FOUNDATIONS OF SATISFIABILITY MODULO THEORIES

SC^2 Summer School

Cesare Tinelli

July 31, 2017

The University of Iowa

Many thanks to

- Clark Barrett
- Dejan Jovanovic
- Albert Oliveras

for contributing some of the material used in these slides.

Disclamer: The literature on SMT and its applications is vast. The bibliographic references provided here are just a sample. Apologies to all authors whose work is not cited.

INTRODUCTION

Historically, automated reasoning $\equiv$ uniform proof-search procedures for First Order Logic

Historically, automated reasoning $\equiv$ uniform proof-search procedures for First Order Logic

Limited success: is FOL the best compromise between expressivity and efficiency?

Historically, automated reasoning $\equiv$ uniform proof-search procedures for First Order Logic

Limited success: is FOL the best compromise between expressivity and efficiency?

Most recently R&D has focused on:

- addressing mostly (expressive enough) decidable fragments of a certain logic
- incorporating domain-specific reasoning, e.g on:
  - arithmetic reasoning
  - equality
  - data structures (arrays, lists, stacks, ...)

Examples of this trend:

SAT: propositional formalization, Boolean reasoning

- \+ high degree of efficiency
- – expressive (all NP-complete problems) but involved encodings

SMT: first-order formalization, Boolean + domain-specific reasoning

- \+ improves expressivity and scalability
- – some (but acceptable) loss of efficiency

Examples of this trend:

SAT: propositional formalization, Boolean reasoning

- $+$ high degree of efficiency
- $-$ expressive (all NP-complete problems) but involved encodings

SMT: first-order formalization, Boolean + domain-specific reasoning

- $+$ improves expressivity and scalability
- $-$ some (but acceptable) loss of efficiency

These lectures: an overview of SMT and its formal foundations

Some problems are more naturally expressed in logics other than propositional logic, e.g:

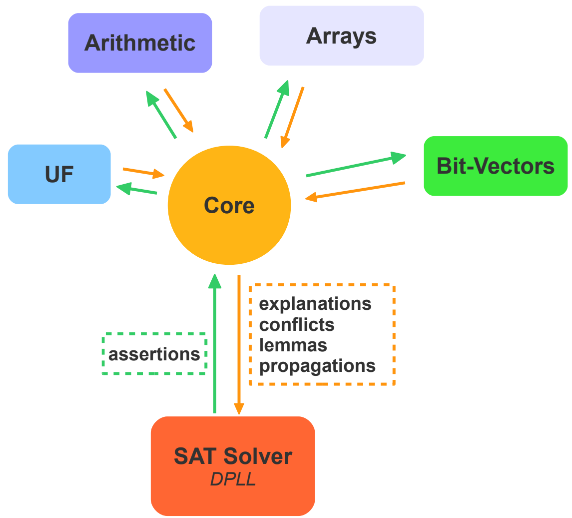
- Software verification needs reasoning about equality, arithmetic, data structures, ...

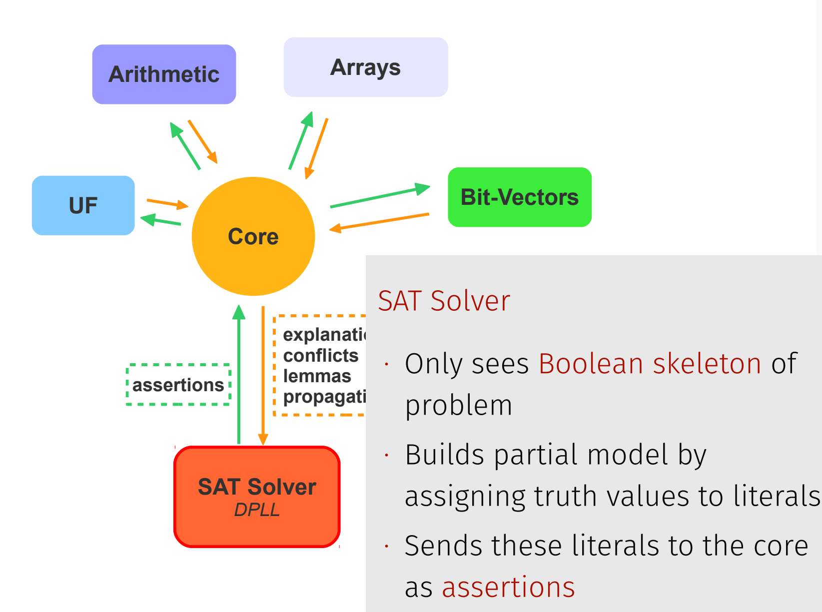
SMT is about deciding the satisfiability of a (usually quantifier-free) FOL formula with respect to some *background theory*

- Example (Equality with Uninterpreted Functions):

$$g(a) = c \quad \wedge \quad (f(g(a)) \neq f(c) \ \vee \ g(a) = d) \quad \wedge \quad c \neq d$$

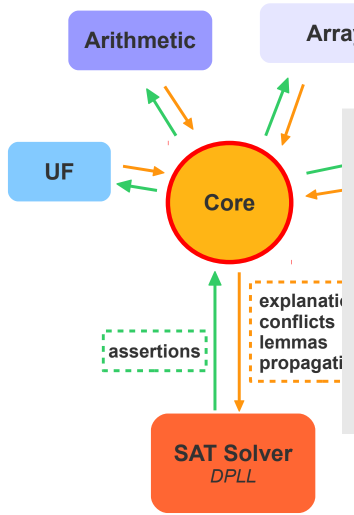
Wide range of applications: Extended Static Checking [FLL+02], Predicate abstraction [LNO06], Model checking [AMP06, HT08], Scheduling [BNO+08b], Test generation [TdH08], ...


