Logic and Mechanized Reasoning

Satisfiability Modulo Theories

Marijn J.H. Heule

Carnegie Mellon University

Based on tutorials by Barrett, Griggio, Jovanović, de Moura, Oliveras, and Tinelli

SMT Overview

SMT Solving

SMT Theories

SMT Overview

SMT Solving

SMT Theories

Overview: Satisfiability Modulo Theories

Problem

Check if a quantifier-free first-order formula can be satisfied with respect to some background theories.

Example (QF_UFLRA)

$$(z = 1 \vee z = 0) \wedge (x - y + z = 1) \wedge (f(x) > f(y))$$

Overview: Satisfiability Modulo Theories

Problem
Check if a quantifier-free first-order formula can be satisfied with respect to some background theories.

Example (QF_UFLRA)

$$(z = 1 \vee z = 0) \wedge (x - y + z = 1) \wedge (f(x) > f(y))$$

1. Linear real arithmetic (LRA).

Overview: Satisfiability Modulo Theories

Problem

Check if a quantifier-free first-order formula can be satisfied with respect to some background theories.

Example (QF_UFLRA)

$$(z = 1 \vee z = 0) \wedge (x - y + z = 1) \wedge (f(x) > f(y))$$

1. Linear real arithmetic (LRA).
2. Uninterpreted functions (UF).

Overview: Satisfiability Modulo Theories

Problem

Check if a quantifier-free first-order formula can be satisfied with respect to some background theories.

Example (QF_UFLRA)

$$(z = 1 \vee z = 0) \wedge (x - y + z = 1) \wedge (f(x) > f(y))$$

1. Linear real arithmetic (LRA).
2. Uninterpreted functions (UF).
3. Satisfiable with $z \mapsto 0, x \mapsto 1, y \mapsto 0, f(1) \mapsto 1, f(0) \mapsto 0$

Overview: Many SMT Applications I

Schedule n jobs, each composed of m consecutive tasks, on m machines using at most t timeslots.

Schedule in 8 time slots.

$d_{i,j}$	Machine 1	Machine 2
Job 1	2	1
Job 2	3	1
Job 3	2	3

Overview: Many SMT Applications I

Schedule n jobs, each composed of m consecutive tasks, on m machines using at most t timeslots.

Schedule in 8 time slots.

$d_{i,j}$	Machine 1	Machine 2
Job 1	2	1
Job 2	3	1
Job 3	2	3

$$
\begin{array}{l}
(t_{1,1} \geq 0) \wedge (t_{1,2} \geq t_{1,1} + 2) \wedge (t_{1,2} + 1 \leq 8) \\
(t_{2,1} \geq 0) \wedge (t_{2,2} \geq t_{2,1} + 3) \wedge (t_{2,2} + 1 \leq 8) \\
(t_{3,1} \geq 0) \wedge (t_{3,2} \geq t_{3,1} + 2) \wedge (t_{3,2} + 3 \leq 8) \\
((t_{1,1} \geq t_{2,1} + 3) \vee (t_{2,1} \geq t_{1,1} + 2)) \\
((t_{1,1} \geq t_{3,1} + 2) \vee (t_{3,1} \geq t_{1,1} + 2)) \\
((t_{2,1} \geq t_{3,1} + 2) \vee (t_{3,1} \geq t_{2,1} + 3)) \\
((t_{1,2} \geq t_{2,2} + 1) \vee (t_{2,2} \geq t_{1,2} + 1)) \\
((t_{1,2} \geq t_{3,2} + 3) \vee (t_{3,2} \geq t_{1,2} + 1)) \\
((t_{2,2} \geq t_{3,2} + 3) \vee (t_{3,2} \geq t_{2,2} + 1))
\end{array}
$$

Example from [DMB11]

Overview: Many SMT Applications I

Schedule n jobs, each composed of m consecutive tasks, on m machines using at most t timeslots.

Schedule in 8 time slots.

$d_{i,j}$	Machine 1	Machine 2
Job 1	2	1
Job 2	3	1
Job 3	2	3

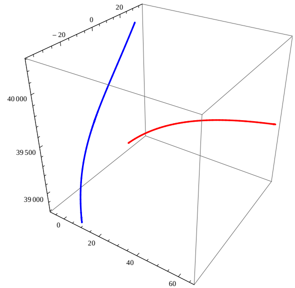
$$
\begin{array}{l}
(t_{1,1} \geq 0) \wedge (t_{1,2} \geq t_{1,1} + 2) \wedge (t_{1,2} + 1 \leq 8) \\
(t_{2,1} \geq 0) \wedge (t_{2,2} \geq t_{2,1} + 3) \wedge (t_{2,2} + 1 \leq 8) \\
(t_{3,1} \geq 0) \wedge (t_{3,2} \geq t_{3,1} + 2) \wedge (t_{3,2} + 3 \leq 8) \\
((t_{1,1} \geq t_{2,1} + 3) \vee (t_{2,1} \geq t_{1,1} + 2)) \\
((t_{1,1} \geq t_{3,1} + 2) \vee (t_{3,1} \geq t_{1,1} + 2)) \\
((t_{2,1} \geq t_{3,1} + 2) \vee (t_{3,1} \geq t_{2,1} + 3)) \\
((t_{1,2} \geq t_{2,2} + 1) \vee (t_{2,2} \geq t_{1,2} + 1)) \\
((t_{1,2} \geq t_{3,2} + 3) \vee (t_{3,2} \geq t_{1,2} + 1)) \\
((t_{2,2} \geq t_{3,2} + 3) \vee (t_{3,2} \geq t_{2,2} + 1))
\end{array}
$$

Example from [DMB11]

Run SMT solver (QF_IDL)

$$t_{1,1} \mapsto 5,\ \ t_{1,2} \mapsto 7,\ \ t_{2,1} \mapsto 2,\ \ t_{2,2} \mapsto 6,\ \ t_{3,1} \mapsto 0,\ \ t_{3,2} \mapsto 3$$

Overview: Many SMT Applications II



$$T_1^x(t) = -3.2484 + 270.7t + 433.12t^2 - 324.83999t^3$$
$$T_1^y(t) = 15.1592 + 108.28t + 121.2736t^2 - 649.67999t^3$$
$$T_1^z(t) = 38980.8 + 5414t - 21656t^2 + 32484t^3$$

