Introduction to Satisfiability Modulo Theories

Dejan Jovanović

SRI International

SAT/SMT/AR Summer School, Lisbon, 2016

Outline

Outline

Satisfiability Modulo Theories

what we're trying to solve

Problem

Check if a given (quantifier-free) formula is satisfiable modulo the union of background theories.

Example (QF_UFLRA)

$$(z = 1 \lor z = 0) \land (x - y + z = 1) \land (f(x) > f(y))$$

Satisfiability Modulo Theories

what we're trying to solve

Problem

Check if a given (quantifier-free) formula is satisfiable modulo the union of 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).

Satisfiability Modulo Theories

what we're trying to solve

Problem

Check if a given (quantifier-free) formula is satisfiable modulo the union of background theories.

Example (QF_UFLRA)

$$(z = 1 \lor z = 0) \land (x - y + z = 1) \land (f(x) > f(y))$$

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

Satisfiability Modulo Theories

what we're trying to solve

Problem

Check if a given (quantifier-free) formula is satisfiable modulo the union of 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 $z \mapsto 0, x \mapsto 1, y \mapsto 0, f(1) \mapsto 1, f(0) \mapsto 0$

Satisfiability Modulo Theories

many applications

Example

Schedule n jobs, each composed of m consecutive tasks, on m machines.

Schedule in 8 time slots.

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

Example from [DMB11]

Satisfiability Modulo Theories

many applications

Example

Schedule *n* jobs, each composed of *m* consecutive tasks, on *m* machines.

Schedule in 8 time slots.

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

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

Example from [DMB11]

Satisfiability Modulo Theories

many applications

Example

Schedule *n* jobs, each composed of *m* consecutive tasks, on *m* machines.

Schedule in 8 time slots.

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

$(t_{1,1} \geq 0) \wedge (t_{1,2} \geq t_{1,1} + 2) \wedge (t_{1,2} + 1 \leq 8)$

$(t_2$

$(t_3$

Run SMT solver (QF_IDL)

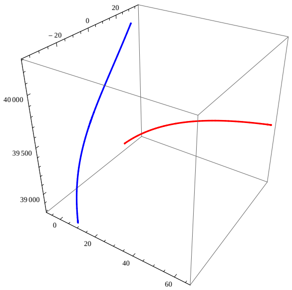
$$t_{1,1} \mapsto 5, \quad t_{1,2} \mapsto 7$$
$$t_{2,1} \mapsto 2, \quad t_{2,2} \mapsto 6$$
$$t_{3,1} \mapsto 0, \quad t_{3,2} \mapsto 3$$

$((t_{2,2} \geq t_{3,2} + 3) \vee (t_{3,2} \geq t_{2,2} + 1))$

Example from [DMB11]

Satisfiability Modulo Theories

many applications



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

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

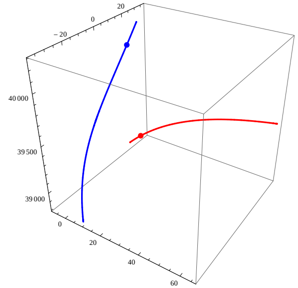
$$D = 5 \qquad H = 1000 \qquad 0 \leq t \leq \frac{1}{20}$$

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

Example from [NM12]

Satisfiability Modulo Theories

many applications



$$T_1^x(t) = -3.2484 + 270.7t + 433.12t^2 - 324.83999t^3$$

$$T_1^y(t) = 15.1592 + 108.28t + 121.2736t^2 - 649.67999t^3$$

$$T_1^z(t) = 38980.8 + 5414t - 21656t^2 + 32484t^3$$

$$T_2^x(t) = 1$$

$$T_2^y(t) = 1$$

$$T_2^z(t) = 4$$

Run SMT solver (QF_NRA)

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

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

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

Example from [NM12]

Satisfiability Modulo Theories

many applications

Example

```
void swap(int* a, int* b) {
  *a = *a + *b;
  *b = *a - *b;
  *a = *a - *b;
}
```

Check if the swap is correct:

- Heap: array $BV_{32} \mapsto BV_{32}$
- Update heap line by line
- Check if correct

Satisfiability Modulo Theories

many applications

Example

```
void swap(int* a, int* b) {
  *a = *a + *b;
  *b = *a - *b;
  *a = *a - *b;
}
```

Check if the swap is correct:

- Heap: array $BV_{32} \mapsto BV_{32}$
- Update heap line by line
- Check if correct

$$h_1 = \text{store}(h_0, a, h_0[a] +_{32} h_0[b])$$
$$h_2 = \text{store}(h_1, b, h_1[a] -_{32} h_1[b])$$
$$h_3 = \text{store}(h_2, a, h_2[a] -_{32} h_2[b])$$
$$\neg(h_3[a] = h_0[b] \wedge h_3[b] = h_0[a])$$

Satisfiability Modulo Theories

many applications

Example

```
void swap(int* a, int* b) {
  *a = *a + *b;
  *b = *a - *b;
  *a = *a - *b;
}
```

Run SMT solver (QF_ABV)

$$a \mapsto 0, b \mapsto 0$$

$$h_0[0] \mapsto 1, h_1[0] \mapsto 2$$

$$h_2[0] \mapsto 0, h_3[0] \mapsto 0$$

Check if the swap is correct:

- Heap: array $BV_{32} \mapsto BV_{32}$
- Update heap line by line
- Check if correct
- Incorrect: aliasing

$$h_1 = \mathsf{store}(h_0, a, h_0[a] +_{32} h_0[b])$$

$$h_2 = \mathsf{store}(h_1, b, h_1[a] -_{32} h_1[b])$$

$$h_3 = \mathsf{store}(h_2, a, h_2[a] -_{32} h_2[b])$$

$$\neg(h_3[a] = h_0[b] \wedge h_3[b] = h_0[a])$$

Satisfiability Modulo Theories

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.

Satisfiability Modulo Theories

common theories of interest

Uninterpreted Functions (QF_UF)

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

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

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

Satisfiability Modulo Theories

common theories of interest

Theory of Arrays [McC93]

Operates over sorts 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

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

Satisfiability Modulo Theories

common theories of interest

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):

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

Satisfiability Modulo Theories

common theories of interest

Bitvectors (QF_BV)

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

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

Semantics similar to programming languages.

Example (a is 32-bits)

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

Satisfiability Modulo Theories

some other interesting theories

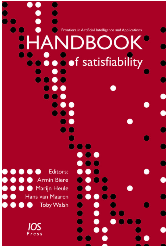
Some other theories

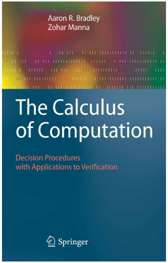
- Floating point [BDG$^+$14, ZWR14]
- Inductive data-types [BST07]
- Strings and regular expressions [LRT$^+$14, KGG$^+$09]
- Quantifiers [DMB07, RTG$^+$13]
- Differential Equations [GKC13]
- ...

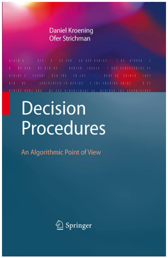
Satisfiability Modulo Theories

more information

- Books and chapters [BSST09, BM07, KS08]







- Online
  - SMT-LIB at `http://smtlib.cs.uiowa.edu`
  - SMT-COMP at `http://smtcomp.sourceforge.net`

Outline

SMT Problem

how to solve it?

Check T-satisfiability of a T-formula F

SMT Problem

how to solve it?

Check T-satisfiability of a T-formula F

1. Convert to DNF

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

SMT Problem

how to solve it?

Check T-satisfiability of a T-formula F

1. Convert to DNF

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

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

SMT Problem

how to solve it?

Check T-satisfiability of a T-formula F

1. Convert to DNF

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

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.

SMT Problem

sat solver instead of dnf?

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.

Very Lazy SMT

example

View

Theory

$$\neg a = b$$

$$(x = a \vee x = b)$$

$$(y = a \vee y = b)$$

$$(z = a \vee z = b)$$

$$\neg x = y$$

Very Lazy SMT

example

View

Boolean

$$\neg B_1$$
$$(B_2 \vee B_3)$$
$$(B_4 \vee B_5)$$
$$(B_6 \vee B_7)$$
$$\neg B_8$$

Very Lazy SMT

example

View

Boolean

$$
\begin{array}{c}
\neg \; B_1 \\
(\; B_2 \; \vee \; B_3 \;) \\
(\; B_4 \; \vee \; B_5 \;) \\
(\; B_6 \; \vee \; B_7 \;) \\
\neg \; B_8
\end{array}
$$

Check with SAT solver

Very Lazy SMT

example

View

Boolean

$$\neg \quad B_1$$
$$(\quad B_2 \quad \vee \quad B_3 \quad)$$
$$(\quad B_4 \quad \vee \quad B_5 \quad)$$
$$(\quad B_6 \quad \vee \quad B_7 \quad)$$
$$\neg \quad B_8$$

Check with SAT solver

$$[\![\neg B_1, \neg B_8, \neg B_3, B_2, \neg B_5, B_4, \neg B_7, B_6]\!]$$

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

$$\llbracket\ \neg a = b,\ \neg x = y,\ \neg x = b,\ x = a,\ \neg y = b,\ y = a,\ \neg z = b,\ z = a\ \rrbracket$$

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 T-solver

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

Check with SAT solver

$$[\![\neg a = b,\ \color{red}{\neg x = y},\ \neg x = b,\ \color{red}{x = a},\ \neg y = b,\ \color{red}{y = a},\ \neg z = b,\ z = a]\!]$$

Very Lazy SMT

example

View

Theory

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

Block

Add clause

Check with SAT solver

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

Very Lazy SMT

example

View

Boolean

$$
\begin{array}{c}
\neg B_1 \\
(B_2 \lor B_3) \\
(B_4 \lor B_5) \\
(B_6 \lor B_7) \\
\neg B_8 \\
(B_8 \lor \neg B_2 \lor \neg B_4)
\end{array}
$$

Check with SAT solver

Very Lazy SMT

example

View

Boolean

$$\neg B_1$$
$$(B_2 \vee B_3)$$
$$(B_4 \vee B_5)$$
$$(B_6 \vee B_7)$$
$$\neg B_8$$
$$(B_8 \vee \neg B_2 \vee \neg B_4)$$

Check with SAT solver

$$[\![\neg B_1, \neg B_8, \neg B_3, B_2, \neg B_4, B_5, \neg B_7, B_6]\!]$$

Very Lazy SMT

example

View

Theory

$$\neg a = b$$

$$(x = a \vee x = b)$$

$$(y = a \vee y = b)$$

$$(z = a \vee z = b)$$

$$\neg x = y$$

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

Check with SAT solver

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

Very Lazy SMT

example

View

Theory

Check with T-solver

Satisfiable

$$a, x, z \mapsto c_1$$

$$b, y \mapsto c_2$$

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

Check with SAT solver

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

Very Lazy SMT

discussion

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 :(.

Very Lazy SMT

discussion

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 the SAT solver: DPLL(T) [DMR02, NOT05]

Incremental: Check *T*-satisfiability along the SAT solver.

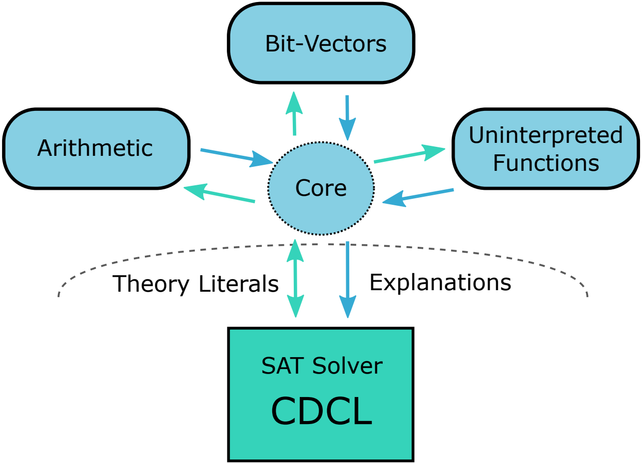
Backtrack: Backtrack with SAT solver and keep context.

Propagation: If existing literals are implied tell SAT solver

Conflict: Small conflict explanations.