Arithmetic
Arrays
UF
Core
Bit-Vectors
assertions
explanations
conflicts
lemmas
propagations
SAT Solver
DPLL



SAT Solver

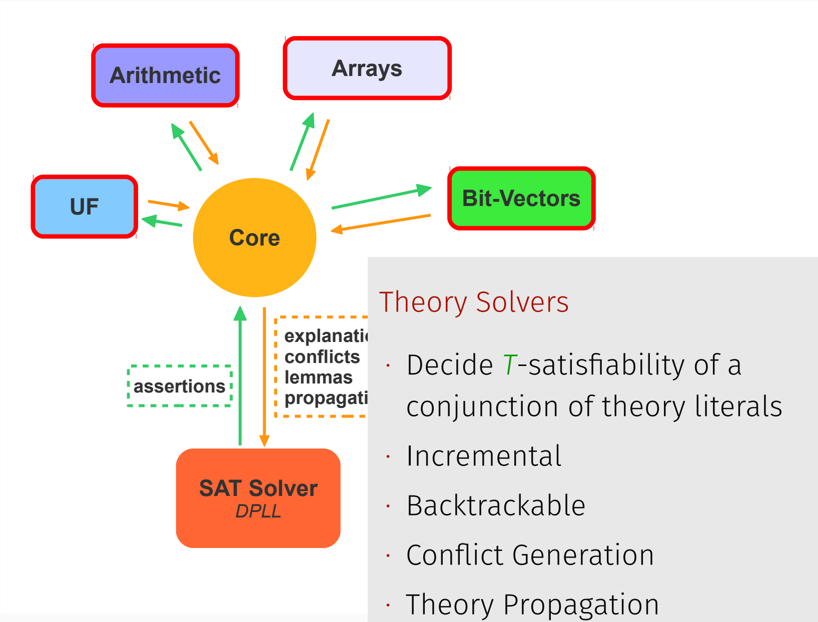
- Only sees Boolean skeleton of problem
- Builds partial model by assigning truth values to literals
- Sends these literals to the core as assertions



Core

- Sends each assertion to the appropriate theory
- Sends deduced literals to other theories/SAT solver
- Handles theory combination

SMT SOLVERS



Theory Solvers

- Decide T-satisfiability of a conjunction of theory literals
- Incremental
- Backtrackable
- Conflict Generation
- Theory Propagation

THEORIES

Equality ($=$) with Uninterpreted Functions [NO80, BD94, NO07a]

Typically used to abstract unsupported constructs, e.g.:

- non-linear multiplication in arithmetic
- ALUs in circuits

Equality ($=$) with Uninterpreted Functions [NO80, BD94, NO07a]

Typically used to abstract unsupported constructs, e.g.:

- non-linear multiplication in arithmetic
- ALUs in circuits

Example: The formula

$$a * (|b| + c) = d \quad \wedge \quad b * (|a| + c) \neq d \quad \wedge \quad a = b$$

is unsatisfiable, but no arithmetic reasoning is needed

Equality ($=$) with Uninterpreted Functions [NO80, BD94, NO07a]

Typically used to abstract unsupported constructs, e.g.:

- non-linear multiplication in arithmetic
- ALUs in circuits

Example: The formula

$$a * (|b| + c) = d \quad \wedge \quad b * (|a| + c) \neq d \quad \wedge \quad a = b$$

is unsatisfiable, but no arithmetic reasoning is needed

If we abstract it to

$$mul(a, add(abs(b), c)) = d \quad \wedge \quad mul(b, add(abs(a), c)) \neq d \quad \wedge \quad a = b$$

it is still unsatisfiable

Very useful, for obvious reasons

Restricted fragments (over the reals or the integers) support more efficient methods:

- Bounds: $x \bowtie k$ with $\bowtie \in \{<, >, \leq, \geq, =\}$ [BBC+05a]
- Difference logic: $x - y \bowtie k$, with $\bowtie \in \{<, >, \leq, \geq, =\}$ [NO05, WIGG05, CM06]
- UTVPI: $\pm x \pm y \bowtie k$, with $\bowtie \in \{<, >, \leq, \geq, =\}$ [LM05]
- Linear arithmetic, e.g: $2x - 3y + 4z \leq 5$ [DdM06a]
- Non-linear arithmetic, e.g: $2xy + 4xz^2 - 5y \leq 10$ [BLNM+09, ZM10, JdM12]

Used in software verification and hardware verification (for memories) [SBDL01, BNO+08a, dMB09a]

Two interpreted function symbols _[_] and store

Axiomatized by:

- $\forall a\, \forall i\, \forall v.\ \mathrm{store}(a, i, v)[i] = v$
- $\forall a\, \forall i\, \forall j\, \forall v.\ i \neq j \Rightarrow \mathrm{store}(a, i, v)[j] = a[j])$

Sometimes also with *extensionality*:

- $\forall a\, \forall b.\ (\forall i.\ a[i] = b[i]\ \Rightarrow a = b)$

Is the following set of literals satisfiable in this theory?

$$\mathrm{store}(a, i, x) \neq b,\ \ b[i] = y,\ \ \mathrm{store}(b, i, x)[j] = y,\ \ a = b,\ \ i = j$$

Useful both in hardware and software verification [BCF+07, BB09a, HBJ+14b]

Universe consists of (fixed-sized) vectors of bits

Different types of operations:

- *String-like*: concat, extract, ...
- *Logical*: bit-wise not, or, and, ...
- *Arithmetic*: add, subtract, multiply, ...
- *Comparison*: $<, >$, ...

Is this formula satisfiable over bit vectors of size 3?

$$a[1:0] \neq b[1:0] \ \wedge \ (a \mid b) = c \ \wedge \ c[0:0] = 0 \ \wedge \ a[1:0] + b[1:0] = 0$$

OTHER INTERESTING THEORIES

- Floating point arithmetic [BDG+14, ZWR14]
- Ordinary differential equations [GKC13]
- (Co)Algebraic data-types [BST07, RB16]
- Strings and regular expressions [LRT+14, KGG+09]
- Finite sets with cardinality [BRBT16]
- Finite relations [MRTB17]
- ...