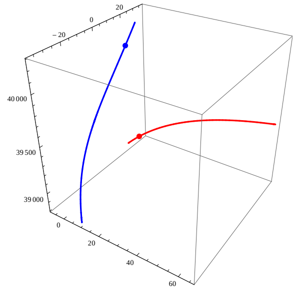
$$T_2^x(t) = 1.0828 - 135.35t + 234.9676t^2 2 + 3248.4t^3$$
$$T_2^y(t) = 18.40759 - 230.6364t - 121.2736t^2 - 649.67999t^3$$
$$T_2^z(t) = 40280.15999 - 10828t + 24061.9816t^2 - 32484t^3$$

$D = 5$ $\qquad H = 1000$ $\qquad 0 \leq t \leq \frac{1}{20}$

$|T_1^z(t) - T_2^z(t)| \leq H \qquad (T_1^x(t) - T_2^x(t))^2 + (T_1^y(t) - T_2^y(t))^2 \leq D^2$

Example from [NM12]

Overview: Many SMT Applications II



$$T_1^x(t) = -3.2484 + 270.7t + 433.12t^2 - 324.83999t^3$$
$$T_1^y(t) = 15.1592 + 108.28t + 121.2736t^2 - 649.67999t^3$$
$$T_1^z(t) = 38980.8 + 5414t - 21656t^2 + 32484t^3$$

$$T_2^x(t) = 1.0828 - 135.35t + 234.9676t^2 2 + 3248.4t^3$$
$$T_2^y(t) = 18.40759 - 230.6364t - 121.2736t^2 - 649.67999t^3$$
$$T_2^z(t) = 40280.15999 - 10828t + 24061.9816t^2 - 32484t^3$$

$$D = 5 \qquad H = 1000 \qquad 0 \le t \le \frac{1}{20}$$

$$|T_1^z(t) - T_2^z(t)| \le H \qquad (T_1^x(t) - T_2^x(t))^2 + (T_1^y(t) - T_2^y(t))^2 \le D^2$$

Example from [NM12]

Run SMT solver (QF_NRA)

$$t \mapsto \frac{319}{16384} \approx 0.019470215$$

Overview: Modeling and Solving

Modeling

- Depending on the problem domain, select a fitting theory.
- Consider expressivity vs solving complexity.

Solving

- Get an SMT solver that supports the theory.
- Hope for the best.

Overview: Uninterpreted Functions

Uninterpreted Functions (QF_UF)

Simplest first-order theory, with equality, applications of uninterpreted functions, and variables of uninterpreted types.

Reflexivity: $x = x$

Symmetry: $x = y \Rightarrow y = x$

Transitivity: $x = y \wedge y = z \Rightarrow x = z$

Congruence: $x = y \Rightarrow f(x) = f(y)$

Example

$$f(f(f(x))) = x \ \wedge \ f(f(f(f(f(x))))) = x \ \wedge \ f(x) \neq x$$

Overview: Arrays

Theory of Arrays [McC93]

Operates over types array, index, element and function symbols

$$\begin{aligned} \_[\_] &: \text{array} \times \text{index} \mapsto \text{element} \\ \text{store} &: \text{array} \times \text{index} \times \text{element} \mapsto \text{array} \quad . \end{aligned}$$

Read-Over-Write-1: $\text{store}(a, i, e)[i] = e$

Read-Over-Write-2: $i \neq j \Rightarrow \text{store}(a, i, e)[j] = a[j]$

Extensionality: $a \neq b \Rightarrow \exists i : a[i] \neq b[i]$

Example

$$\text{store}(\text{store}(a, i, a[j]), j, a[i]) = \text{store}(\text{store}(a, j, a[i]), i, a[j])$$

Overview: Arithmetic

Arithmetic

Arithmetic constraints (inequalities, equalities) over arithmetic (real or integer) variables.

- Difference logic (QF_RDL, QF_IDL):

$$x - y \leq 1 \ , \qquad\qquad x - y > 10 \ .$$

- Linear arithmetic (QF_LRA, QF_LIA):

$$2x - 3y + 4z \leq 5 \ .$$

- Non-linear arithmetic (QF_NRA, QF_NIA):

$$x^2 + 3xy + y^2 > 0 \ .$$

Overview: Bitvectors

Bitvectors (QF_BV)

Operates over fixed-size bit-vectors, with bit-vector operations:

- concatenation $a \circ b$, extraction $a[i:j]$
- bit-wise operators $\sim a$, $a|b$, $a \& b$, ...
- shifts $a \ll k$, $b \gg k$ (logical, arithmetic)
- arithmetic $a + b$, $a - b$, $a * b$, a/b, ...
- predicates $=$, $<$, $\leq$, ... (signed and unsigned)

Semantics similar to programming languages.

Example (a is 32-bits)

$$(\sim a \,\&\, (a+1)) >_u a$$

Overview: Other Interesting Theories

Some other theories

- Floating point [BDG+14, ZWR14]
- Inductive data-types [BST07]
- Strings and regular expressions [LRT+14, KGG+09]
- Quantifiers [DMB07, RTG+13]
- Differential Equations [GKC13]
- ...

SMT Overview

SMT Solving

SMT Theories

Solving: Introduction

Check T-satisfiability of a T-formula Γ

Solving: Introduction

Check T-satisfiability of a T-formula Γ

1. Convert to DNF

$$\Gamma \Leftrightarrow \bigvee_{i=1}^{D} (L_1^i \wedge L_2^i \wedge \cdots \wedge L_{n_i}^i) \ .$$

Solving: Introduction

Check T-satisfiability of a T-formula Γ

1. Convert to DNF

$$\Gamma \Leftrightarrow \bigvee_{i=1}^{D} (L_1^i \wedge L_2^i \wedge \cdots \wedge L_{n_i}^i) \ .$$

2. If any of disjuncts is T-satisfiable, return SAT, else UNSAT.

Solving: Introduction

Check T-satisfiability of a T-formula Γ

1. Convert to DNF

$$\Gamma \Leftrightarrow \bigvee_{i=1}^{D} (L_1^i \wedge L_2^i \wedge \cdots \wedge L_{n_i}^i) \ .$$

2. If any of disjuncts is T-satisfiable, return SAT, else UNSAT.

Theory solver/Decision procedure for T

Procedure to decide satisfiability of a conjunction of T-literals.