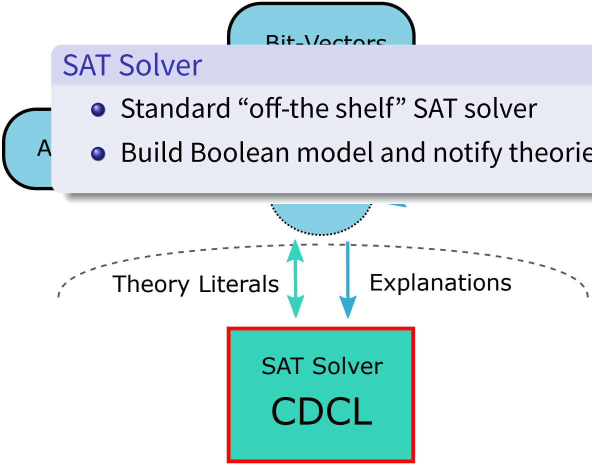
DPLL(T) Framework

typical architecture



DPLL(T) Framework

typical architecture

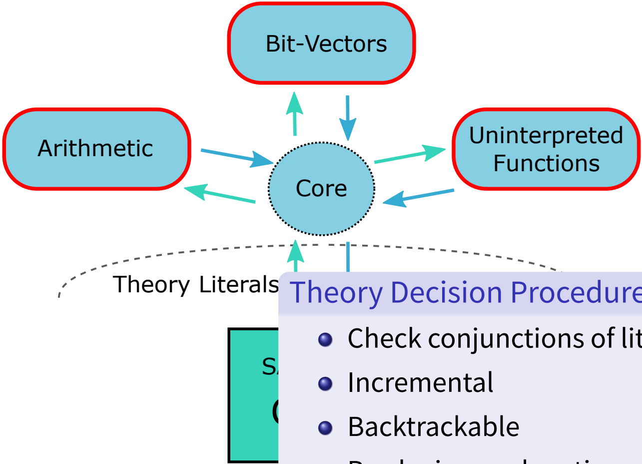


SAT Solver

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

DPLL(T) Framework

typical architecture

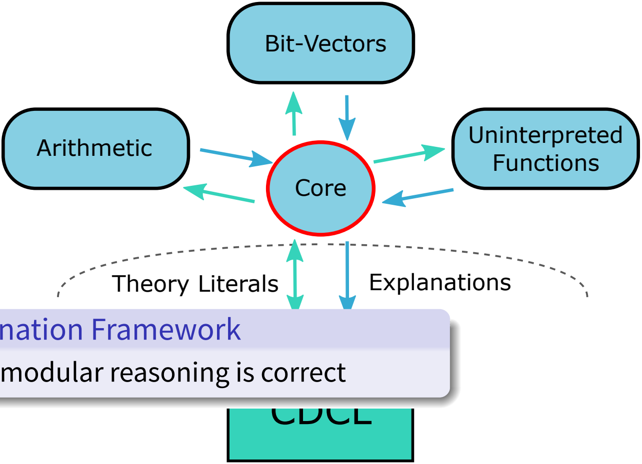


Theory Decision Procedures

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

DPLL(T) Framework

typical architecture

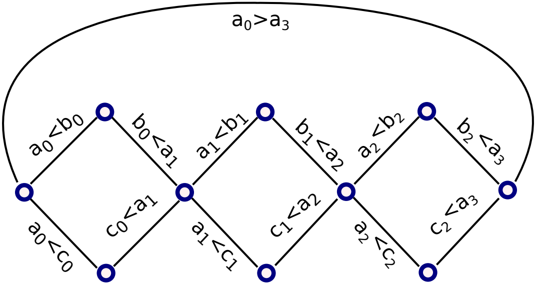


Combination Framework

Ensure modular reasoning is correct

DPLL(T) Framework

great but not perfect

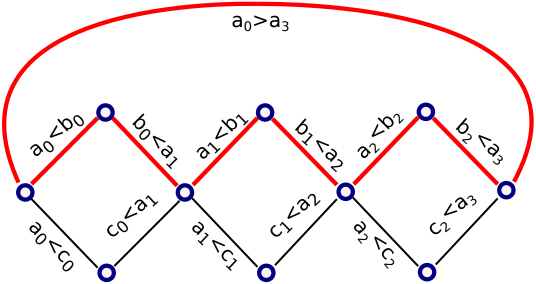


Example (Diamonds)

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

DPLL(T) Framework

great but not perfect

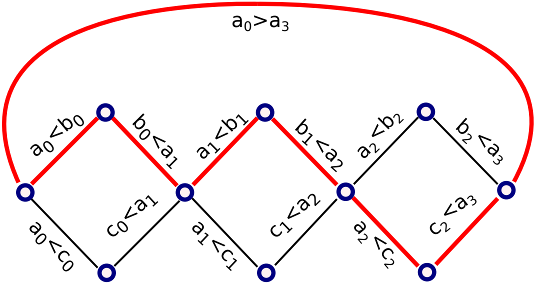


Example (Diamonds)

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

DPLL(T) Framework

great but not perfect

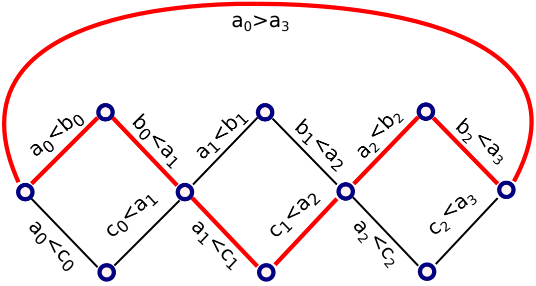


Example (Diamonds)

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

DPLL(T) Framework

great but not perfect

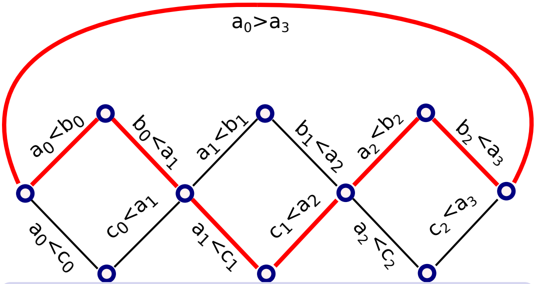


Example (Diamonds)

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

DPLL(T) Framework

great but not perfect



And so on...

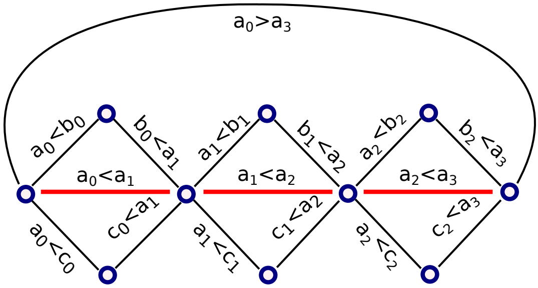
Exponential enumeration of paths.

Example (D

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

DPLL(T) Framework

great but not perfect

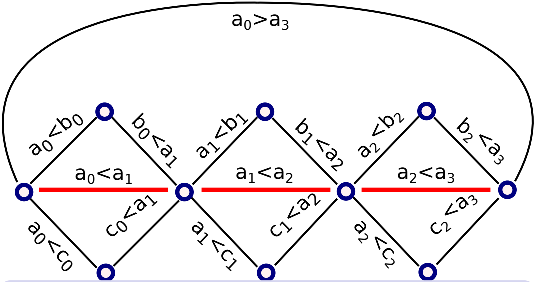


Example (Diamonds)

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

DPLL(T) Framework

great but not perfect



Feature/Flaw

Can only use existing literals!

Example (D

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

Outline

Uninterpreted Functions

the theory

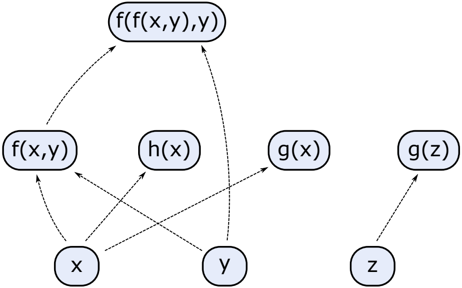
- Literals are of the form $x = y, x \neq y, x = f(x, f(y, z))$.
- Can be decided in $O(n \log(n))$ based on congruence closure.
- Efficient theory propagation for equalities.
- Can generate:
  - small explanations [DNS05],
  - minimal explanations [NO07],
  - smallest explanations NP-hard [FFHP].
-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

Uninterpreted Functions

congruence closure

Example

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

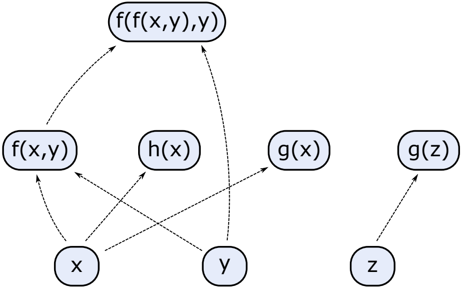


Uninterpreted Functions

congruence closure

Example

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

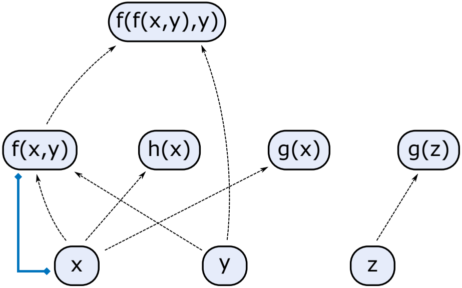


Uninterpreted Functions

congruence closure

Example

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

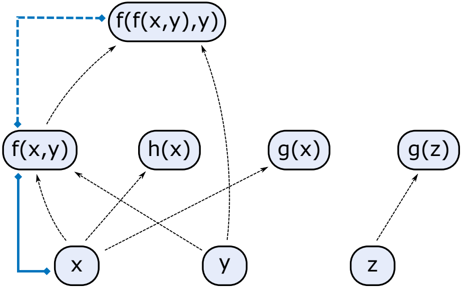


Uninterpreted Functions

congruence closure

Example

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

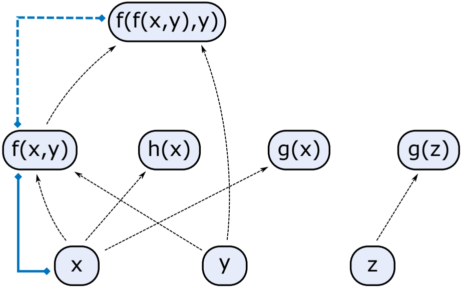


Uninterpreted Functions

congruence closure

Example

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

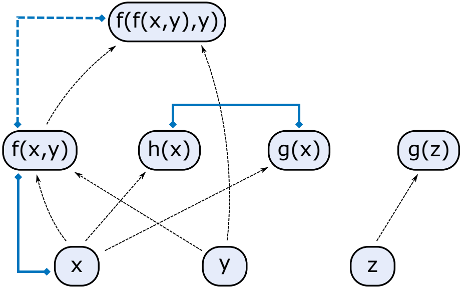


Uninterpreted Functions

congruence closure

Example

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

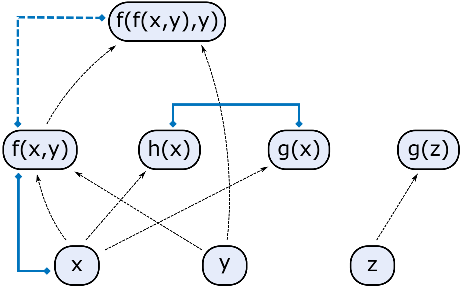


Uninterpreted Functions

congruence closure

Example

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

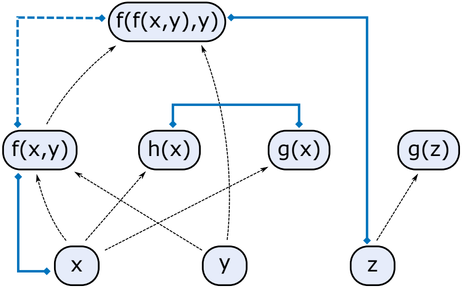


Uninterpreted Functions

congruence closure

Example

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

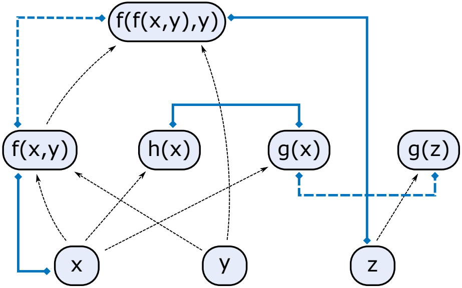


Uninterpreted Functions

congruence closure

Example

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

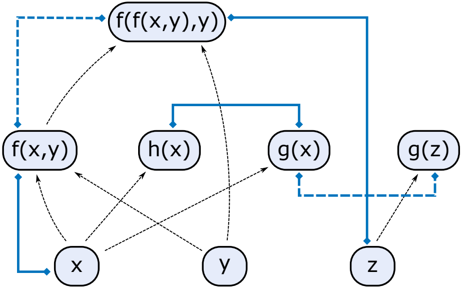


Uninterpreted Functions

congruence closure

Example

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



Uninterpreted Functions

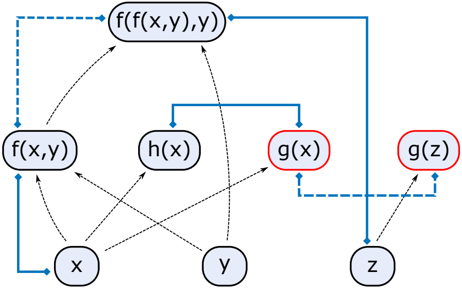
congruence closure

Example

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

Conflict:

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



Uninterpreted Functions

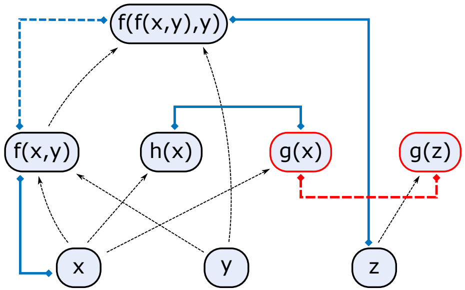
congruence closure

Example

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

Conflict:

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



Uninterpreted Functions

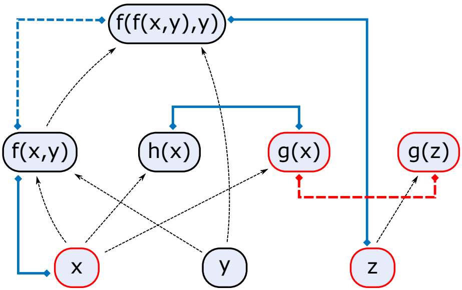
congruence closure

Example

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

Conflict:

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



Uninterpreted Functions

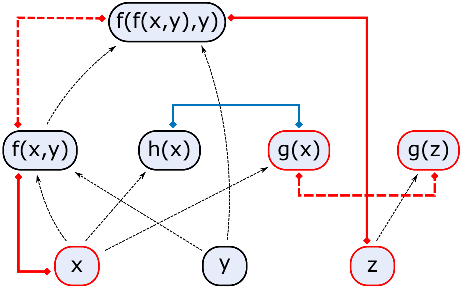
congruence closure

Example

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

Conflict:

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



Uninterpreted Functions

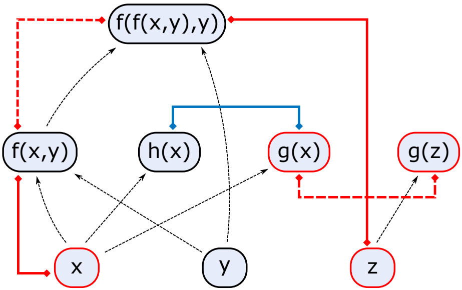
congruence closure

Example

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

Conflict:

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



Uninterpreted Functions

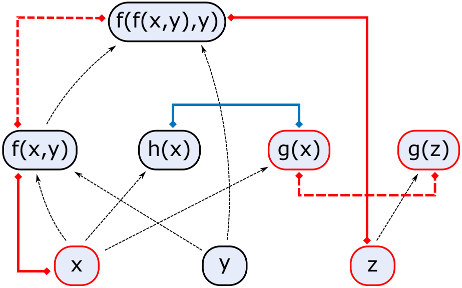
congruence closure

Example

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

Conflict:

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



Difference Logic

the theory

- 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 $x - y \leq k$.

Difference Logic

the 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.

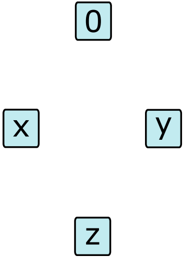
Difference Logic

from literals to graph

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.



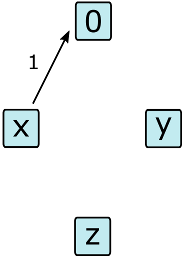
Difference Logic

from literals to graph

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.



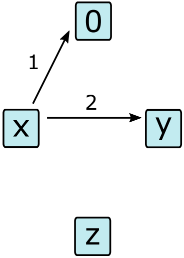
Difference Logic

from literals to graph

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.



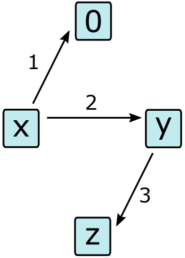
Difference Logic

from literals to graph

Example

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

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



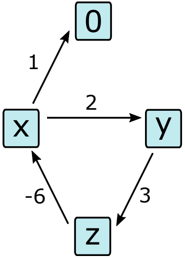
Difference Logic

from literals to graph

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.



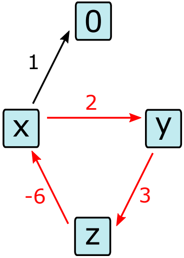
Difference Logic

from literals to graph

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.



Difference Logic

from literals to graph

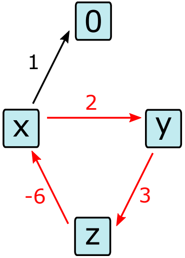
Example

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

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

Theorem

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



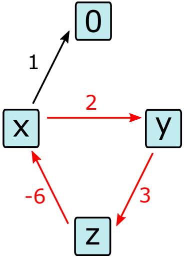
Difference Logic

from literals to graph

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

Linear Arithmetic

the theory

Language:

- Atoms are of the form $a_1 x_1 + \cdots a_n x_n \leq b$.
- We can rewrite $t = b$ to $(t \leq b) \wedge (t \geq b)$.
- In integers, we can rewrite $t < b$ to $t \leq b - 1$.
- In reals, we can rewrite $t < b$ to $t \leq b - \delta$.

Variant of simplex designed for DPLL(T) [DDM06]:

- Incremental
- Cheap backtracking
- Can do theory propagation
- Can generate minimal explanations
- Worst case exponential (fast in practice)

Linear Arithmetic

tableau

- Rewrite each $\sum a_i x_i \leq b$ to $s \leq b$ with $s = \sum a_i x_i$.
- We get tableau of equations + simple bounds on variables.
  - Tableau is fixed.
  - Bounds can be asserted and retracted.

Tableau

$$
\begin{aligned}
s_1 &= a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n \\
&\vdots \\
s_i &= a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n \\
&\vdots \\
s_m &= a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n
\end{aligned}
$$

Bounds

$$
\begin{aligned}
-\infty \leq x_1 &\leq u_1 \\
l_2 \leq x_2 &\leq +\infty \\
&\vdots \\
l_i \leq s_i &\leq u_i \\
l_j \leq x_j &\leq u_j
\end{aligned}
$$

Linear Arithmetic

tableau

Tableau

$$
\begin{aligned}
s_1 &= a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n \\
&\vdots \\
s_i &= a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n \\
&\vdots \\
s_m &= a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n
\end{aligned}
$$

Bounds

$$
\begin{gathered}
-\infty \le x_1 \le u_1 \\
l_2 \le x_2 \le +\infty \\
\vdots \\
l_i \le s_i \le u_i \\
l_j \le x_j \le u_j
\end{gathered}
$$

Linear Arithmetic

tableau

Tableau

$$s_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n$$

$$\vdots$$

$$s_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n$$

$$\vdots$$

$$s_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n$$

Bounds

$$-\infty \leq x_1 \leq u_1$$

$$l_2 \leq x_2 \leq +\infty$$

$$\vdots$$

$$l_i \leq s_i \leq u_i$$

$$l_j \leq x_j \leq u_j$$

- Variables can be basic and non-basic in the tableau.

Linear Arithmetic

tableau

Tableau

$$s_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n$$
$$\vdots$$
$$s_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n$$
$$\vdots$$
$$s_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n$$

Bounds

$$-\infty \leq x_1 \leq u_1$$
$$l_2 \leq x_2 \leq +\infty$$
$$\vdots$$
$$l_i \leq s_i \leq u_i$$
$$l_j \leq x_j \leq u_j$$

- Variables can be basic and non-basic in the tableau.

Linear Arithmetic

tableau

Tableau

$$
\begin{aligned}
s_1 &= a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n \\
&\vdots \\
s_i &= a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n \\
&\vdots \\
s_m &= a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n
\end{aligned}
$$

Bounds

$$
\begin{aligned}
-\infty \leq x_1 &\leq u_1 \\
l_2 \leq x_2 &\leq +\infty \\
&\vdots \\
l_i \leq s_i &\leq u_i \\
l_j \leq x_j &\leq u_j
\end{aligned}
$$

- Variables can be basic and non-basic in the tableau.

Linear Arithmetic

tableau

Tableau

$$
\begin{aligned}
s_1 &= a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n \\
&\vdots \\
s_i &= a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n \\
&\vdots \\
s_m &= a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n
\end{aligned}
$$

Bounds

$$
\begin{gathered}
-\infty \leq x_1 \leq u_1 \\
l_2 \leq x_2 \leq +\infty \\
\vdots \\
l_i \leq s_i \leq u_i \\
l_j \leq x_j \leq u_j
\end{gathered}
$$

- Variables can be basic and non-basic in the tableau.
- Keep an assignment v of all variables:
  - v satisfies the tableau,
  - v satisfies bounds on the non-basic variables.

Linear Arithmetic

tableau

Tableau

$$
\begin{aligned}
s_1 &= a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n \\
&\vdots \\
s_i &= a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n \\
&\vdots \\
s_m &= a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n
\end{aligned}
$$

Bounds

$$
\begin{aligned}
-\infty \leq x_1 &\leq u_1 \\
l_2 \leq x_2 &\leq +\infty \\
&\vdots \\
l_i \leq s_i &\leq u_i \\
l_j \leq x_j &\leq u_j
\end{aligned}
$$

- Variables can be basic and non-basic in the tableau.
- Keep an assignment v of all variables:
  - v satisfies the tableau,
  - v satisfies bounds on the non-basic variables.
- Initially $v(x) = 0$ and $-\infty \leq x \leq +\infty$.

Linear Arithmetic

tableau

Tableau

$$s_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n$$

$$\vdots$$

$$s_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n$$

$$\vdots$$

$$s_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n$$

Bounds

$$-\infty \leq x_1 \leq u_1$$

$$l_2 \leq x_2 \leq +\infty$$

$$\vdots$$

$$l_i \leq s_i \leq u_i$$

$$l_j \leq x_j \leq u_j$$

Case 1:

- v satisfies bound on the basic variables too.
- Satisfiable, v is the model!

Linear Arithmetic

tableau

Tableau

$$s_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n$$

$$\vdots$$

$$s_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n$$

$$\vdots$$

$$s_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n$$

Bounds

$$-\infty \leq x_1 \leq u_1$$

$$l_2 \leq x_2 \leq +\infty$$

$$\vdots$$

$$l_i \leq s_i \leq u_i$$

$$l_j \leq x_j \leq u_j$$

Case 2:

- v doesn't satisfy bound on the basic variables s_i, and
- all x_j's that s_i depends on are at their bounds (can't fix).
- Unsatisfiable, the row is the explanation.

Linear Arithmetic

tableau

Tableau

$$s_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n$$

$$\vdots$$

$$\color{red}{s_i} = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot \color{red}{x_j} + \cdots + a_{i,n} \cdot x_n$$

$$\vdots$$

$$s_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n$$

Bounds

$$-\infty \le x_1 \le u_1$$

$$l_2 \le x_2 \le +\infty$$

$$\vdots$$

$$l_i \le s_i \le u_i$$

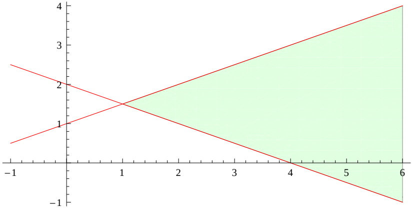
$$l_j \le x_j \le u_j$$

Case 3:

- v doesn't satisfy bound on the basic variables s_i, and
- exists x_j's that s_i depends on, with slack available.
- Pivot, update, and continue.

