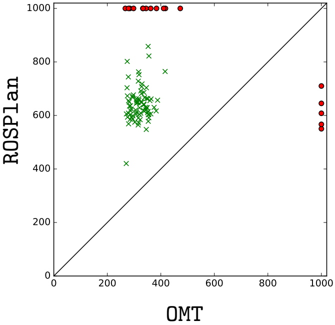
Encoding idea: $\mathit{Init}(s_0) \wedge \mathit{Trans}(s_0, s_1) \wedge \ldots \wedge \mathit{Trans}(s_{k-1}, s_k) \wedge \mathit{Bad}(s_0, \ldots, s_k)$

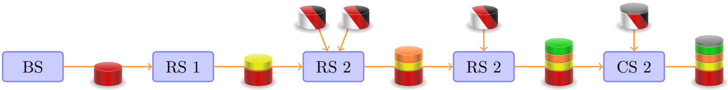
Application examples:
- Error localisation and explanation
- Equivalence checking
- Test case generation
- Worst-case execution time

Source: D. Kroening. **CBMC home page.** `http://www.cprover.org/cbmc/`

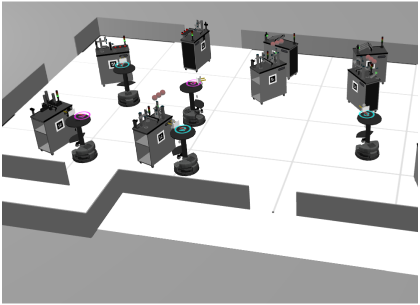
Planning

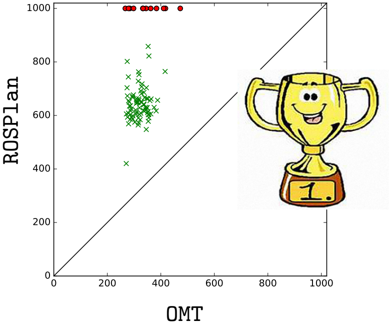


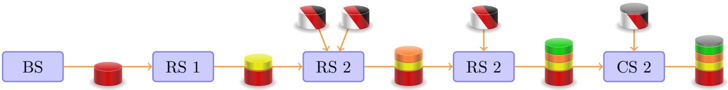




Source: F. Leofante, E. Giunchiglia, E. Ábrahám, A. Tacchella. **Optimal Planning Modulo Theories.** Proc. of IJCAI'20.







Source: F. Leofante, E. Giunchiglia, E. Ábrahám, A. Tacchella. **Optimal Planning Modulo Theories.** Proc. of IJCAI'20.

Application example: Security at Amazon

https://www.amazon.science/blog/a-billion-smt-queries-a-day

:arted 7 Simple Ways to Free … Career Paths in Comp…

amazon | science

Research areas Blog News and features Publications Conferences Collaborations Careers

AUTOMATED REASONING

A billion SMT queries a day

CAV keynote lecture by the director of applied science for AWS Identity explains how AWS is making the power of automated reasoning available to all customers.

By Neha Rungta

August 18, 2022

Share

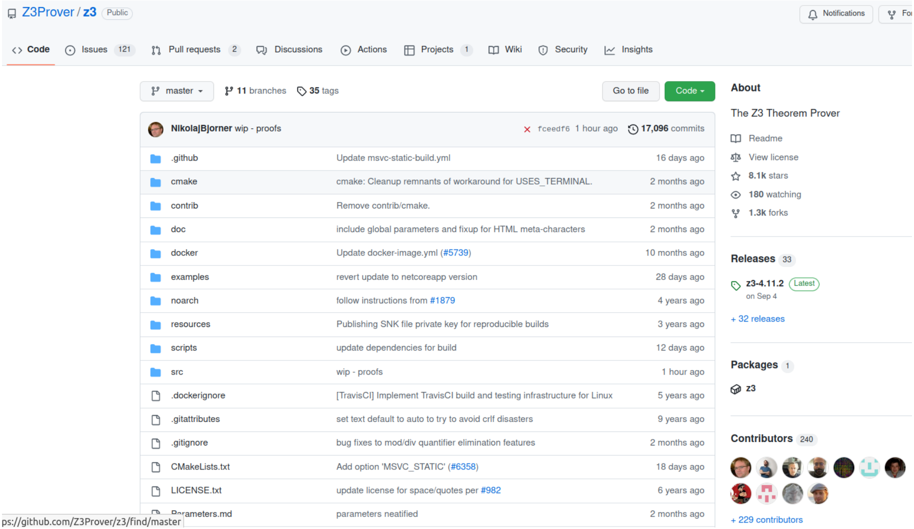
Conference

FLoC 2022

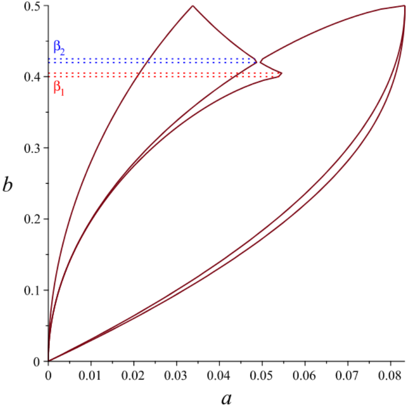
At this year's Computer-Aided Verification (CAV) conference — a leading automated-reasoning conference collocated with the Federated Logic Conferences (FLoC) — Amazon's Neha Rungta delivered a keynote talk in which she suggested that innovations at Amazon have "ushered in the golden age of automated reasoning".

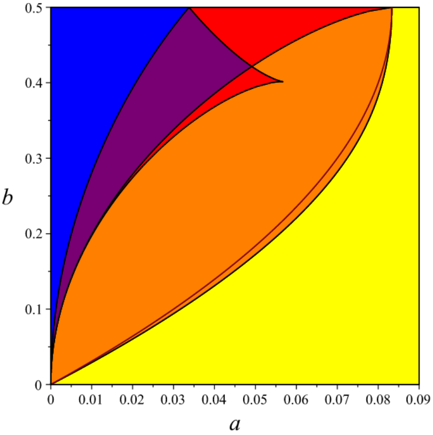
Amazon scientists and engineers are

Application example: Z3 from Microsoft



Application example: Biology





Numeric computation

Symbolic computation

Source: Röst, Gergely, and AmirHosein Sadeghimanesh. 'Exotic Bifurcations in Three Connected Populations with Allee Effect'. International Journal of Bifurcation and Chaos 31, no. 13 (October 2021): 2150202. https://doi.org/10.1142/S0218127421502023.

Learning target

- What is the satisfiability checking problem?
- How can SAT/SMT solvers be used in applications?