Solving: Apply a SAT Solver

Use a SAT solver

- Instead of DNF: Apply a SAT solver.
- Check the literals selected by the SAT solver.
- If not T-satisfiable, add a blocking clause.

Solving: Very Lazy SMT Example

View

Theory

$$\neg a = b$$
$$(x = a \vee x = b)$$
$$(y = a \vee y = b)$$
$$(z = a \vee z = b)$$
$$\neg x = y$$

Solving: Very Lazy SMT Example

View
Boolean

$$
\begin{array}{l}
\neg p_1 \\
(\; p_2 \;\lor\; p_3 \;) \\
(\; p_4 \;\lor\; p_5 \;) \\
(\; p_6 \;\lor\; p_7 \;) \\
\neg p_8
\end{array}
$$

Solving: Very Lazy SMT Example

View
Boolean

$$
\begin{array}{l}
\neg p_1 \\
(\; p_2 \; \vee \; p_3 \;) \\
(\; p_4 \; \vee \; p_5 \;) \\
(\; p_6 \; \vee \; p_7 \;) \\
\neg p_8
\end{array}
$$

Check with SAT solver

Solving: Very Lazy SMT Example

View

Boolean

$$\neg p_1$$
$$(\ p_2 \lor p_3\ )$$
$$(\ p_4 \lor p_5\ )$$
$$(\ p_6 \lor p_7\ )$$
$$\neg p_8$$

Check with SAT solver

$$[\![\ \neg p_1\ ,\ \neg p_8\ ,\ \neg p_3\ ,\ p_2\ ,\ \neg p_5\ ,\ p_4\ ,\ \neg p_7\ ,\ p_6\ ]\!]$$

Solving: Very Lazy SMT Example

View

Theory

$$\neg a = b$$
$$(x = a \vee x = b)$$
$$(y = a \vee y = b)$$
$$(z = a \vee z = b)$$
$$\neg x = y$$

Check with SAT solver

$$[\![\neg a = b, \neg x = y, \neg x = b, x = a, \neg y = b, y = a, \neg z = b, z = a]\!]$$

Solving: Very Lazy SMT Example

$$\neg a = b$$
$$(x = a \vee x = b)$$
$$(y = a \vee y = b)$$
$$(z = a \vee z = b)$$
$$\neg x = y$$

View

Theory

Check with SAT solver

$$[\![\neg a = b, \neg x = y, \neg x = b, x = a, \neg y = b, y = a, \neg z = b, z = a]\!]$$

Check with T-solver

$$x = a \wedge y = a \Rightarrow x = y$$

Solving: Very Lazy SMT Example

$$\neg a = b$$
$$(x = a \vee x = b)$$
$$(y = a \vee y = b)$$
$$(z = a \vee z = b)$$
$$\neg x = y$$
$$(x = y \vee \neg x = a \vee \neg y = a)$$

View

Theory

Check with SAT solver

$$[\![\neg a = b, \neg x = y, \neg x = b, x = a, \neg y = b, y = a, \neg z = b, z = a]\!]$$

Check with T-solver

$$x = a \wedge y = a \Rightarrow x = y$$

Add blocking clause: $x = y \vee \neg x = a \vee \neg y = a$

Solving: Very Lazy SMT Example

View
Boolean

$$\begin{array}{l} \neg p_1 \\ (\ p_2 \ \vee \ p_3 \ ) \\ (\ p_4 \ \vee \ p_5 \ ) \\ (\ p_6 \ \vee \ p_7 \ ) \\ \neg p_8 \\ (\ p_8 \ \vee \ \neg p_2 \ \vee \ \neg p_4 \ ) \end{array}$$

Check with SAT solver

Solving: Very Lazy SMT Example

View
Boolean

$$\neg p_1$$
$$(\ p_2 \ \lor\ p_3\ )$$
$$(\ p_4 \ \lor\ p_5\ )$$
$$(\ p_6 \ \lor\ p_7\ )$$
$$\neg p_8$$
$$(\ p_8 \ \lor\ \neg p_2 \ \lor\ \neg p_4\ )$$

Check with SAT solver

$$[\![\ \neg p_1\ ,\ \neg p_8\ ,\ \neg p_3\ ,\ p_2\ ,\ \neg p_4\ ,\ p_5\ ,\ \neg p_7\ ,\ p_6\ ]\!]$$

Solving: Very Lazy SMT Example

View

Theory

$$
\begin{aligned}
&\neg a = b \\
&(x = a \vee x = b) \\
&(y = a \vee y = b) \\
&(z = a \vee z = b) \\
&\neg x = y \\
&(x = y \vee \neg x = a \vee \neg y = a)
\end{aligned}
$$

Check with SAT solver

$$[\![\neg a = b, \neg x = y, \neg x = b, x = a, \neg y = a, y = b, \neg z = b, z = a]\!]$$

Solving: Very Lazy SMT Example

$$\neg a = b$$
$$(x = a \vee x = b)$$
$$(y = a \vee y = b)$$
$$(z = a \vee z = b)$$
$$\neg x = y$$
$$(x = y \vee \neg x = a \vee \neg y = a)$$

View
Theory

Check with SAT solver

$$[\![\neg a = b, \neg x = y, \neg x = b, x = a, \neg y = a, y = b, \neg z = b, z = a]\!]$$

Check with T-solver

Satisfiable: $a, x, z \mapsto c_1, \quad b, y \mapsto c_2$

Solving: Very Lazy SMT

Properties

- SAT and T-solver only communicate via existing literals.
- Number of possible conflicts finite $\Rightarrow$ termination.
- Reuse the improvements in SAT solving.
- SAT solver is "blind" and can get lost :(.

Solving: Very Lazy SMT

Properties

- SAT and T-solver only communicate via existing literals.
- Number of possible conflicts finite $\Rightarrow$ termination.
- Reuse the improvements in SAT solving.
- SAT solver is "blind" and can get lost :(.