Linear Arithmetic

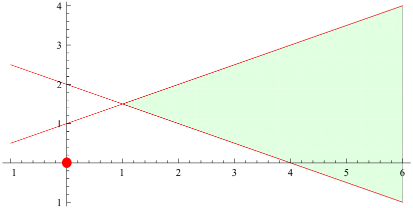
example



$$\llbracket\ 2y - x - 2 \leq 0,\ -2y - x + 4 \leq 0\ \rrbracket$$

Linear Arithmetic

example



$$[\![\; s_1 \leq 2, \; s_2 \leq -4 \;]\!]$$

Tableau

$$s_1 = 2y - x$$
$$s_2 = -2y - x$$

Bounds

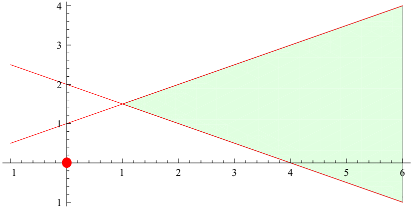
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq \; +\infty$$
$$-\infty \leq s_2 \leq \; +\infty$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 0$$
$$s_1 \mapsto 0$$
$$s_2 \mapsto 0$$

Linear Arithmetic

example



$$\llbracket\, s_1 \leq 2,\ s_2 \leq -4 \,\rrbracket$$

Tableau

$$s_1 = 2y - x$$
$$s_2 = -2y - x$$

Bounds

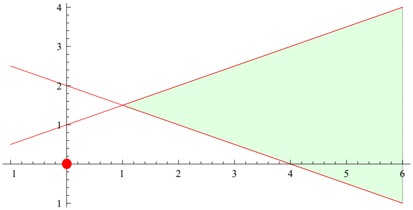
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq +\infty$$
$$-\infty \leq s_2 \leq +\infty$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 0$$
$$s_1 \mapsto 0$$
$$s_2 \mapsto 0$$

Linear Arithmetic

example



$$[\![\ \color{red}{s_1 \leq 2},\ s_2 \leq -4\ ]\!]$$

Tableau

$$s_1 = 2y - x$$
$$s_2 = -2y - x$$

Bounds

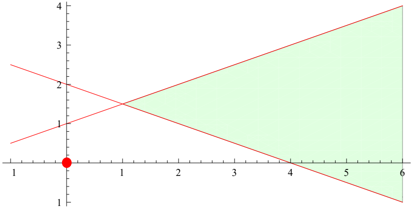
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq +\infty$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 0$$
$$s_1 \mapsto 0$$
$$s_2 \mapsto 0$$

Linear Arithmetic

example



$$[\![\ s_1 \leq 2,\ s_2 \leq -4\ ]\!]$$

Tableau

$$s_1 = 2y - x$$

$$s_2 = -2y - x$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq +\infty$$

Assignment

$$x \mapsto 0$$

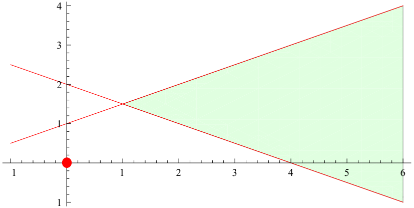
$$y \mapsto 0$$

$$s_1 \mapsto 0$$

$$s_2 \mapsto 0$$

Linear Arithmetic

example



$$[\![\, s_1 \leq 2,\ {\color{red} s_2 \leq -4} \,]\!]$$

Tableau

$$s_1 = 2y - x$$
$$s_2 = -2y - x$$

Bounds

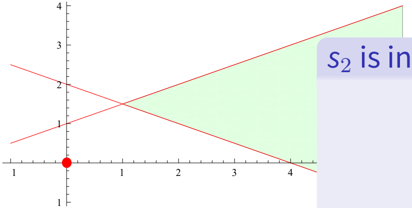
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 0$$
$$s_1 \mapsto 0$$
$$s_2 \mapsto 0$$

Linear Arithmetic

example



$$[\![\, s_1 \leq 2,\ s_2 \leq -4\,]\!]$$

s_2 is inconsistent

Tableau

$$s_1 = 2y - x$$
$$s_2 = -2y - x$$

Bounds

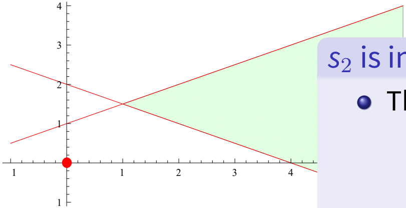
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 0$$
$$s_1 \mapsto 0$$
$$s_2 \mapsto 0$$

Linear Arithmetic

example



$[\![\; s_1 \leq 2, \; s_2 \leq -4 \;]\!]$

s_2 is inconsistent

- There is slack in y.

Tableau

$$s_1 = 2y - x$$
$$s_2 = -2y - x$$

Bounds

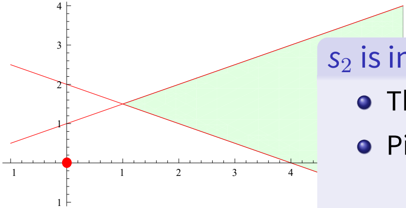
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 0$$
$$s_1 \mapsto 0$$
$$s_2 \mapsto 0$$

Linear Arithmetic

example



$$[\![\, s_1 \leq 2,\ s_2 \leq -4 \,]\!]$$

s_2 is inconsistent

- There is slack in y.
- Pivot s_2 and y.

Tableau

$$s_1 = 2y - x$$
$$s_2 = -2y - x$$

Bounds

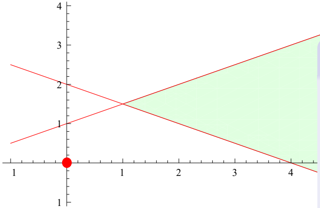
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 0$$
$$s_1 \mapsto 0$$
$$s_2 \mapsto 0$$

Linear Arithmetic

example



$$[\![\, s_1 \leq 2,\ s_2 \leq -4\,]\!]$$

s_2 is inconsistent

- There is slack in y.
- Pivot s_2 and y.

Tableau

$$s_1 = -s_2 - 2x$$

$$y = -\frac{1}{2}s_2 - \frac{1}{2}x$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$

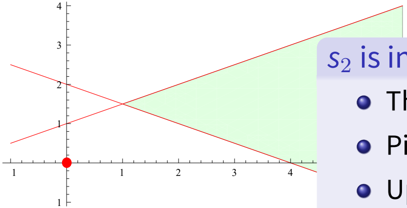
$$y \mapsto 0$$

$$s_1 \mapsto 0$$

$$s_2 \mapsto 0$$

Linear Arithmetic

example



$[\![\ s_1 \leq 2,\ s_2 \leq -4\ ]\!]$

s_2 is inconsistent

- There is slack in y.
- Pivot s_2 and y.
- Update s_2 value.

Tableau

$$s_1 = -s_2 - 2x$$
$$y = -\frac{1}{2}s_2 - \frac{1}{2}x$$

Bounds

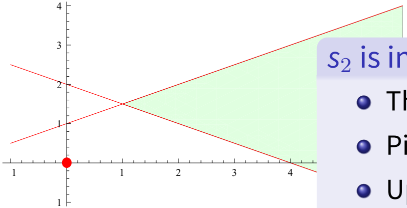
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 0$$
$$s_1 \mapsto 0$$
$$s_2 \mapsto 0$$

Linear Arithmetic

example



$$[\![\, s_1 \leq 2,\ s_2 \leq -4 \,]\!]$$

s_2 is inconsistent

- There is slack in y.
- Pivot s_2 and y.
- Update s_2 value.

Tableau

$$s_1 = -s_2 - 2x$$

$$y = -\frac{1}{2}s_2 - \frac{1}{2}x$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$

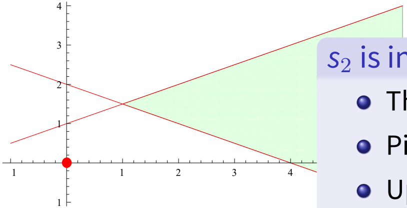
$$y \mapsto 0$$

$$s_1 \mapsto 0$$

$$s_2 \mapsto -4$$

Linear Arithmetic

example



$$[\![\, s_1 \leq 2,\ s_2 \leq -4 \,]\!]$$

s_2 is inconsistent

- There is slack in y.
- Pivot s_2 and y.
- Update s_2 value.
- Update basics.

Tableau

$$s_1 = -s_2 - 2x$$
$$y = -\frac{1}{2}s_2 - \frac{1}{2}x$$

Bounds

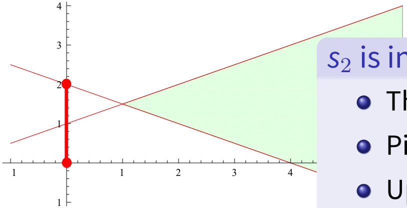
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 0$$
$$s_1 \mapsto 0$$
$$s_2 \mapsto -4$$

Linear Arithmetic

example



$[\![s_1 \leq 2,\ s_2 \leq -4\ ]\!]$

s_2 is inconsistent

- There is slack in y.
- Pivot s_2 and y.
- Update s_2 value.
- Update basics.

Tableau

$$s_1 = -s_2 - 2x$$

$$y = -\frac{1}{2}s_2 - \frac{1}{2}x$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$

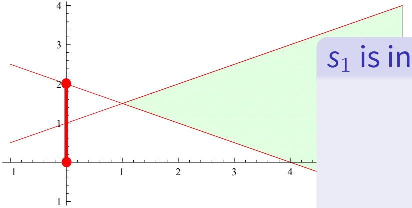
$$y \mapsto 2$$

$$s_1 \mapsto 4$$

$$s_2 \mapsto -4$$

Linear Arithmetic

example



$$[\![\, s_1 \leq 2,\ s_2 \leq -4\ ]\!]$$

s_1 is inconsistent

Tableau

$$s_1 = -s_2 - 2x$$

$$y = -\frac{1}{2}s_2 - \frac{1}{2}x$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$

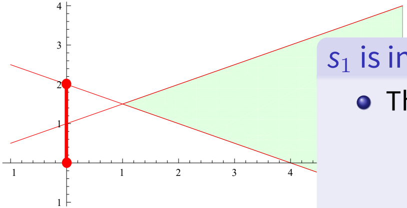
$$y \mapsto 2$$

$$s_1 \mapsto 4$$

$$s_2 \mapsto -4$$

Linear Arithmetic

example



$$[\![\; s_1 \leq 2,\; s_2 \leq -4\;]\!]$$

s_1 is inconsistent

- There is slack in x.

Tableau

$$s_1 = -s_2 - 2x$$

$$y = -\frac{1}{2}s_2 - \frac{1}{2}x$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$

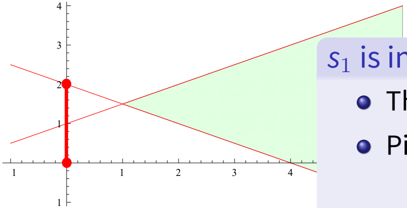
$$y \mapsto 2$$

$$s_1 \mapsto 4$$

$$s_2 \mapsto -4$$

Linear Arithmetic

example



$$[\![\, s_1 \leq 2,\ s_2 \leq -4 \,]\!]$$

s_1 is inconsistent

- There is slack in x.
- Pivot s_1 and x.

Tableau

$$s_1 = -s_2 - 2x$$

$$y = -\frac{1}{2}s_2 - \frac{1}{2}x$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq \; -4$$

Assignment

$$x \mapsto 0$$

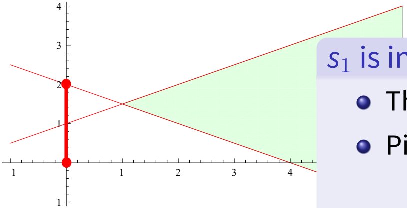
$$y \mapsto 2$$

$$s_1 \mapsto 4$$

$$s_2 \mapsto -4$$

Linear Arithmetic

example



$$[\![\, s_1 \leq 2,\ {\color{red} s_2 \leq -4}$$

s_1 is inconsistent

- There is slack in x.
- Pivot s_1 and x.

Tableau

$$x = -\frac{1}{2}s_1 - \frac{1}{2}s_2$$

$$y = \frac{1}{4}s_1 - \frac{1}{4}s_2$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$

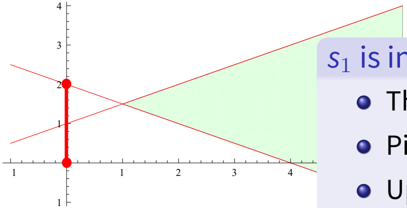
$$y \mapsto 2$$

$$s_1 \mapsto 4$$

$$s_2 \mapsto -4$$

Linear Arithmetic

example



$[\![\ s_1 \leq 2,\ s_2 \leq -4\ ]\!]$

s_1 is inconsistent

- There is slack in x.
- Pivot s_1 and x.
- Update s_1 value.

Tableau

$$x = -\frac{1}{2}s_1 - \frac{1}{2}s_2$$

$$y = \frac{1}{4}s_1 - \frac{1}{4}s_2$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$

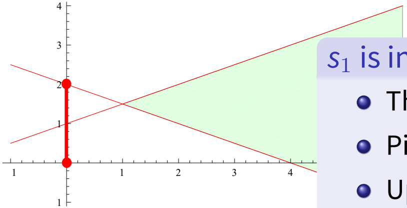
$$y \mapsto 2$$

$$s_1 \mapsto 4$$

$$s_2 \mapsto -4$$

Linear Arithmetic

example



$$[\![\ s_1 \leq 2,\ s_2 \leq -4\ ]\!]$$

s_1 is inconsistent

- There is slack in x.
- Pivot s_1 and x.
- Update s_1 value.