THEORY SOLVERS

Given a theory T, a *Theory Solver* for T takes as input a set Φ of literals and determines whether Φ is T-satisfiable.

Φ is *T-satisfiable* iff there is some model M of T such that each formula in Φ holds in M.

- Literals are of the form $t_1 = t_2$ and $t_1 \neq t_2$
- Can be decided in $O(n \log(n))$ based on congruence closure
- Efficient theory propagation for equalities
- Can generate:
  - small explanations [DNS05]
  - minimal (i.e., non-redundant) explanations [NO07b]
  - smallest explanations (NP-hard) [FFHP]
- Typically the core of the SMT solver and used in other theories

Main idea: apply congruence axiom:

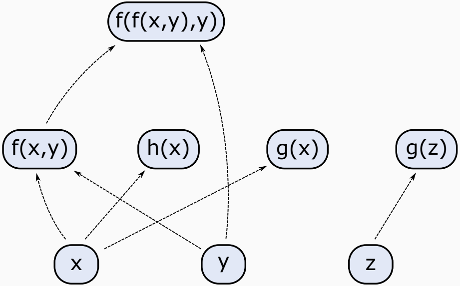
$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Main idea: apply congruence axiom:

$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

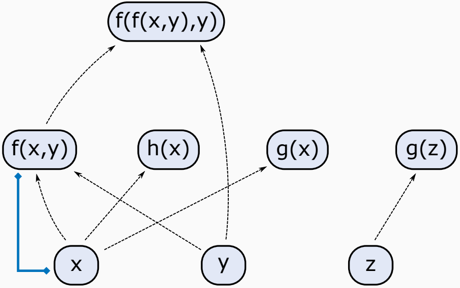


Main idea: apply congruence axiom:

$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

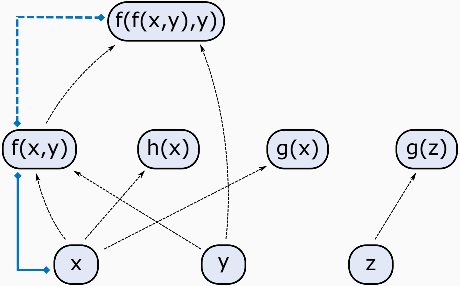


Main idea: apply congruence axiom:

$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

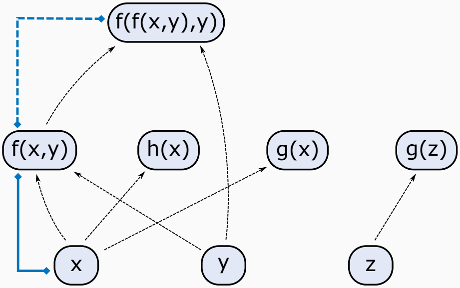


Main idea: apply congruence axiom:

$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

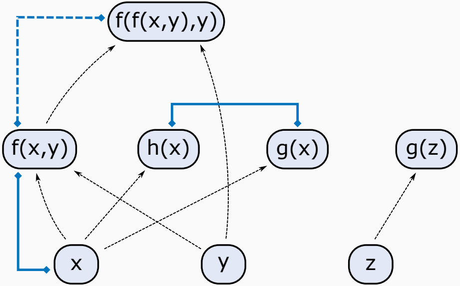


Main idea: apply congruence axiom:

$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

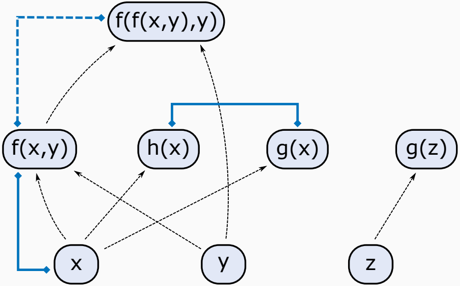


Main idea: apply congruence axiom:

$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

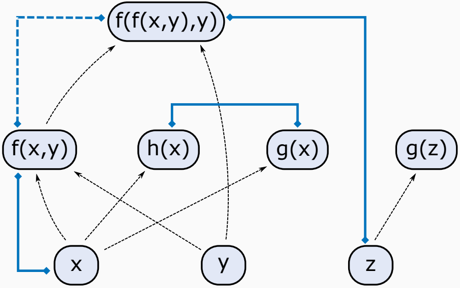


Main idea: apply congruence axiom:

$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

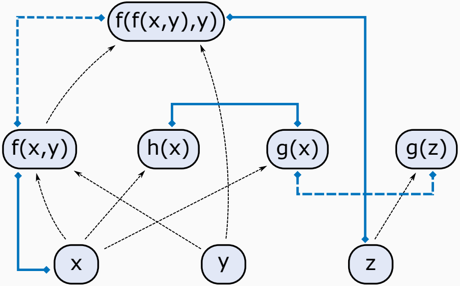


Main idea: apply congruence axiom:

$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

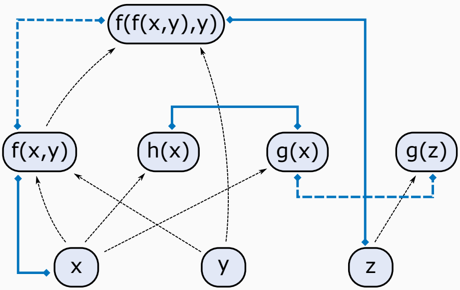


Main idea: apply congruence axiom:

$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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



Main idea: apply congruence axiom:

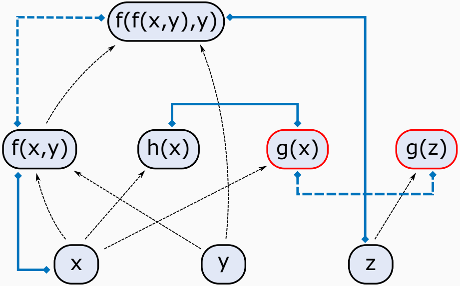
$$x_1 = y_1 \land \cdots \land x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