Integrate closely with SAT solver: DPLL(T) [DMR02, NOT05]

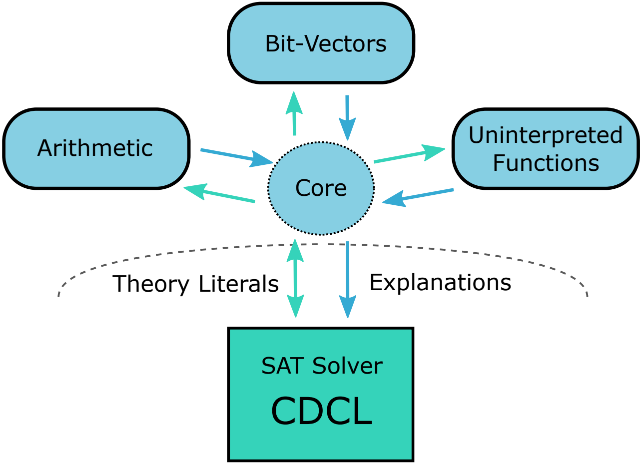
Incremental: Check T-satisfiability along the SAT solver.

Backtrack: Backtrack with SAT solver and keep context.

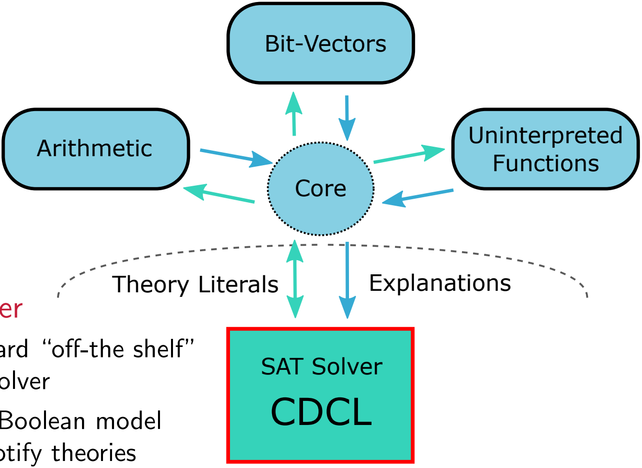
Propagation: If existing literals are implied tell SAT solver

Conflict: Small conflict explanations.

Solving: Typical Architecture



Solving: Typical Architecture



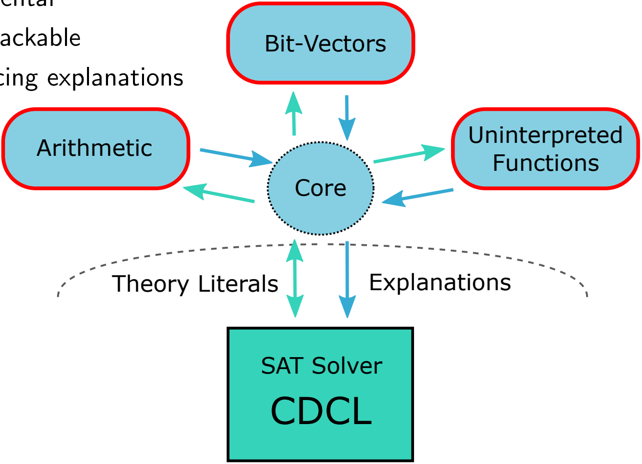
SAT Solver

- Standard "off-the shelf" SAT solver
- Build Boolean model and notify theories

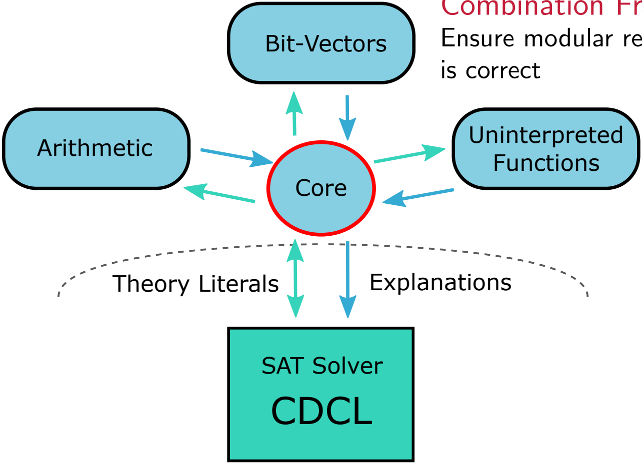
Solving: Typical Architecture

Theory Decision Procedures

- Check conjunctions of literals
- Incremental
- Backtrackable
- Producing explanations



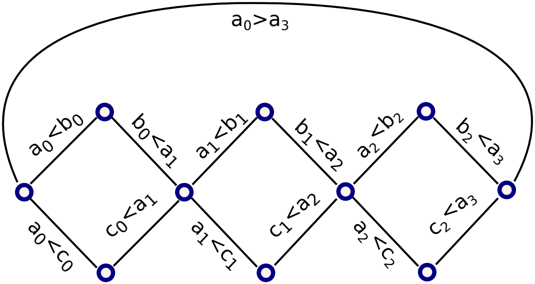
Solving: Typical Architecture



Combination Framework

Ensure modular reasoning is correct

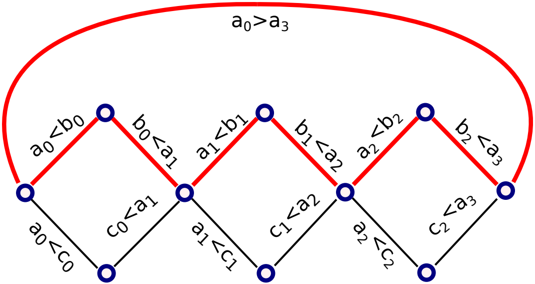
Solving: Great but not Perfect



Example (Diamonds)

$$a_0 > a_n \wedge \bigwedge_{k=0}^{n-1} \left((a_k < b_k \wedge b_k < a_{k+1}) \vee (a_k < c_k \wedge c_k < a_{k+1}) \right)$$

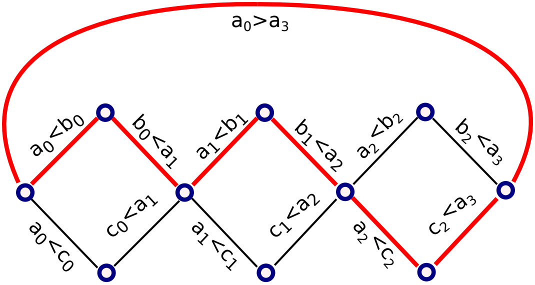
Solving: Great but not Perfect



Example (Diamonds)

$$a_0 > a_n \wedge \bigwedge_{k=0}^{n-1} ((a_k < b_k \wedge b_k < a_{k+1}) \vee (a_k < c_k \wedge c_k < a_{k+1}))$$