Tableau

$$x = -\frac{1}{2}s_1 - \frac{1}{2}s_2$$

$$y = \frac{1}{4}s_1 - \frac{1}{4}s_2$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$

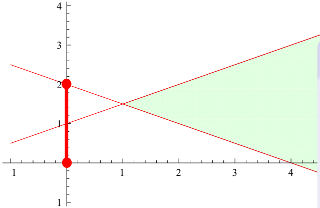
$$y \mapsto 2$$

$$s_1 \mapsto 2$$

$$s_2 \mapsto -4$$

Linear Arithmetic

example



$$[\![\, s_1 \leq 2, \; {\color{red} s_2 \leq -4} \,]\!]$$

s_1 is inconsistent

- There is slack in x.
- Pivot s_1 and x.
- Update s_1 value.
- Update basics.

Tableau

$$x = -\frac{1}{2}s_1 - \frac{1}{2}s_2$$

$$y = \frac{1}{4}s_1 - \frac{1}{4}s_2$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$

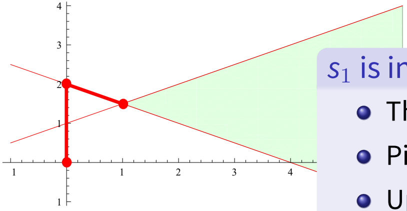
$$y \mapsto 2$$

$$s_1 \mapsto 2$$

$$s_2 \mapsto -4$$

Linear Arithmetic

example



$[\![\, s_1 \leq 2,\; s_2 \leq -4 \,]\!]$

s_1 is inconsistent

- There is slack in x.
- Pivot s_1 and x.
- Update s_1 value.
- Update basics.

Tableau

$$x = -\frac{1}{2}s_1 - \frac{1}{2}s_2$$

$$y = \frac{1}{4}s_1 - \frac{1}{4}s_2$$

Bounds

$$-\infty \leq x \leq +\infty$$

$$-\infty \leq y \leq +\infty$$

$$-\infty \leq s_1 \leq 2$$

$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 1$$

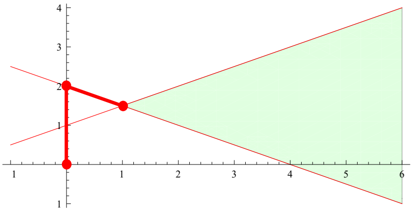
$$y \mapsto \frac{3}{2}$$

$$s_1 \mapsto 2$$

$$s_2 \mapsto -4$$

Linear Arithmetic

example



$$[\![\; s_1 \leq 2, \; s_2 \leq -4 \;]\!]$$

Tableau

$$x = -\frac{1}{2}s_1 - \frac{1}{2}s_2$$
$$y = \frac{1}{4}s_1 - \frac{1}{4}s_2$$

Bounds

$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 1$$
$$y \mapsto \frac{3}{2}$$
$$s_1 \mapsto 2$$
$$s_2 \mapsto -4$$

Linear Arithmetic

integers

- Classic NP-complete problem [Pap81].
- Admits quantifier elimination [Coo72].
- Common approach:
  - Simplex + Branch-And-Bound [DDM06, Gri12, Kin14]
  - Use Simplex as if variables were real.
  - If UNSAT in reals, then UNSAT in integers too.
  - If SAT and solution is integral, then SAT (lucky).
  - If non-integral solution $v(x)$, then refine:
    - Branch-and-Bound lemmas: $(x \leq \lfloor v(x) \rfloor) \vee (x \geq \lceil v(x) \rceil)$.
    - Cutting plane lemmas: new implied inequality refuting v.
  - Additionally solve integer equalities.

Linear Arithmetic

integers

- Classic NP-complete problem [Pap81].
- Admits quantifier elimination [Coo72].
- Common approach:
  - Simplex + Branch-And-Bound [DDM06, Gri12, Kin14]
  - Use Simplex as if variables were real.
  - If UNSAT in reals, then UNSAT in integers too.
  - If SAT and solution is integral, then SAT (lucky).
  - If non-integral solution $v(x)$, then refine:
    - Branch-and-Bound lemmas: $(x \leq \lfloor v(x) \rfloor) \vee (x \geq \lceil v(x) \rceil)$.
    - Cutting plane lemmas: new implied inequality refuting v.
  - Additionally solve integer equalities.
  - Sad, but not guaranteed to terminate.

Linear Arithmetic

integers

- Classic NP-complete problem [Pap81].
- Admits quantifier elimination [Coo72].
- Common approach:
  - Simplex + Branch-And-Bound [DDM06, Gri12, Kin14]
  - Use Simplex as if variables were real.
  - If UNSAT in reals, then UNSAT in integers too.
  - If SAT and solution is integral, then SAT (lucky).
  - If non-integral solution $v(x)$, then refine:
    - Branch-and-Bound lemmas: $(x \leq \lfloor v(x) \rfloor) \vee (x \geq \lceil v(x) \rceil)$.
    - Cutting plane lemmas: new implied inequality refuting v.
  - Additionally solve integer equalities.
  - Sad, but not guaranteed to terminate.
- Alternatives [JdM13, BSW15] not yet mature.

Arrays

the theory

$$\forall a, i, e \ : \ \text{store}(a, i, e)[i] = e$$

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

$$\forall a, b \ : \ a \neq b \Rightarrow \exists i : a[i] \neq b[i]$$

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.

Bit-Vectors

the theory

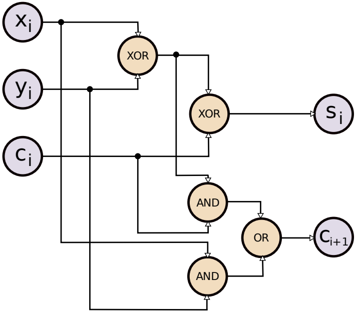
Common approach:

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

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

Bit-Vectors

bit-blasting

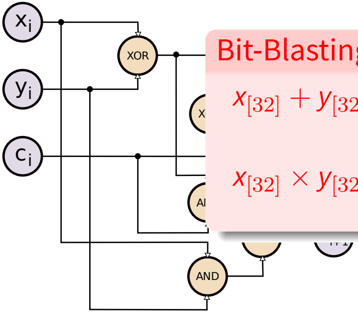


Translation to CNF

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

Bit-Vectors

bit-blasting



Translation to CNF

Bit-Blasting Addition/Multiplication

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

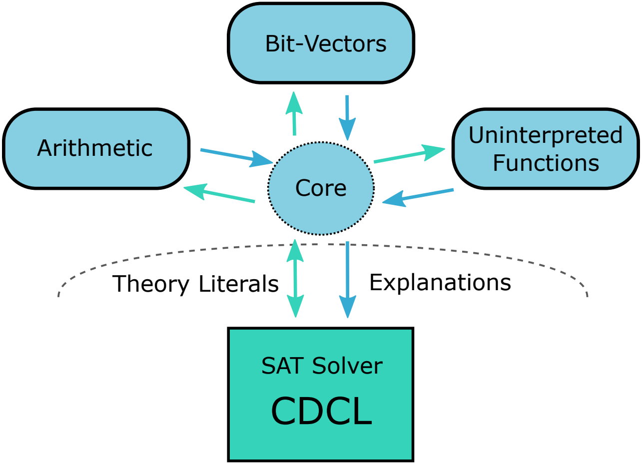
$x_{[32]} \times y_{[32]}$ 10016 new clauses, 3008 new variables

- **17 new clauses**
- **5 new variables**

Outline

DPLL(T)

architecture



DPLL(T)

pros and cons

Good

- Simple interface
- Only conjunctions of constraints

DPLL(T)

pros and cons

Good

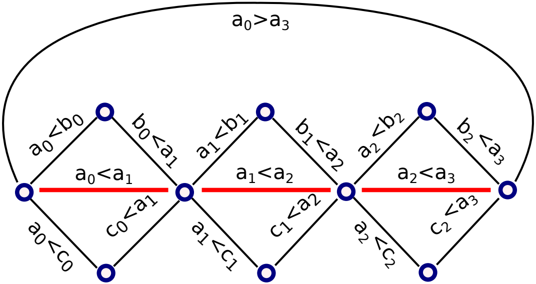
- Simple interface
- Only conjunctions of constraints

What can be improved?

- Simple interface can be restrictive
- Arbitrary conjunctions of constraints

DPLL(T)

simple interface issues



Example (Diamonds)

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

DPPL(T)

simple interface issues

How to fix this?

- Extensions of DPLL(T) can add new literals [BNOT06].
- Magic needed to discover these literals [BDdM08, HAMM14].
- More pragmatic approach would be desirable.

Rethink the Architecture!

- Why is SAT solver special?
- Why the restriction on the interface?
- Let's dig deeper into how SAT solvers work.

Boolean Satisfiability

history

$$\boxed{x_n \vee \cdots \vee x_1 \vee \overline{y_m} \vee \cdots \vee \overline{y_1}}$$

- **Resolution procedure** by Davis, Putnam [DP60]
- **Search procedure** by Davis, Logemann, Loveland [DLL62]

Resolution (DP)

- Find a proof
- Saturation
- Exponential

Search (DLL)

- Find a model
- Search and backtracking
- Exponential

Boolean Satisfiability

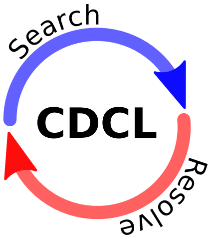
modern

Marques-Silva, Sakallah
[SS97] GRASP: A new search algorithm for satisfiabiliy

Moskewicz, Madigan, Zhao, Zhang, Malik
[MMZ+01] CHAFF: Engineering an efficient SAT solver

Conflict-Directed Clause Learning

- Use the search to guide resolution
- Use resolution to guide the search



Boolean Satisfiability

cdcl mechanics

Model Construction

Build partial model by assigning variables to values

$$[\![\ldots, x, \ldots, \bar{y}, \ldots, z, \ldots]\!] \quad .$$

Unit Reasoning

Reason about unit constraints

$$(\bar{x} \vee y \vee \bar{z} \vee w) \quad .$$

Explain Conflicts

Explain conflicts using clausal reasons

$$(\bar{x} \vee y \vee \bar{z}) \quad .$$

Linear Real Arithmetic

Linear Arithmetic

$$a_1 x_1 + \cdots + a_n x_n \geq b \qquad a_1 x_1 + \cdots + a_n x_n = b$$

DPLL(T): Simplex

A model builder for a conjunction of linear constraints.

- Search for a model
- Escape conflicts through pivoting
- Built for the DPLL(T) framework

[DDM06] A fast linear-arithmetic solver for DPLL(T)

Linear Real Arithmetic

Linear Arithmetic

$$a_1x_1 + \cdots + a_nx_n \geq b \qquad a_1x_1 + \cdots + a_nx_n = b$$

Fourier-Motzkin Resolution

$$\dfrac{\dfrac{2x + 3y - z \geq -1}{6x + 9y - 3z \geq -3} \qquad \dfrac{-3x - 2y + 4z \geq 2}{-6x - 4y + 8z \geq 4}}{5y + 5z \geq 1}$$

- Feels like Boolean resolution (elimination).
- Behaves like Boolean resolution (exponential).