Conflict Set:

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



Main idea: apply congruence axiom:

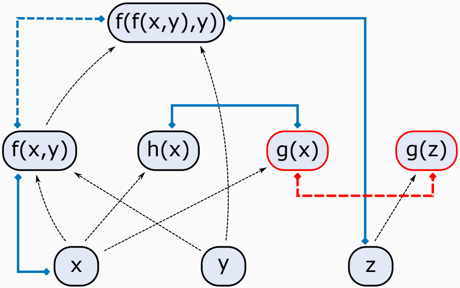
$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

Conflict Set:

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



Main idea: apply congruence axiom:

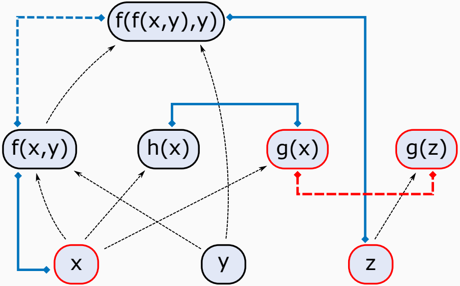
$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

Conflict Set:

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



Main idea: apply congruence axiom:

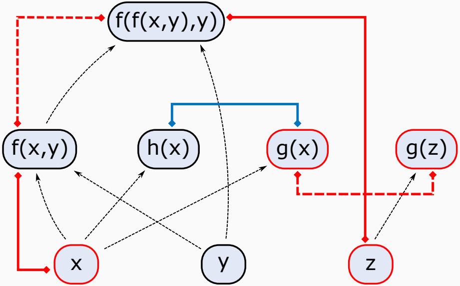
$$x_1 = y_1 \land \cdots \land x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

Conflict Set:

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



Main idea: apply congruence axiom:

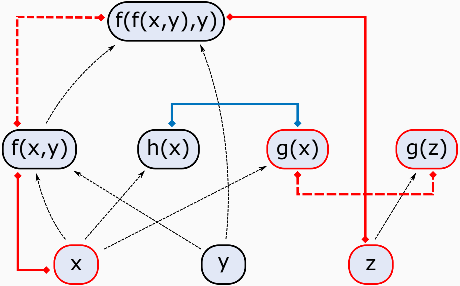
$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

Conflict Set:

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



Main idea: apply congruence axiom:

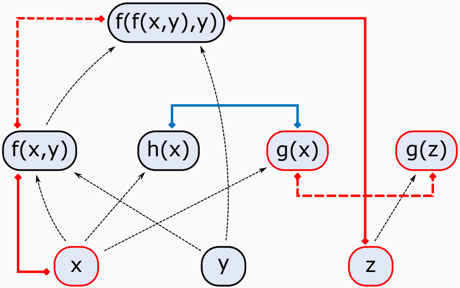
$$x_1 = y_1 \wedge \cdots \wedge x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Example

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

Conflict Set:

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



$$\begin{aligned}
\forall a, i, e &: \ \mathrm{store}(a, i, e)[i] = e \\
\forall a, i, j, e &: \ i \neq j \Rightarrow \mathrm{store}(a, i, e)[j] = a[j] \\
\forall a, b &: \ a \neq b \Rightarrow \exists i : a[i] \neq b[i]
\end{aligned}$$

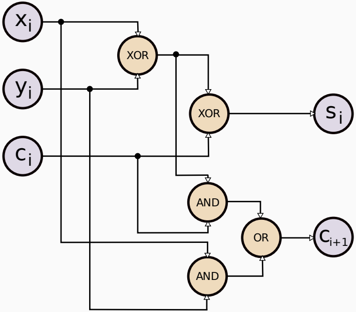
Common approach:

- UF + lemmas on demand [BB09b, DMB09b]
- Use EUF as if store and _[_] were uninterpreted
- If UNSAT in EUF, then UNSAT in arrays too
- If SAT and solution satisfies array axioms, then SAT (lucky case)
- If not, then refine by instantiating violated axioms

Common approach:

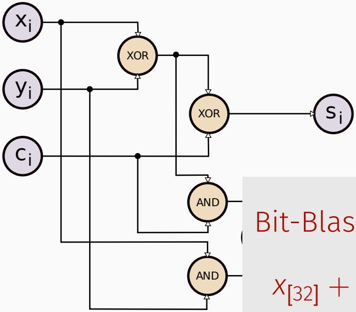
1. Simplify/preprocess (heavily)
2. Encode to SAT (aka, bit blasting)
3. Send to a SAT solver

Alternatives [HBJ+14a, ZWR16] not yet mature



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



Translation to CNF

- Each node a new variables
- XOR introduces 4 clauses
- AND introduces 3 clauses

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

Language:

- Literals of the form

$$x - y \leq k$$

  with x and y variables (integer or real) and k constant (integer or real)

- Reductions:

$$\begin{array}{lcll} x - y = k & \longrightarrow & x - y \leq k \ \wedge \ y - x \leq k & \\ x - y < k & \longrightarrow & x - y \leq k - 1 & \text{(integers)} \\ x - y < k & \longrightarrow & x - y \leq k - \delta & \text{(reals)} \end{array}$$

- Any solution to a set of literals can be shifted:
  - if v is a satisfying assignment, so is $v' = \lambda x.\ v(x) + k$
- 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,
  - in reals, satisfiability is polynomial
  - in integers, satisfiability is NP-hard

Common approach: Cycle detection

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

Theorem *Literals unsatisfiable* $\Leftrightarrow \exists$ *negative path*

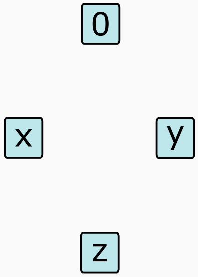
DIFFERENCE LOGIC: FROM LITERALS TO GRAPH