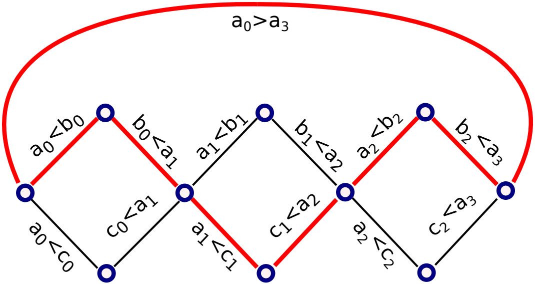
Solving: Great but not Perfect



Example (Diamonds)

$$a_0 > a_n \wedge \bigwedge_{k=0}^{n-1} ((a_k < b_k \wedge b_k < a_{k+1}) \vee (a_k < c_k \wedge c_k < a_{k+1}))$$

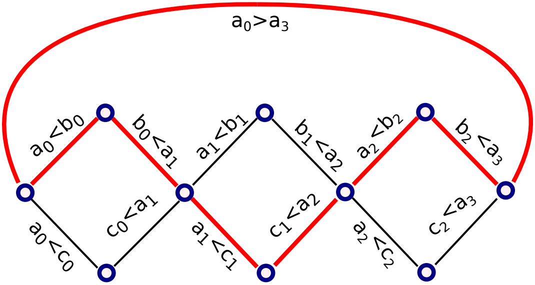
Solving: Great but not Perfect



Example (Diamonds)

$$a_0 > a_n \wedge \bigwedge_{k=0}^{n-1} ((a_k < b_k \wedge b_k < a_{k+1}) \vee (a_k < c_k \wedge c_k < a_{k+1}))$$

Solving: Great but not Perfect

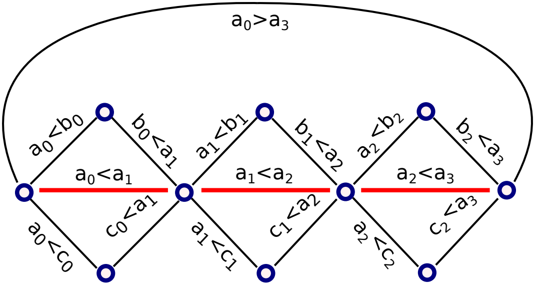


Example (Diamonds)

$$a_0 > a_n \wedge \bigwedge_{k=0}^{n-1} \left((a_k < b_k \wedge b_k < a_{k+1}) \vee (a_k < c_k \wedge c_k < a_{k+1}) \right)$$

And so on... Exponential enumeration of paths.

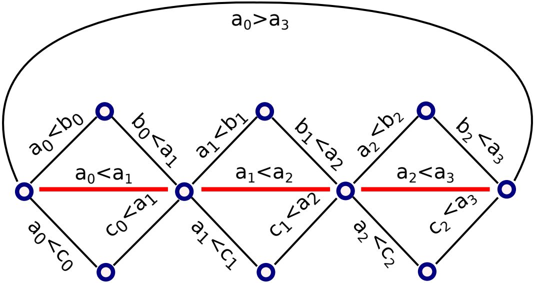
Solving: Great but not Perfect



Example (Diamonds)

$$a_0 > a_n \wedge \bigwedge_{k=0}^{n-1} \left((a_k < b_k \wedge b_k < a_{k+1}) \vee (a_k < c_k \wedge c_k < a_{k+1})\right)$$

Solving: Great but not Perfect



Example (Diamonds)

$$a_0 > a_n \wedge \bigwedge_{k=0}^{n-1} \left((a_k < b_k \wedge b_k < a_{k+1}) \vee (a_k < c_k \wedge c_k < a_{k+1}) \right)$$

Feature/Flaw: Can only use existing literals!

SMT Overview

SMT Solving

SMT Theories

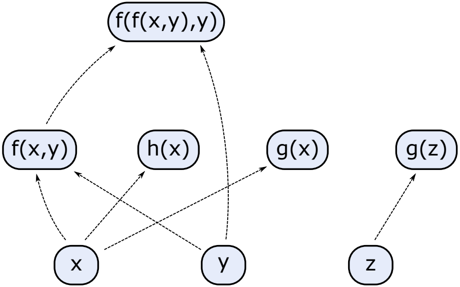
Theories: Uninterpreted Functions

- Literals are of the form $x = y$, $x \neq y$, $x = f(x,f(y,z))$.
- Can be decided in $O(n\log(n))$ based on congruence closure.
- Efficient theory propagation for equalities.
- Can generate:
  - small explanations [DNS05],
  - minimal explanations [NO07],
  - smallest explanations NP-hard [FFHP].
- Typically the core of the SMT solver and used in other theories.

Theories: Uninterpreted Functions and Congruence Closure

Example

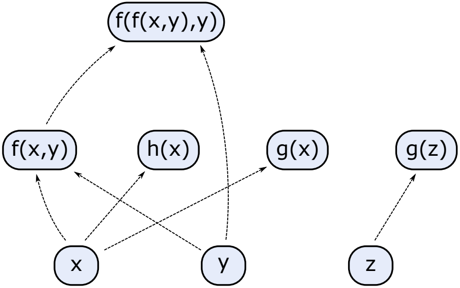
$$\llbracket\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\ \rrbracket$$



Theories: Uninterpreted Functions and Congruence Closure

Example

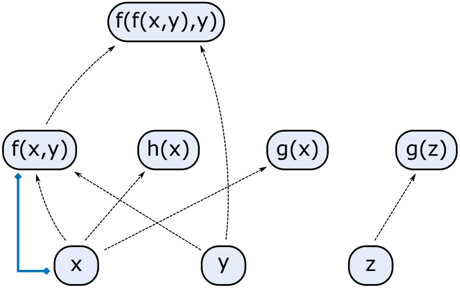
$$\llbracket\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\ \rrbracket$$



Theories: Uninterpreted Functions and Congruence Closure

Example

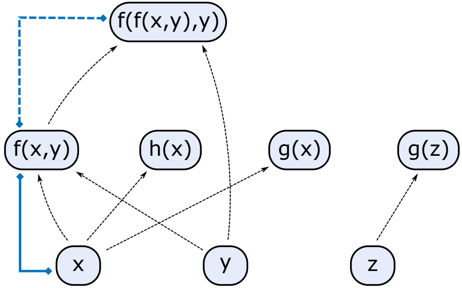
$$\llbracket\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\, \rrbracket$$