Linear Real Arithmetic

Model Construction

Build partial model by assigning variables to values

$$[\![\ldots, C_1, C_2, \ldots, x \mapsto 1/2, \ldots, y \mapsto 1/2, \ldots, z \mapsto -1, \ldots]\!] \ .$$

Unit Reasoning

Reason about unit constraints

$$C_1 \equiv (x + y + z + w \geq 0) \qquad C_2 \equiv (x + y + z - w > 0) \ .$$

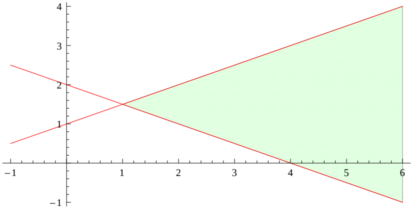
Explain Conflicts

Explain conflicts using valid clausal reasons

$$(\overline{C_1} \vee \overline{C_2} \vee x + y + z > 0) \ .$$

Linear Real Arithmetic

example

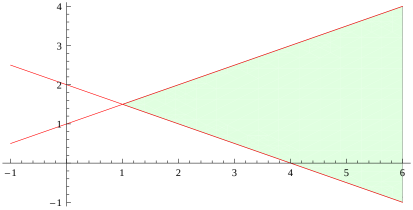


$$\overbrace{2y - x - 2 < 0}^{c_1} \quad \wedge \quad \overbrace{-2y - x + 4 < 0}^{c_2}$$

$$\llbracket\rrbracket$$

Linear Real Arithmetic

example

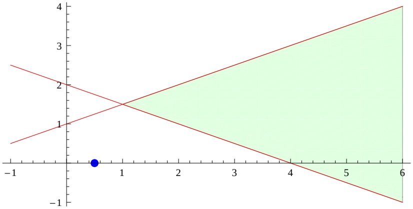


$$\overbrace{2y - x - 2 < 0}^{c_1} \wedge \overbrace{-2y - x + 4 < 0}^{c_2}$$

$$[\![c_1, c_2]\!]$$

Linear Real Arithmetic

example

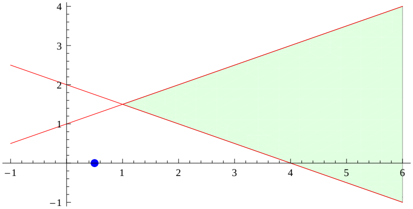


$$\overbrace{2y - x - 2 < 0}^{C_1} \;\wedge\; \overbrace{-2y - x + 4 < 0}^{C_2}$$

$$[\![C_1, C_2, x \mapsto 0.5]\!]$$

Linear Real Arithmetic

example



Unit Constraint Reasoning

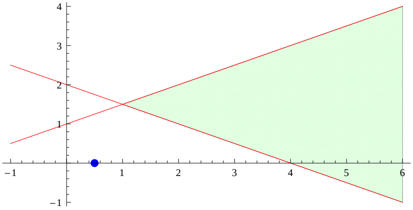
$$2y - x - 2 < 0 \Rightarrow (y < 1.25)$$

$$-2y - x + 4 < 0 \Rightarrow (y > 1.75)$$

$$[\![C_1, C_2, x \mapsto 0.5]\!]$$

Linear Real Arithmetic

example



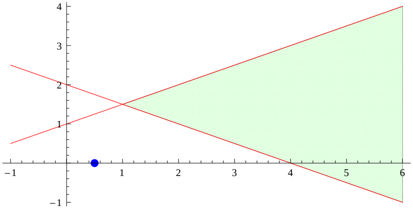
$$\overbrace{2y - x - 2 < 0}^{C_1} \ \wedge \ \overbrace{-2y - x + 4 < 0}^{C_2}$$

$$[\![C_1, C_2, {\color{red} x \mapsto 0.5}]\!]$$

Explanation $C_1 \wedge C_2 \Rightarrow x \neq 0.5$

Linear Real Arithmetic

example



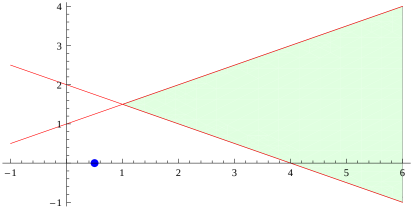
$$\overbrace{2y - x - 2 < 0}^{C_1} \;\wedge\; \overbrace{-2y - x + 4 < 0}^{C_2}$$

$$[\![C_1, C_2, x \mapsto 0.5]\!]$$

Explanation $C_1 \wedge C_2 \Rightarrow$

Linear Real Arithmetic

example



Fourier-Motzkin

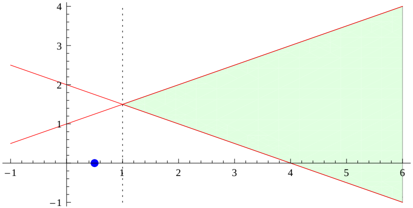
$$\frac{2y - x - 2 < 0 \qquad -2y - x + 4 < 0}{-2x + 2 < 0}$$

$$[\![C_1, C_2, x \mapsto 0.5]\!]$$

Explanation $C_1 \wedge C_2 \Rightarrow$

Linear Real Arithmetic

example



Fourier-Motzkin

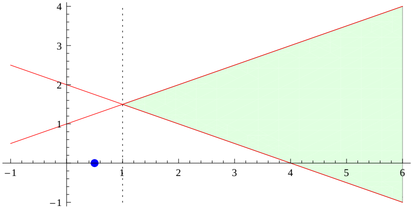
$$\frac{2y - x - 2 < 0 \qquad -2y - x + 4 < 0}{-2x + 2 < 0}$$

$$[\![C_1, C_2, x \mapsto 0.5]\!]$$

Explanation $C_1 \wedge C_2 \Rightarrow x > 1$

Linear Real Arithmetic

example



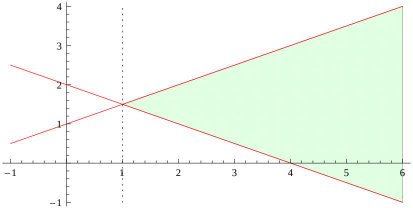
$$\overbrace{2y - x - 2 < 0}^{C_1} \;\wedge\; \overbrace{-2y - x + 4 < 0}^{C_2}$$

$$[\![C_1, C_2, x \mapsto 0.5]\!]$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee (x > 1)$

Linear Real Arithmetic

example



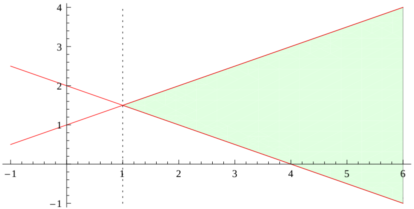
$$\overbrace{2y - x - 2 < 0}^{C_1} \;\wedge\; \overbrace{-2y - x + 4 < 0}^{C_2}$$

$$[\![C_1, C_2]\!]$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee (x > 1)$

Linear Real Arithmetic

example



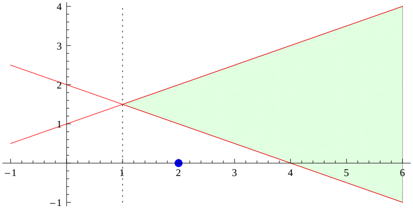
$$\overbrace{2y - x - 2 < 0}^{C_1} \;\wedge\; \overbrace{-2y - x + 4 < 0}^{C_2}$$

$$[\![C_1, C_2, x > 1]\!]$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee (x > 1)$

Linear Real Arithmetic

example



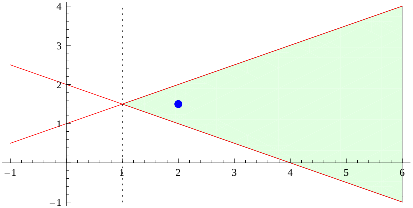
$$\overbrace{2y - x - 2 < 0}^{C_1} \;\wedge\; \overbrace{-2y - x + 4 < 0}^{C_2}$$

$$[\![C_1, C_2, x > 1, {\color{red} x \mapsto 2}]\!]$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee (x > 1)$

Linear Real Arithmetic

example



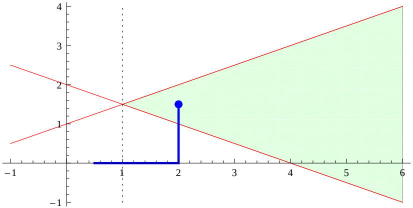
$$\overbrace{2y - x - 2 < 0}^{C_1} \;\wedge\; \overbrace{-2y - x + 4 < 0}^{C_2}$$

$$\llbracket C_1, C_2, x > 1, {\color{red} x \mapsto 2, y \mapsto 1.5} \rrbracket$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee (x > 1)$

Linear Real Arithmetic

example



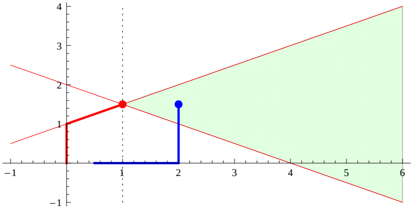
$$\overbrace{2y - x - 2 < 0}^{C_1} \;\wedge\; \overbrace{-2y - x + 4 < 0}^{C_2}$$

$$[\![C_1, C_2, x > 1, {\color{red} x \mapsto 2, y \mapsto 1.5}]\!]$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee (x > 1)$

Linear Real Arithmetic

example



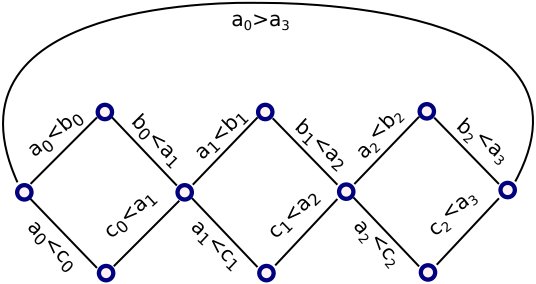
$$\overbrace{2y - x - 2 < 0}^{C_1} \wedge \overbrace{-2y - x + 4 < 0}^{C_2}$$

$$[\![C_1, C_2, x > 1, {\color{red} x \mapsto 2, y \mapsto 1.5}]\!]$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee (x > 1)$

Linear Real Arithmetic

comparison to dpll(t)

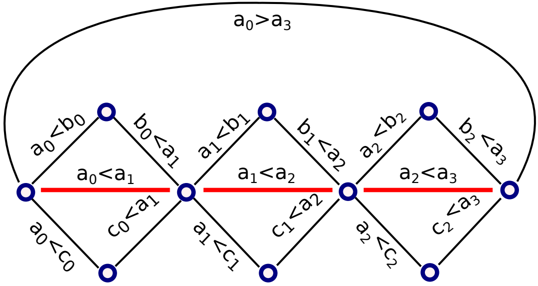


Example (Diamonds)

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

Linear Real Arithmetic

comparison to dpll(t)



Example (Diamonds)

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

Linear Real Arithmetic

comparison to dpll(t)

	mcsat		cvc4		z3		mathsat5		yices	
set	solved	time (s)	solved	time (s)	solved	time (s)	solved	time (s)	solved	time (s)
clocksynchro (36)	**36**	**123.11**	36	1166.55	36	1828.74	36	1732.59	36	1093.80
DTPScheduling (91)	**91**	**31.33**	91	72.92	91	100.55	89	1980.96	91	926.22
miplib (42)	8	97.16	**27**	**3359.40**	23	3307.92	19	5447.46	23	466.44
sal (107)	107	12.68	107	13.46	107	6.37	107	7.99	**107**	**2.45**
sc (144)	144	1655.06	144	1389.72	144	954.42	144	880.27	**144**	**401.64**
spiderbenchmarks (42)	42	2.38	42	2.47	42	1.66	42	1.22	**42**	**0.44**
TM (25)	25	1125.21	25	82.12	**25**	**51.64**	25	1142.98	25	55.32
ttastartup (72)	70	4443.72	72	1305.93	72	1647.94	72	2607.49	**72**	**1218.68**
uart (73)	73	5244.70	73	1439.89	73	1379.90	73	1481.86	**73**	**679.54**
	596	12735.35	**617**	**8832.46**	613	9279.14	607	15282.82	613	4844.53

Linear Real Arithmetic

comparison to dpll(t)

DPLL(T) Simplex (CVC4)

```
Total Physical Source Lines of Code (SLOC)                = 22,597
Development Effort Estimate, Person-Years (Person-Months) = 5.28 (63.38)
 (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months)                         = 1.01 (12.10)
 (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule)  = 5.24
Total Estimated Cost to Develop                           = $ 713,502
 (average salary = $56,286/year, overhead = 2.40).
```

MCSAT Fourier-Motzkin (CVC4)

```
Total Physical Source Lines of Code (SLOC)                = 1,966
Development Effort Estimate, Person-Years (Person-Months) = 0.41 (4.88)
 (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months)                         = 0.38 (4.57)
 (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule)  = 1.07
Total Estimated Cost to Develop                           = $ 54,942
 (average salary = $56,286/year, overhead = 2.40).
```

Generated using David A. Wheeler's SLOCCount.