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

Theorem *Literals unsatisfiable* $\Leftrightarrow \exists$ *negative path*

Example

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

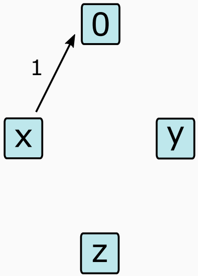


1. Construct a graph from literals
2. Check if there is a negative path

Theorem *Literals unsatisfiable* $\Leftrightarrow \exists$ *negative path*

Example

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

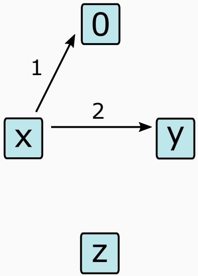


1. Construct a graph from literals
2. Check if there is a negative path

Theorem *Literals unsatisfiable* $\Leftrightarrow \exists$ *negative path*

Example

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

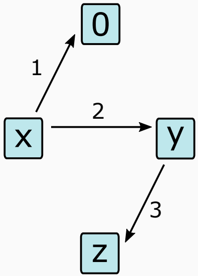


1. Construct a graph from literals
2. Check if there is a negative path

Theorem *Literals unsatisfiable* $\Leftrightarrow \exists$ *negative path*

Example

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

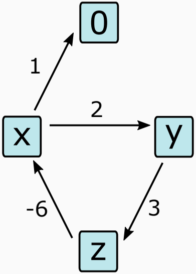


1. Construct a graph from literals
2. Check if there is a negative path

Theorem *Literals unsatisfiable* $\Leftrightarrow \exists$ *negative path*

Example

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

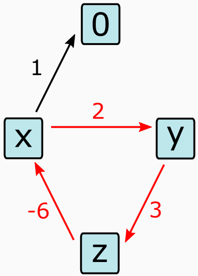


1. Construct a graph from literals
2. Check if there is a negative path

Theorem *Literals unsatisfiable* $\Leftrightarrow \exists$ *negative path*

Example

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



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

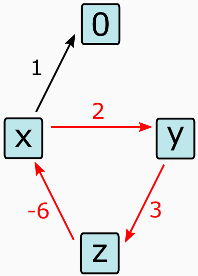
Theorem *Literals unsatisfiable* $\Leftrightarrow \exists$ *negative path*

Example

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

Conflict Set:

$x - y \leq 2,\ y - z \leq 3,\ z - x \leq -6$



Language:

- Literals of the form $a_1x_1 + \cdots + a_nx_n \bowtie b$ with a_1 positive and $\bowtie \in \{\leq, \geq\}$
- Reductions:

$$\begin{array}{lcll} t = b & \longrightarrow & t \leq b \ \wedge \ t \geq b & \\ t < b & \longrightarrow & t \leq b - 1 & \text{(integer arith.)} \\ t < b & \longrightarrow & t \leq b - \delta & \text{(real arith.)} \end{array}$$

Common approach: variant of Simplex designed for SMT [DDM06b]

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

Rewrite each $\sum a_i x_i \bowtie b$ as $s \bowtie b$ with $s = \sum a_i x_i$

We get tableau of equations + simple bounds on variables

- Tableau is fixed (modulo pivoting and substitutions)
- 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

$$
\begin{gathered}
\vdots \\
l_i \le s_i \le u_i \\
\vdots \\
l_j \le x_j \le u_j \\
\vdots
\end{gathered}
$$

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

$$\vdots$$

$$l_i \leq s_i \leq u_i$$

$$\vdots$$

$$l_j \leq x_j \leq u_j$$

$$\vdots$$

Tableau

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

Bounds

$$
\begin{gathered}
\vdots \\
l_i \leq s_i \leq u_i \\
\vdots \\
l_j \leq x_j \leq u_j \\
\vdots
\end{gathered}
$$

- lhs variables are basic, rhs variables are non-basic

Tableau | **Bounds**

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

$$
\begin{gathered}
\vdots \\
l_i \leq s_i \leq u_i \\
\vdots \\
l_j \leq x_j \leq u_j \\
\vdots
\end{gathered}
$$

- lhs variables are basic, rhs variables are non-basic

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

$$
\begin{gathered}
\vdots \\
l_i \leq s_i \leq u_i \\
\vdots \\
l_j \leq x_j \leq u_j \\
\vdots
\end{gathered}
$$

- lhs variables are basic, rhs variables are non-basic

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

$$
\begin{gathered}
\vdots \\
l_i \leq s_i \leq u_i \\
\vdots \\
l_j \leq x_j \leq u_j \\
\vdots
\end{gathered}
$$

- lhs variables are basic, rhs variables are non-basic
- Keep an assignment v of all variables:
  - v satisfies the tableau,
  - v satisfies bounds on the non-basic variables

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

$$
\begin{gathered}
\vdots \\
l_i \leq s_i \leq u_i \\
\vdots \\
l_j \leq x_j \leq u_j \\
\vdots
\end{gathered}
$$

- lhs variables are basic, rhs variables are non-basic
- Keep an assignment v of all variables:
  - v satisfies the tableau,
  - v satisfies bounds on the non-basic variables
- Initially $v(x) = 0$ and $-\infty \leq x \leq +\infty$

Tableau | **Bounds**

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

$$
\begin{gathered}
\vdots \\
l_i \leq s_i \leq u_i \\
\vdots \\
l_j \leq x_j \leq u_j \\
\vdots
\end{gathered}
$$

Case 1:

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

Tableau	Bounds
$s_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n$	$\vdots$
$\vdots$	$l_i \leq s_i \leq u_i$
$s_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n$	$\vdots$
$\vdots$	$l_j \leq x_j \leq u_j$
$s_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n$	$\vdots$

Case 2:

- v doesn't satisfy bound on some 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