Theories: Uninterpreted Functions and Congruence Closure

Example

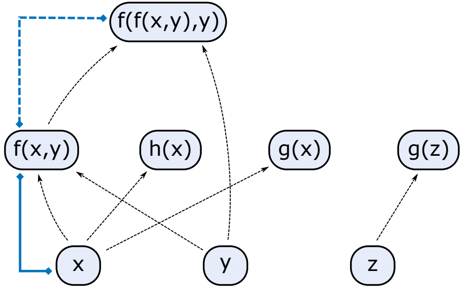
$$[\![\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\ ]\!]$$



Theories: Uninterpreted Functions and Congruence Closure

Example

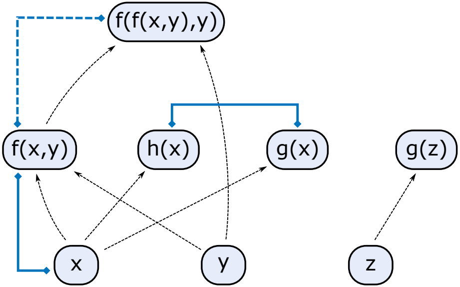
$$[\![\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\ ]\!]$$



Theories: Uninterpreted Functions and Congruence Closure

Example

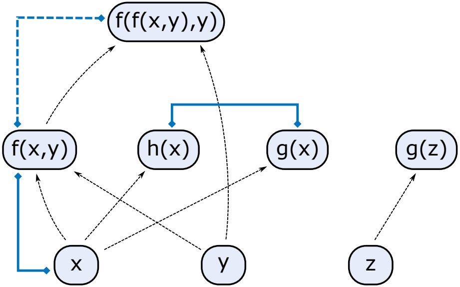
$$\llbracket\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\ \rrbracket$$



Theories: Uninterpreted Functions and Congruence Closure

Example

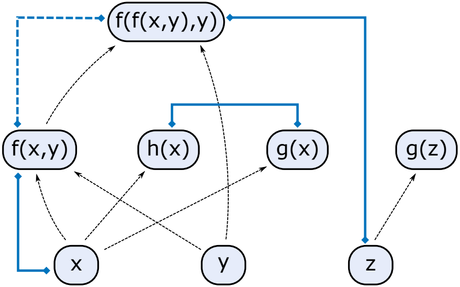
$$[\![\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\ ]\!]$$



Theories: Uninterpreted Functions and Congruence Closure

Example

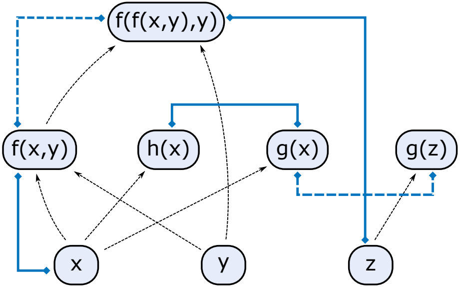
$$\llbracket\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\ \rrbracket$$



Theories: Uninterpreted Functions and Congruence Closure

Example

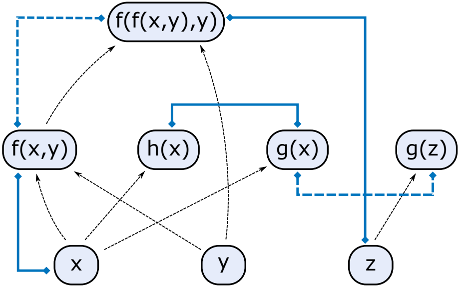
$$\llbracket\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\, \rrbracket$$



Theories: Uninterpreted Functions and Congruence Closure

Example

$$\llbracket\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ {\color{red} g(x) \neq g(z)}\ \rrbracket$$



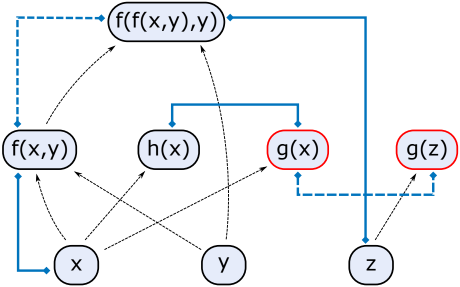
Theories: Uninterpreted Functions and Congruence Closure

Example

$$[\![\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z) \,]\!]$$

Conflict:

1. $g(x) \neq g(z)$



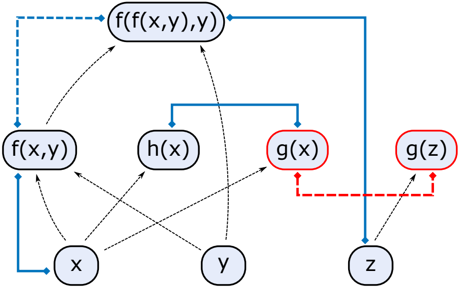
Theories: Uninterpreted Functions and Congruence Closure

Example

$$\llbracket f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\ \rrbracket$$

Conflict:

1. $g(x) \neq g(z)$



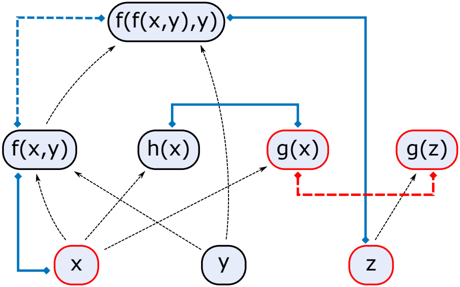
Theories: Uninterpreted Functions and Congruence Closure

Example

$$[\![\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\ ]\!]$$

Conflict:

1. $g(x) \neq g(z)$



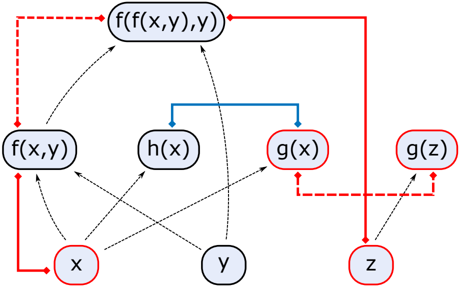
Theories: Uninterpreted Functions and Congruence Closure

Example

$$\llbracket f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\ \rrbracket$$

Conflict:

1. $g(x) \neq g(z)$



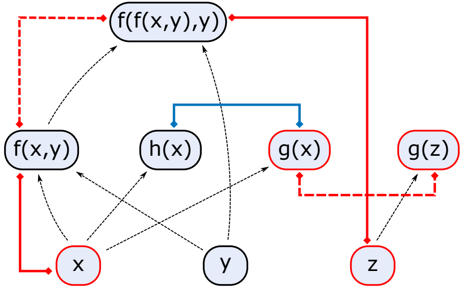
Theories: Uninterpreted Functions and Congruence Closure

Example

$$\llbracket\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z)\, \rrbracket$$

Conflict:

1. $g(x) \neq g(z)$
2. $f(f(x,y),y) = z$



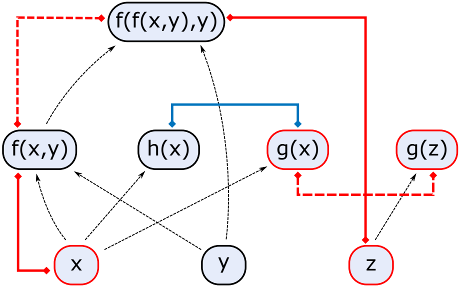
Theories: Uninterpreted Functions and Congruence Closure

Example

$$[\![\, f(x,y) = x,\ h(x) = g(x),\ f(f(x,y),y) = z,\ g(x) \neq g(z) \,]\!]$$

Conflict:

1. $g(x) \neq g(z)$
2. $f(f(x,y),y) = z$
3. $f(x,y) = x$



Theories: Difference Logic

- Literals are of the form $x - y \bowtie k$, where
  - $\bowtie \in \{\leq, <, =, \neq, >, \geq\}$,
  - x and y are arithmetic variables (integer or real),
  - k is a constant (integer or real).
- We can rewrite $x - y = k$ to $(x - y \leq k) \wedge (x - y \geq k)$.
- In integers, we can rewrite $x - y < k$ to $x - y \leq k - 1$.
- In reals, we can rewrite $x - y < k$ to $x - y \leq k - \delta$.
- Can assume: all literals of the form $x - y \leq k$.

Theories: Difference Logic Theory

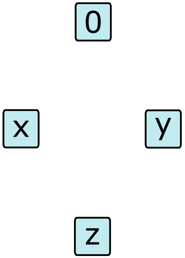
- Any solution to a set of literals can be shifted:
  - if v is a satisfying assignment, so is $v'(x) = v(x) + k$.
- We can use this to also process simple bounds $x \leq k$:
  - introduce fresh variable z (for zero),
  - rewrite each $x \leq k$ to $x - z \leq k$,
  - given a solution v, shift it so that $v(z) = 0$.
- If we allow (dis)equalities as literals, then:
  - in reals, satisfiability is polynomial;
  - in integers, satisfiability is NP-hard.

Theories: Difference Logic Example

Example

$$\llbracket\, x \leq 1,\ x - y \leq 2,\ y - z \leq 3,\ z - x \leq -6 \,\rrbracket$$

- Construct a graph from literals.
- Check if there is a negative path.

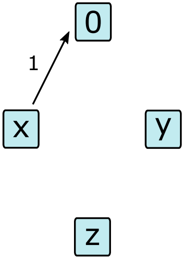


Theories: Difference Logic Example

Example

$$[\![\, x \leq 1,\ x - y \leq 2,\ y - z \leq 3,\ z - x \leq -6 \,]\!]$$

- Construct a graph from literals.
- Check if there is a negative path.

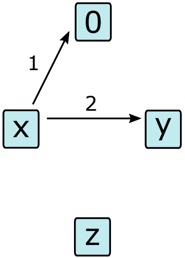


Theories: Difference Logic Example

Example

$$[\![\, x \leq 1,\ x - y \leq 2,\ y - z \leq 3,\ z - x \leq -6 \,]\!]$$

- Construct a graph from literals.
- Check if there is a negative path.

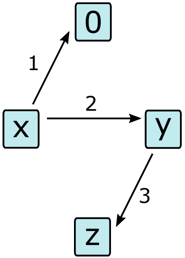


Theories: Difference Logic Example

Example

$$[\![\, x \leq 1,\ x - y \leq 2,\ y - z \leq 3,\ z - x \leq -6 \,]\!]$$

- Construct a graph from literals.
- Check if there is a negative path.

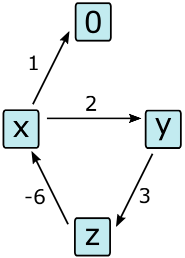


Theories: Difference Logic Example

Example

$$[\![\ x \leq 1,\ x - y \leq 2,\ y - z \leq 3,\ z - x \leq -6\ ]\!]$$

- Construct a graph from literals.
- Check if there is a negative path.

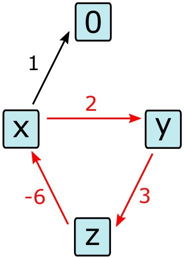


Theories: Difference Logic Example

Example

$$[\![\, x \leq 1,\ x - y \leq 2,\ y - z \leq 3,\ z - x \leq -6 \,]\!]$$

- Construct a graph from literals.
- Check if there is a negative path.



Theories: Difference Logic Example

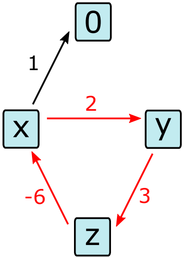
Example

$$\llbracket\, x \leq 1,\; x - y \leq 2,\; y - z \leq 3,\; z - x \leq -6 \,\rrbracket$$

- Construct a graph from literals.
- Check if there is a negative path.

Theorem

literals unsatisfiable $\Leftrightarrow \exists$ *negative path.*

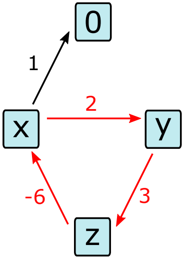


Theories: Difference Logic Example

Example

$$[\![\ x \leq 1,\ x - y \leq 2,\ y - z \leq 3,\ z - x \leq -6\ ]\!]$$

- Construct a graph from literals.
- Check if there is a negative path.



Theorem
literals unsatisfiable $\Leftrightarrow \exists$ *negative path.*

- Conflict:
  $(x - y \leq 2)$,
  $(y - z \leq 3)$,
  $(z - x \leq -6)$.