Non-Linear Arithmetic

$$f(\vec{y}, x) = a_m \cdot x^{d_m} + a_{m-1} \cdot x^{d_{m-1}} + \cdots + a_1 \cdot x^{d_1} + a_0$$

f is in $\mathbb{Z}[\vec{y}, x]$, a_i are in $\mathbb{Z}[\vec{y}]$

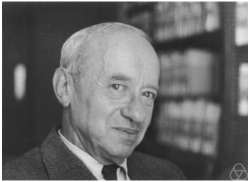
Examples

$$f(x, y) = (x^2 - 1)y^2 + (x + 1)y - 1 \in \mathbb{Z}[x, y]$$
$$g(x) = 16x^3 - 8x^2 + x + 16 \in \mathbb{Z}[x]$$

Polynomial Constraints

$$f(x, y) > 0 \wedge g(x) < 0$$

Non-Linear Arithmetic
history

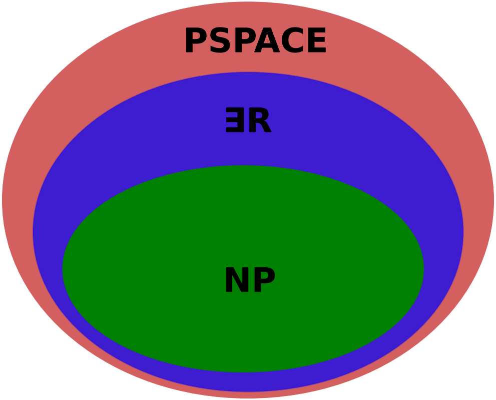


Tarski [Tar48]
Quantifier elimination
Decidable, non-elementary

Collins [Col75]
Cylindrical Algebraic Decomposition
Doubly-exponential

Non-Linear Real Arithmetic

complexity



Canny [Can88], Grigor'ev [Gri88]

Non-Linear Real Arithmetic

cylindrical algebraic decomposition

$$p_1 > 0 \vee (p_2 = 0 \wedge p_3 < 0) \qquad p_1, p_2, p_3 \in \mathbb{Z}[x_1, \ldots, x_n]$$

Projection (Saturation)

Project polynomials using a projection P

$$\{p_1, p_2, p_3\} \mapsto \{p_1, p_2, p_3, p_4, \ldots, p_n\} \quad .$$

Lifting (Model construction)

For each variable x_k

1. Isolate roots of $p_i(\alpha, x_k)$.
2. Choose a cell C and assign $x_k \mapsto \alpha_k \in C$, continue.
3. If no more cells, backtrack.

Non-Linear Real Arithmetic

Model Construction

Build partial model by assigning variables to values

$$[\![\ldots, C_1, C_2, \ldots, x \mapsto \sqrt{2}/2, \ldots]\!] \ .$$

Unit Reasoning

Reason about unit constraints

$$C_1 \equiv (x^2 + y^2 < 1) \qquad\qquad C_2 \equiv (xy > 1) \ .$$

Explain Conflicts

Explain conflicts using valid clausal reasons

$$(\overline{C_1} \vee \overline{C_2} \vee x \leq 0 \vee x \geq 1) \ .$$

Non-Linear Real Arithmetic

Unit Reasoning

Reason about unit constraints

$$C_1 \equiv (x^2 + y^2 < 1) \qquad\qquad C_2 \equiv (xy > 1) \ .$$

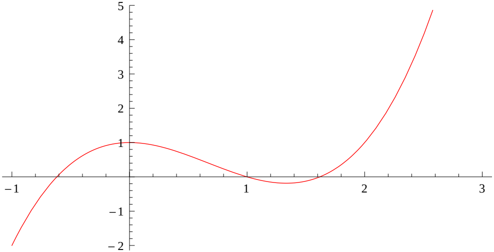
Non-Linear Real Arithmetic

unit constraints

$$x^3 - 2x^2 + 1 > 0$$

Non-Linear Real Arithmetic

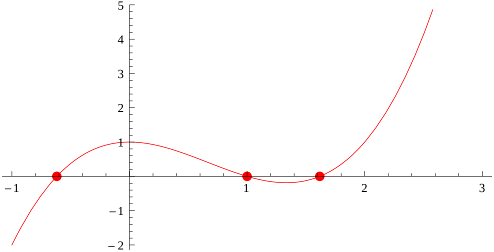
unit constraints



$$x^3 - 2x^2 + 1 > 0$$

Non-Linear Real Arithmetic

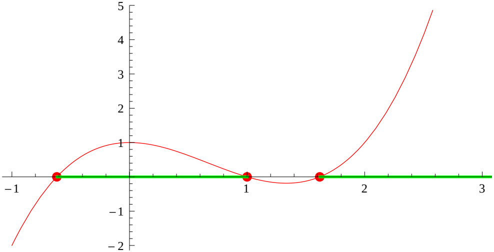
unit constraints



$$x^3 - 2x^2 + 1 > 0$$

Non-Linear Real Arithmetic

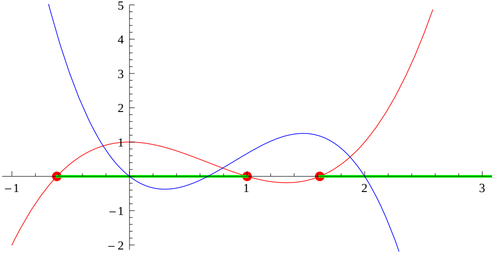
unit constraints



$$x^3 - 2x^2 + 1 > 0$$

Non-Linear Real Arithmetic

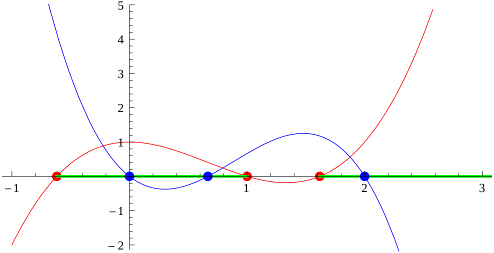
unit constraints



$$x^3 - 2x^2 + 1 > 0 \qquad\qquad -3x^3 + 8x^2 - 4x > 0$$

Non-Linear Real Arithmetic

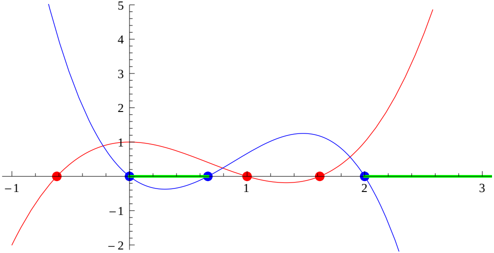
unit constraints



$$x^3 - 2x^2 + 1 > 0 \qquad\qquad - 3x^3 + 8x^2 - 4x > 0$$

Non-Linear Real Arithmetic

unit constraints



$$x^3 - 2x^2 + 1 > 0 \qquad\qquad -3x^3 + 8x^2 - 4x > 0$$

Non-Linear Real Arithmetic

Model Construction

Build partial model by assigning variables to values

$$[\![\ldots, C_1, C_2, \ldots, x \mapsto \sqrt{2}/2, \ldots]\!] \ .$$

Unit Reasoning

Reason about unit constraints

$$C_1 \equiv (x^2 + y^2 < 1) \qquad\qquad C_2 \equiv (xy > 1) \ .$$

Explain Conflicts

Explain conflicts using valid clausal reasons

$$(\overline{C_1} \vee \overline{C_2} \vee x \leq 0 \vee x \geq 1) \ .$$

Non-Linear Real Arithmetic

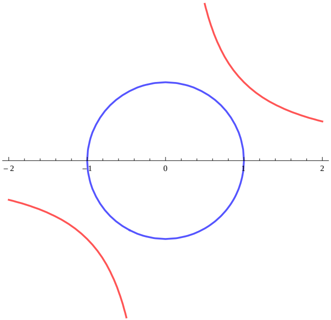
Explain Conflicts

Explain conflicts using valid clausal reasons

$$(\overline{C_1} \vee \overline{C_2} \vee x \leq 0 \vee x \geq 1) \ \ .$$

Non-Linear Real Arithmetic

example

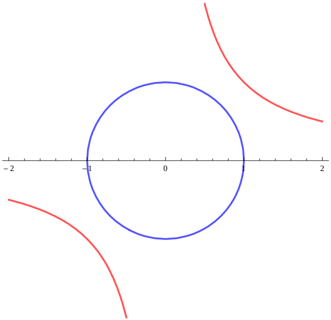


$$\overbrace{x^2 + y^2 < 1}^{c_1} \;\wedge\; \overbrace{xy > 1}^{c_2}$$

$$[\![\ ]\!]$$

Non-Linear Real Arithmetic

example

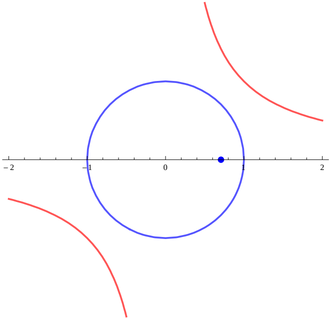


$$\overbrace{x^2 + y^2 < 1}^{c_1} \;\wedge\; \overbrace{xy > 1}^{c_2}$$

$$[\![c_1, c_2]\!]$$

Non-Linear Real Arithmetic

example

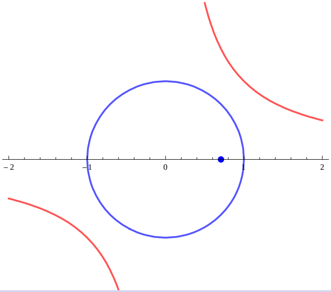


$$\overbrace{x^2 + y^2 < 1}^{c_1} \;\land\; \overbrace{xy > 1}^{c_2}$$

$$[\![c_1, c_2, x \mapsto \sqrt{2}/2]\!]$$

Non-Linear Real Arithmetic

example



Unit Constraint Reasoning

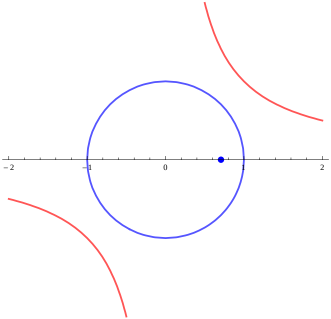
$$x^2 + y^2 < 1 \Rightarrow -\sqrt{3/2} < y < \sqrt{3/2}$$

$$-2y - x + 4 < 0 \Rightarrow y > \sqrt{2}$$

$$[\![C_1, C_2, x \mapsto \sqrt{2}/2]\!]$$

Non-Linear Real Arithmetic

example



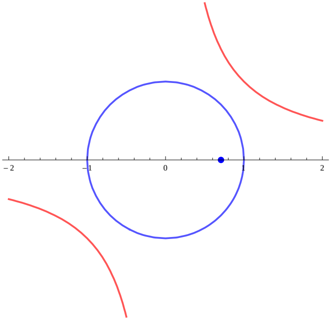
$$\overbrace{x^2 + y^2 < 1}^{C_1} \;\wedge\; \overbrace{xy > 1}^{C_2}$$

$$[\![C_1, C_2, x \mapsto \sqrt{2}/2]\!]$$

Explanation $C_1 \wedge C_2 \Rightarrow x \neq \sqrt{2}/2$

Non-Linear Real Arithmetic

example



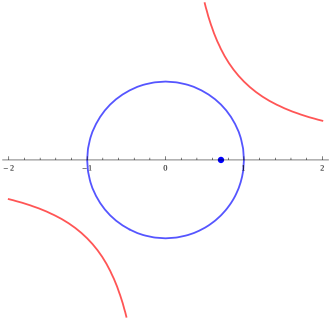
$$\overbrace{x^2 + y^2 < 1}^{C_1} \;\wedge\; \overbrace{xy > 1}^{C_2}$$

$$[\![C_1, C_2, x \mapsto \sqrt{2}/2]\!]$$

Explanation $C_1 \wedge C_2 \Rightarrow$

Non-Linear Real Arithmetic

example



$$\overbrace{x^2 + y^2 < 1}^{c_1} \wedge \overbrace{xy > 1}^{c_2}$$

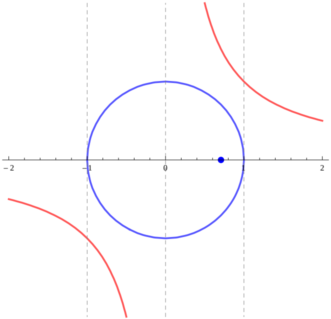
CAD Projection

$$\mathsf{P} = \{x, -4 + 4x^2, 1 - x^2 + x^4\}$$

Explanation $c_1 \wedge c_2 \Rightarrow$

Non-Linear Real Arithmetic

example



$$\overbrace{x^2 + y^2 < 1}^{c_1} \;\wedge\; \overbrace{xy > 1}^{c_2}$$

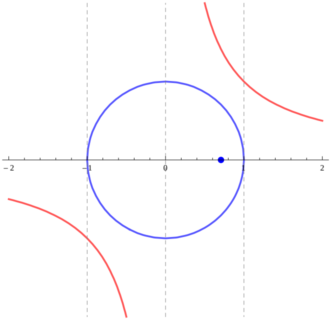
CAD Projection

$$\mathsf{P} = \{x, -4 + 4x^2, 1 - x^2 + x^4\}$$

Explanation $c_1 \wedge c_2 \Rightarrow$

Non-Linear Real Arithmetic

example



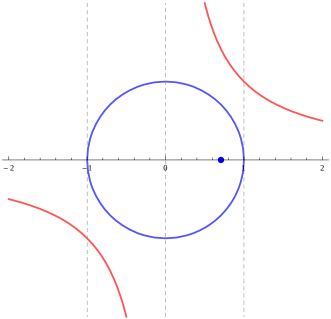
$$\overbrace{x^2 + y^2 < 1}^{C_1} \;\wedge\; \overbrace{xy > 1}^{C_2}$$

$$[\![C_1, C_2, x \mapsto \sqrt{2}/2]\!]$$

Explanation $C_1 \wedge C_2 \Rightarrow x \leq 0 \vee x \geq 1$

Non-Linear Real Arithmetic

example



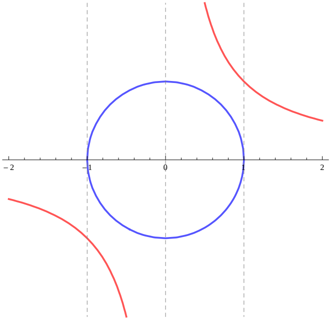
$$\overbrace{x^2 + y^2 < 1}^{C_1} \;\wedge\; \overbrace{xy > 1}^{C_2}$$

$$[\![C_1, C_2, x \mapsto \sqrt{2}/2]\!]$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee x \leq 0 \vee x \geq 1$