Tableau

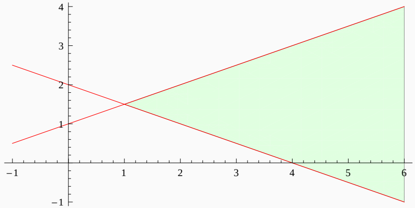
$$
\begin{aligned}
s_1 &= a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n \\
&\vdots \\
s_i &= a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n \\
&\vdots \\
s_m &= a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n
\end{aligned}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

Bounds

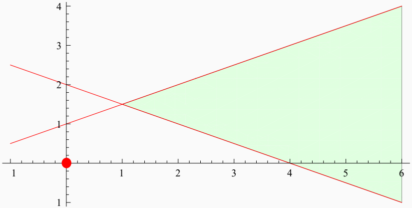
$$
\begin{gathered}
\vdots \\
l_i \leq s_i \leq u_i \\
\vdots \\
l_j \leq x_j \leq u_j \\
\vdots
\end{gathered}
$$

Case 3:

- v doesn't satisfy bound on some s_i, and some x_j's that s_i depends on has some slack
- Pivot, substitute, and continue

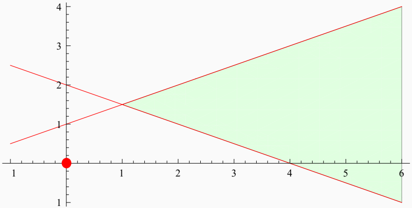


$$[\; 2y - x - 2 \leq 0,\;\; -2y - x + 4 \leq 0 \;]$$



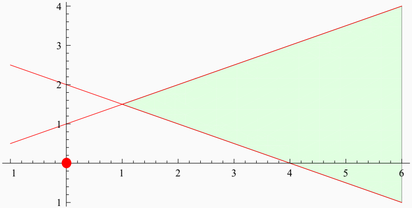
$$[\, s_1 \leq 2,\ s_2 \leq -4 \,]$$

Tableau	Bounds	Assignment
	$-\infty \leq x \leq +\infty$	$x \mapsto 0$
$s_1 = 2y - x$	$-\infty \leq y \leq +\infty$	$y \mapsto 0$
$s_2 = -2y - x$	$-\infty \leq s_1 \leq +\infty$	$s_1 \mapsto 0$
	$-\infty \leq s_2 \leq +\infty$	$s_2 \mapsto 0$



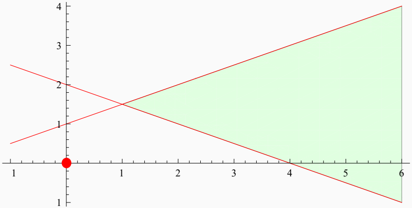
$$[\, s_1 \le 2,\ s_2 \le -4 \,]$$

Tableau	Bounds	Assignment
	$-\infty \le x \le +\infty$	$x \mapsto 0$
$s_1 = 2y - x$	$-\infty \le y \le +\infty$	$y \mapsto 0$
$s_2 = -2y - x$	$-\infty \le s_1 \le +\infty$	$s_1 \mapsto 0$
	$-\infty \le s_2 \le +\infty$	$s_2 \mapsto 0$



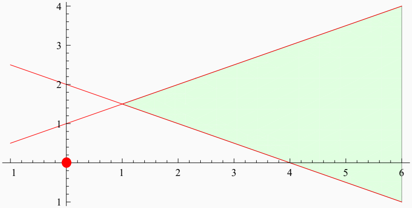
$[\, s_1 \leq 2,\ s_2 \leq -4 \,]$

Tableau	Bounds	Assignment
	$-\infty \leq x \leq +\infty$	$x \mapsto 0$
$s_1 = 2y - x$	$-\infty \leq y \leq +\infty$	$y \mapsto 0$
$s_2 = -2y - x$	$-\infty \leq s_1 \leq 2$	$s_1 \mapsto 0$
	$-\infty \leq s_2 \leq +\infty$	$s_2 \mapsto 0$



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

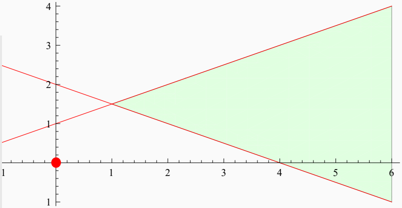
Tableau	Bounds	Assignment
	$-\infty \leq x \leq +\infty$	$x \mapsto 0$
$s_1 = 2y - x$	$-\infty \leq y \leq +\infty$	$y \mapsto 0$
$s_2 = -2y - x$	$-\infty \leq s_1 \leq 2$	$s_1 \mapsto 0$
	$-\infty \leq s_2 \leq +\infty$	$s_2 \mapsto 0$



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

Tableau	Bounds	Assignment
	$-\infty \leq x \leq +\infty$	$x \mapsto 0$
$s_1 = 2y - x$	$-\infty \leq y \leq +\infty$	$y \mapsto 0$
$s_2 = -2y - x$	$-\infty \leq s_1 \leq 2$	$s_1 \mapsto 0$
	$-\infty \leq s_2 \leq -4$	$s_2 \mapsto 0$

s_2's bound violated



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

Tableau

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

Bounds

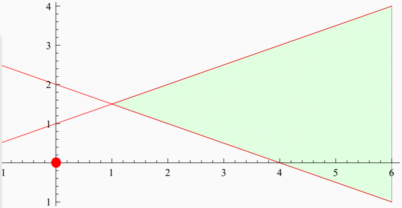
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

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

s_2's bound violated

- There is slack in y



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

Tableau

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

Bounds

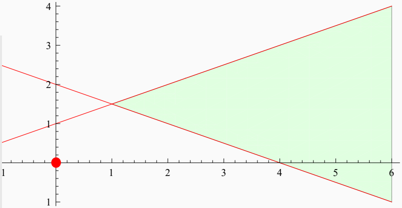
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