Theories: Arrays

$$\forall a, i, e \ : \ \mathsf{store}(a, i, e)[i] = e$$
$$\forall a, i, j, e \ : \ i \neq j \Rightarrow \mathsf{store}(a, i, e)[j] = a[j]$$
$$\forall a, b \ : \ a \neq b \Rightarrow \exists i : a[i] \neq b[i]$$

Common approach:

- UF + lemmas on demand [BB09, DMB09].
- Use UF as if store and $\_[\_]$ were uninterpreted.
- If UNSAT in UF, then UNSAT in arrays too.
- If SAT and solution respects array axioms, then SAT (lucky).
- If not, then refine by instantiating violated axioms.

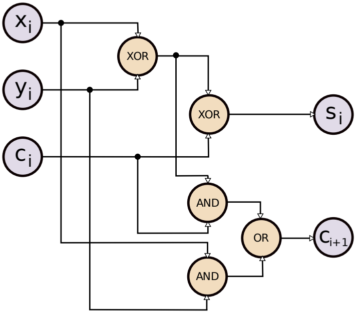
Theories: Bit-Vectors

Common approach:

1. Heavy preprocessing
2. Encode into SAT (bit-blasting)
3. Run a SAT solver

Alternatives [HBJ$^+$14, ZWR16] not yet mature.

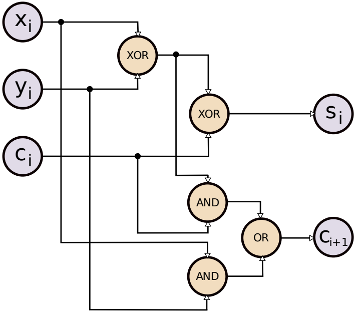
Theories: Bit-Vectors and Bit-Blasting



Translation to CNF

- Each node a new variables
- XOR introduces 4 clauses
- AND introduces 3 clauses
- OR introduces 3 clauses
- **17 new clauses**
- **5 new variables**

Theories: Bit-Vectors and Bit-Blasting



Translation to CNF

- Each node a new variables
- XOR introduces 4 clauses
- AND introduces 3 clauses
- OR introduces 3 clauses
- **17 new clauses**
- **5 new variables**

Bit-Blasting Addition/Multiplication

$x_{[32]} + y_{[32]}$ 544 new clauses, 160 new variables

$x_{[32]} \times y_{[32]}$ 10016 new clauses, 3008 new variables

References I

Robert Brummayer and Armin Biere.
Lemmas on demand for the extensional theory of arrays.
Journal on Satisfiability, Boolean Modeling and Computation, 6:165–201, 2009.

Martin Brain, Vijay D'Silva, Alberto Griggio, Leopold Haller, and Daniel Kroening.
Deciding floating-point logic with abstract conflict driven clause learning.
Formal Methods in System Design, 45(2):213–245, 2014.

Clark Barrett, Igor Shikanian, and Cesare Tinelli.
An abstract decision procedure for satisfiability in the theory of recursive data types.
Electronic Notes in Theoretical Computer Science, 174(8):23–37, 2007.

Leonardo De Moura and Nikolaj Bjørner.
Efficient e-matching for smt solvers.
In *International Conference on Automated Deduction*, pages 183–198. Springer, 2007.

Leonardo De Moura and Nikolaj Bjørner.
Generalized, efficient array decision procedures.
In *Formal Methods in Computer-Aided Design*, pages 45–52. IEEE, 2009.

Leonardo De Moura and Nikolaj Bjørner.
Satisfiability modulo theories: introduction and applications.
Communications of the ACM, 54(9):69–77, 2011.

Leonardo De Moura and Harald Rueß.
Lemmas on demand for satisfiability solvers.
2002.

References II

David Detlefs, Greg Nelson, and James B Saxe.
Simplify: a theorem prover for program checking.
Journal of the ACM (JACM), 52(3):365–473, 2005.

Andreas Fellner, Pascal Fontaine, Georg Hofferek, and Bruno Woltzenlogel Paleo.
Np-completeness of small conflict set generation for congruence closure.

Sicun Gao, Soonho Kong, and Edmund M Clarke.
Satisfiability modulo ODEs.
In *Formal Methods in Computer-Aided Design (FMCAD), 2013*, pages 105–112. IEEE, 2013.

Liana Hadarean, Kshitij Bansal, Dejan Jovanović, Clark Barrett, and Cesare Tinelli.
A tale of two solvers: Eager and lazy approaches to bit-vectors.
In *International Conference on Computer Aided Verification*, pages 680–695. Springer, 2014.

Adam Kiezun, Vijay Ganesh, Philip J Guo, Pieter Hooimeijer, and Michael D Ernst.
HAMPI: a solver for string constraints.
In *Proceedings of the eighteenth international symposium on Software testing and analysis*, pages 105–116. ACM, 2009.

Tianyi Liang, Andrew Reynolds, Cesare Tinelli, Clark Barrett, and Morgan Deters.
A DPLL(T) theory solver for a theory of strings and regular expressions.
In *International Conference on Computer Aided Verification*, pages 646–662. Springer, 2014.

John McCarthy.
Towards a mathematical science of computation.
In *Program Verification*, pages 35–56. Springer, 1993.

References III

Anthony Narkawicz and César A Munoz.
Formal verification of conflict detection algorithms for arbitrary trajectories.
Reliable Computing, 17(2):209–237, 2012.

Robert Nieuwenhuis and Albert Oliveras.
Fast congruence closure and extensions.
Information and Computation, 205(4):557–580, 2007.

Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli.
Abstract dpll and abstract dpll modulo theories.
In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 36–50. Springer, 2005.

Andrew Reynolds, Cesare Tinelli, Amit Goel, Sava Krstić, Morgan Deters, and Clark Barrett.
Quantifier instantiation techniques for finite model finding in smt.
In *International Conference on Automated Deduction*, pages 377–391. Springer, 2013.

Aleksandar Zeljić, Christoph M Wintersteiger, and Philipp Rümmer.
Approximations for model construction.
In *International Joint Conference on Automated Reasoning*, pages 344–359. Springer, 2014.

Aleksandar Zeljić, Christoph M Wintersteiger, and Philipp Rümmer.
Deciding bit-vector formulas with mcsat.
In *International Conference on Theory and Applications of Satisfiability Testing*, pages 249–266. Springer, 2016.