Non-Linear Real Arithmetic

example



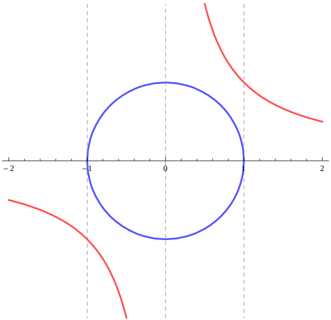
$$\overbrace{x^2 + y^2 < 1}^{C_1} \;\wedge\; \overbrace{xy > 1}^{C_2}$$

$$[\![C_1, C_2]\!]$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee x \leq 0 \vee x \geq 1$

Non-Linear Real Arithmetic

example



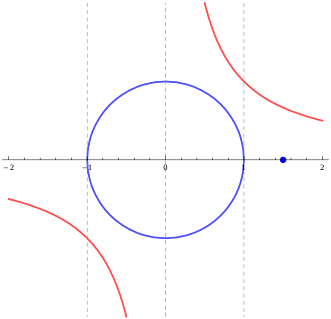
$$\overbrace{x^2 + y^2 < 1}^{C_1} \;\wedge\; \overbrace{xy > 1}^{C_2}$$

$$[\![C_1, C_2, x \geq 1]\!]$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee x \leq 0 \vee x \geq 1$

Non-Linear Real Arithmetic

example



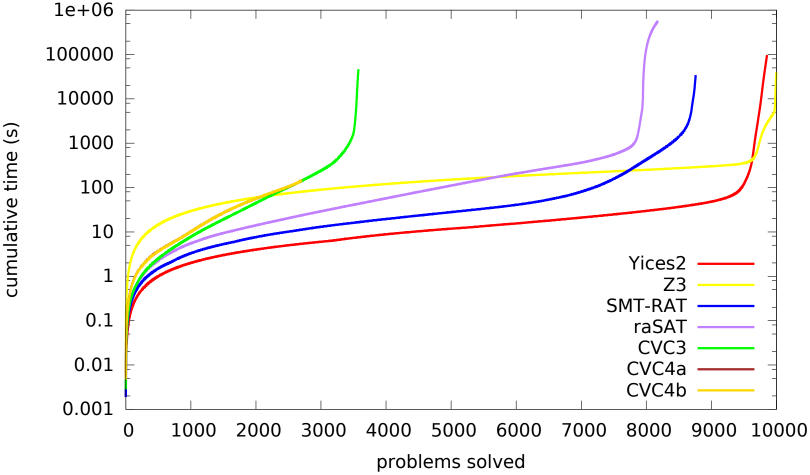
$$\overbrace{x^2 + y^2 < 1}^{C_1} \;\land\; \overbrace{xy > 1}^{C_2}$$

$$[\![C_1, C_2, {\color{red} x \geq 1}, {\color{red} x \mapsto 3/2}]\!]$$

Explanation $\overline{C_1} \vee \overline{C_2} \vee x \leq 0 \vee x \geq 1$

Non-Linear Real Arithmetic

smt-comp 2015



Model-Based Procedures

Linear Real Arithmetic

- Generalizing DPLL to Richer Logics [MKS09]
- Conflict Resolution [KTV09]
- Natural Domain SMT [Cot10]

Linear Integer Arithmetic

- Cutting to the Chase: Solving Linear Integer Arithmetic [JdM13]

Non-Linear Real Arithmetic

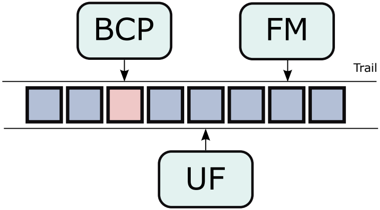
- Solving Non-Linear Arithmetic [JDM12, Jov12]

General Framework

- Model-Constructing Satisfiability Calculus [DMJ13, JBdM13]

MCSAT

simple architecture



Each plugins reasons about their domain:

1. Track when a constraint becomes unit [MMZ$^+$01].
2. Unit constraints imply feasible sets of individual variables.
3. Propagate any constraints/variables whose value is implied.
4. Explain any unit conflicts with clausal explanations.
5. When asked, decide unassigned variable to feasible value.

MCSAT

implementations

- NRA + NIA + UF in Yices2 Link
- QF_NRA in Z3 Link
- QF_UFLRA in CVC4 Link

Outline

Research Challenges

todo

- Extensible and simple SMT solver ala MiniSAT.
- Integer arithmetic: a complete and practical procedure.
- Bit-vectors: other than bit-blasting [HBJ+14, ZWR16].
- Proofs: can we have them without too much trouble.
- Quantifiers: push-button, with model generation.
- MCSAT: arrays, bit-vectors.

Homework

for the practical

Download and install the following solvers

- CVC4 ▸ Link
- MathSAT5 ▸ Link
- Yices2 ▸ Link
- Z3 ▸ Link

Homework

for the practical

Download and install the following solvers

- CVC4 Link
- MathSAT5 Link
- Yices2 Link
- Z3 Link

THE END

References I

[BB09] 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.

[BDdM08] Nikolaj Bjørner, Bruno Dutertre, and Leonardo de Moura.
Accelerating lemma learning using joins-dppl (join).
Proceedings of short papers at LPAR, 8, 2008.

[BDG+14] 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.

[BM07] Aaron R Bradley and Zohar Manna.
The calculus of computation: decision procedures with applications to verification.
Springer Science & Business Media, 2007.

[BNOT06] Clark Barrett, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli.
Splitting on demand in sat modulo theories.
In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 512–526, 2006.

[BSST09] Clark W Barrett, Roberto Sebastiani, Sanjit A Seshia, and Cesare Tinelli.
Satisfiability modulo theories.
Handbook of satisfiability, 185:825–885, 2009.

References II

[BST07] 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.

[BSW15] Martin Bromberger, Thomas Sturm, and Christoph Weidenbach.
Linear integer arithmetic revisited.
In *International Conference on Automated Deduction*, pages 623–637. Springer, 2015.

[Can88] John Canny.
Some algebraic and geometric computations in pspace.
In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 460–467, 1988.

[Col75] George E Collins.
Quantifier elimination for real closed fields by cylindrical algebraic decomposition.
In *Automata Theory and Formal Languages*, pages 134–183, 1975.

[Coo72] David C Cooper.
Theorem proving in arithmetic without multiplication.
Machine Intelligence, 7(91-99):300, 1972.

References III

[Cot10] Scott Cotton.
Natural domain SMT: A preliminary assessment.
Formal Modeling and Analysis of Timed Systems, pages 77–91, 2010.

[DDM06] Bruno Dutertre and Leonardo De Moura.
A fast linear-arithmetic solver for DPLL(T).
In *International Conference on Computer Aided Verification*, pages 81–94. Springer, 2006.

[DLL62] Martin Davis, George Logemann, and Donald Loveland.
A machine program for theorem-proving.
Communications of the ACM, 5(7):394–397, 1962.

[DMB07] Leonardo De Moura and Nikolaj Bjørner.
Efficient e-matching for smt solvers.
In *International Conference on Automated Deduction*, pages 183–198. Springer, 2007.

[DMB09] Leonardo De Moura and Nikolaj Bjørner.
Generalized, efficient array decision procedures.
In *Formal Methods in Computer-Aided Design*, pages 45–52. IEEE, 2009.

[DMB11] Leonardo De Moura and Nikolaj Bjørner.
Satisfiability modulo theories: introduction and applications.
Communications of the ACM, 54(9):69–77, 2011.

References IV

[DMJ13] Leonardo De Moura and Dejan Jovanović.
A model-constructing satisfiability calculus.
In *Verification, Model Checking, and Abstract Interpretation*, pages 1–12, 2013.

[DMR02] Leonardo De Moura and Harald Rueß.
Lemmas on demand for satisfiability solvers.
2002.

[DNS05] 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.

[DP60] Martin Davis and Hilary Putnam.
A computing procedure for quantification theory.
Journal of the ACM (JACM), 7(3):201–215, 1960.

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

[GKC13] 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.

[Gri88] D Yu Grigor'ev.
Complexity of deciding tarski algebra.
Journal of symbolic Computation, 5(1):65–108, 1988.

References V

[Gri12] Alberto Griggio.
A practical approach to satisfiability modulo linear integer arithmetic.
Journal on Satisfiability, Boolean Modeling and Computation, 8:1–27, 2012.

[HAMM14] Julien Henry, Mihail Asavoae, David Monniaux, and Claire Maïza.
How to compute worst-case execution time by optimization modulo theory and a clever encoding of program semantics.
In *Proceedings of the 2014 SIGPLAN/SIGBED conference on Languages, compilers and tools for embedded systems*, pages 43–52. ACM, 2014.

[HBJ+14] 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.

[JBdM13] Dejan Jovanović, Clark Barrett, and Leonardo de Moura.
The design and implementation of the model constructing satisfiability calculus.
Formal Methods in Computer-Aided Design, 2013.

[JDM12] Dejan Jovanović and Leonardo De Moura.
Solving non-linear arithmetic.
In *International Joint Conference on Automated Reasoning*, pages 339–354. Springer, 2012.

References VI

[JdM13] Dejan Jovanović and Leonardo de Moura.
Cutting to the chase.
Journal of automated reasoning, 51(1):79–108, 2013.

[Jov12] Dejan Jovanović.
SMT Beyond DPLL(T): A New Approach to Theory Solvers and Theory Combination.
PhD thesis, Courant Institute of Mathematical Sciences New York, 2012.

[KGG+09] 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.

[Kin14] Tim King.
Effective Algorithms for the Satisfiability of Quantifier-Free Formulas Over Linear Real and Integer Arithmetic.
PhD thesis, Courant Institute of Mathematical Sciences New York, 2014.

[KS08] Daniel Kroening and Ofer Strichman.
Decision procedures: an algorithmic point of view.
Springer Science & Business Media, 2008.

[KTV09] Konstantin Korovin, Nestan Tsiskaridze, and Andrei Voronkov.
Conflict resolution.
In *Principles and Practice of Constraint Programming*, pages 509–523. 2009.

References VII

[LRT+14] 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.

[McC93] John McCarthy.
Towards a mathematical science of computation.
In *Program Verification*, pages 35–56. Springer, 1993.

[MKS09] Kenneth L McMillan, Andreas Kuehlmann, and Mooly Sagiv.
Generalizing DPLL to richer logics.
In *Computer Aided Verification*, pages 462–476, 2009.

[MMZ+01] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik.
Chaff: Engineering an efficient sat solver.
In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.

[NM12] Anthony Narkawicz and César A Munoz.
Formal verification of conflict detection algorithms for arbitrary trajectories.
Reliable Computing, 17(2):209–237, 2012.

[NO07] Robert Nieuwenhuis and Albert Oliveras.
Fast congruence closure and extensions.
Information and Computation, 205(4):557–580, 2007.

References VIII

[NOT05] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli.

In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 36–50. Springer, 2005.

[Pap81] Christos H Papadimitriou.

Journal of the ACM (JACM), 28(4):765–768, 1981.

[RTG+13] Andrew Reynolds, Cesare Tinelli, Amit Goel, Sava Krstić, Morgan Deters, and Clark Barrett.

In *International Conference on Automated Deduction*, pages 377–391. Springer, 2013.

[SS97] João P Marques Silva and Karem A Sakallah.

In *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, pages 220–227. IEEE Computer Society, 1997.

[Tar48] Alfred Tarski.

1948.

References IX

[ZWR14] Aleksandar Zeljić, Christoph M Wintersteiger, and Philipp Rümmer.

In *International Joint Conference on Automated Reasoning*, pages 344–359. Springer, 2014.

[ZWR16] Aleksandar Zeljić, Christoph M Wintersteiger, and Philipp Rümmer.

In *International Conference on Theory and Applications of Satisfiability Testing*, pages 249–266. Springer, 2016.