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

s_2's bound violated

- There is slack in y
- Pivot s_2 and y



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

Bounds

Assignment

Tableau

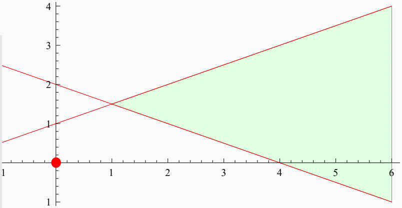
$$s_1 = 2y - x$$
$$s_2 = -2y - x$$

$-\infty \leq x \leq +\infty$	$x \mapsto 0$
$-\infty \leq y \leq +\infty$	$y \mapsto 0$
$-\infty \leq s_1 \leq 2$	$s_1 \mapsto 0$
$-\infty \leq s_2 \leq -4$	$s_2 \mapsto 0$

LINEAR RATIONAL ARITHMETIC: EXAMPLE

s_2's bound violated

- There is slack in y
- Pivot s_2 and y



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

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

$$x \mapsto 0$$

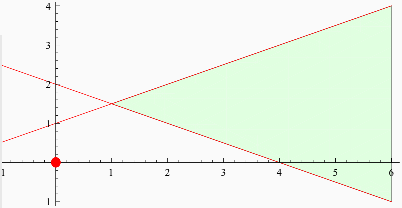
$$y \mapsto 0$$

$$s_1 \mapsto 0$$

$$s_2 \mapsto 0$$

s_2's bound violated

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



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

Tableau

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

Bounds

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

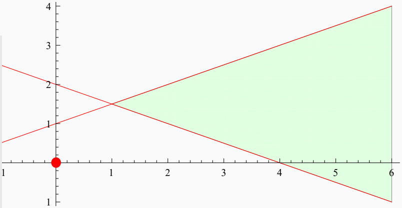
Assignment

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

LINEAR RATIONAL ARITHMETIC: EXAMPLE

s_2's bound violated

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



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

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

$$x \mapsto 0$$

$$y \mapsto 0$$

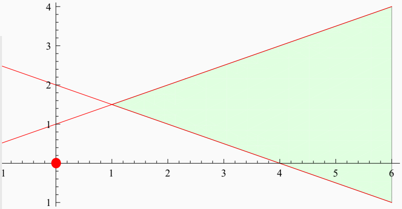
$$s_1 \mapsto 0$$

$$s_2 \mapsto -4$$

LINEAR RATIONAL ARITHMETIC: EXAMPLE

s_2's bound violated

- There is slack in y
- Pivot s_2 and y
- Update s_2 value
- Update basic vars



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

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

$$x \mapsto 0$$

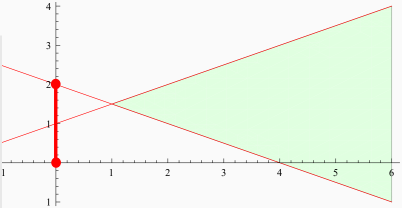
$$y \mapsto 0$$

$$s_1 \mapsto 0$$

$$s_2 \mapsto -4$$

s_2's bound violated

- There is slack in y
- Pivot s_2 and y
- Update s_2 value
- Update basic vars



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

Tableau

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

Bounds

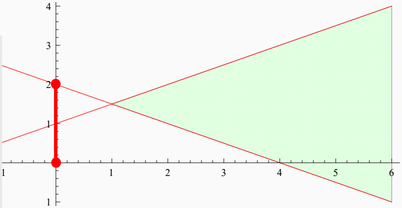
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 2$$
$$s_1 \mapsto 4$$
$$s_2 \mapsto -4$$

LINEAR RATIONAL ARITHMETIC: EXAMPLE

s_1's bound violated



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

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

$$x \mapsto 0$$

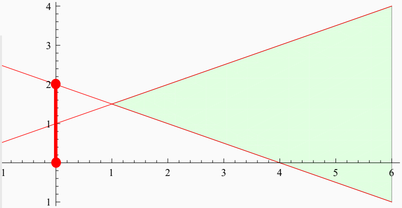
$$y \mapsto 2$$

$$s_1 \mapsto 4$$

$$s_2 \mapsto -4$$

s_1's bound violated

- There is slack in x



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

Tableau

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

Bounds

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

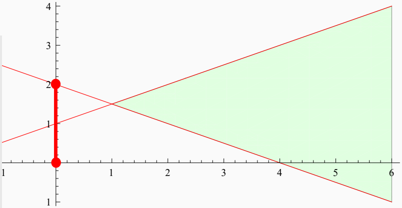
Assignment

$$x \mapsto 0$$
$$y \mapsto 2$$
$$s_1 \mapsto 4$$
$$s_2 \mapsto -4$$

LINEAR RATIONAL ARITHMETIC: EXAMPLE

s_1's bound violated

- There is slack in x
- Pivot s_1 and x



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

Tableau

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

Bounds

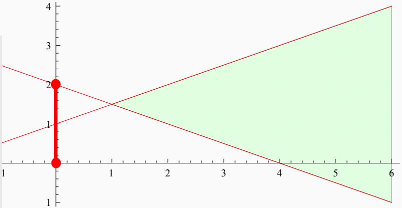
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 2$$
$$s_1 \mapsto 4$$
$$s_2 \mapsto -4$$

s_1's bound violated

- There is slack in x
- Pivot s_1 and x



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

Tableau

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

Bounds

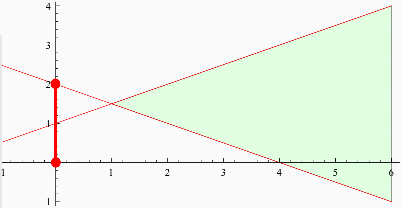
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 2$$
$$s_1 \mapsto 4$$
$$s_2 \mapsto -4$$

s_1's bound violated

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



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

Tableau

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

Bounds

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

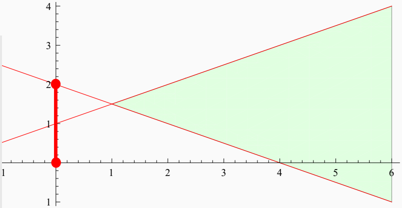
Assignment

$$x \mapsto 0$$
$$y \mapsto 2$$
$$s_1 \mapsto 4$$
$$s_2 \mapsto -4$$

LINEAR RATIONAL ARITHMETIC: EXAMPLE

s_1's bound violated

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



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

Tableau

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

Bounds

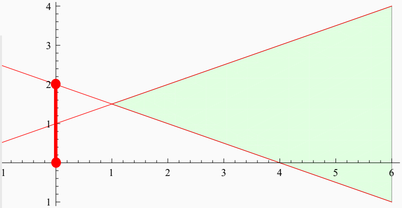
$$-\infty \leq x \leq +\infty$$
$$-\infty \leq y \leq +\infty$$
$$-\infty \leq s_1 \leq 2$$
$$-\infty \leq s_2 \leq -4$$

Assignment

$$x \mapsto 0$$
$$y \mapsto 2$$
$$s_1 \mapsto 2$$
$$s_2 \mapsto -4$$

s_1's bound violated

- There is slack in x
- Pivot s_1 and x
- Update s_1 value
- Update basic vars



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

Tableau

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

Bounds

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

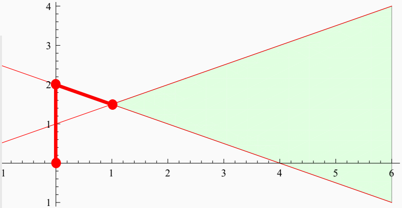
Assignment

$$x \mapsto 0$$
$$y \mapsto 2$$
$$s_1 \mapsto 2$$
$$s_2 \mapsto -4$$

LINEAR RATIONAL ARITHMETIC: EXAMPLE

s_1's bound violated

- There is slack in x
- Pivot s_1 and x
- Update s_1 value
- Update basic vars



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

Tableau

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

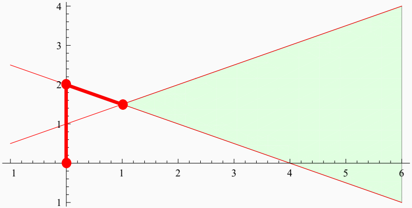
Bounds

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

Assignment

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

LINEAR RATIONAL 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

$$x \mapsto 1$$

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

$$s_1 \mapsto 2$$

$$s_2 \mapsto -4$$

Classic NP-complete problem [Pap81]

Admits quantifier elimination [Coo72]

Common approach:

- Simplex + Branch-And-Bound [DDM06b, Gri12, Kin14]
- Use Simplex to solve real relaxation (treat variables as real)
- If UNSAT over reals, then UNSAT over integers too
- If SAT and solution v is integral, then SAT (lucky case)
- Otherwise, refine:
  - Add branch-and-bound lemmas: $x \leq \lfloor v(x) \rfloor \ \vee \ x \geq \lceil v(x) \rceil$
  - Add cutting plane lemmas: new implied inequality falsified by v
- Additionally solve integer equalities
- Not guaranteed to terminate

Common approach:

- Simplex + Branch-And-Bound [DDM06b, Gri12, Kin14]
- Use Simplex to solve real relaxation (treat variables as real)
- If UNSAT over reals, then UNSAT over integers too
- If SAT and solution v is integral, then SAT (lucky case)
- Otherwise, refine:
  - Add branch-and-bound lemmas: $x \leq \lfloor v(x) \rfloor \ \vee \ x \geq \lceil v(x) \rceil$
  - Add cutting plane lemmas: new implied inequality falsified by v
- Additionally solve integer equalities
- Not guaranteed to terminate

Alternatives [JdM13, BSW15] not yet mature

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

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

Examples

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

Polynomial Constraints

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

$$p_1 > 0 \lor (p_2 = 0 \land p_3 < 0) \qquad p_1, p_2, p_3 \in \mathbb{Z}[x_1, \dots, x_n]$$

Projection (Saturation)

Project polynomials using a projection P

$$\{p_1, p_2, p_3\} \mapsto \{p_1, p_2, p_3, p_4, \dots, p_n\}$$

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

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

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

Explain Conflicts

Explain conflicts using valid clausal reasons

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

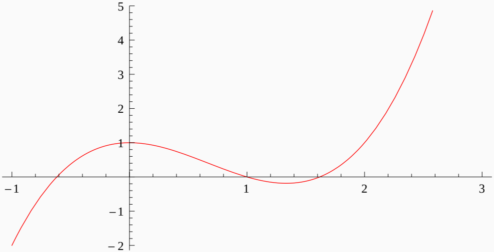
Unit Reasoning

Reason about unit constraints

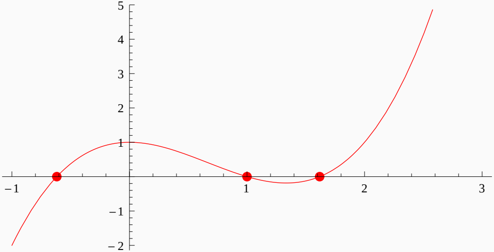
$$C_1 \equiv (x^2 + y^2 < 1) \qquad C_2 \equiv (xy > 1)$$

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

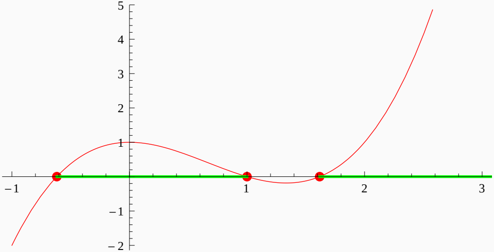
NON-LINEAR REAL ARITHMETIC



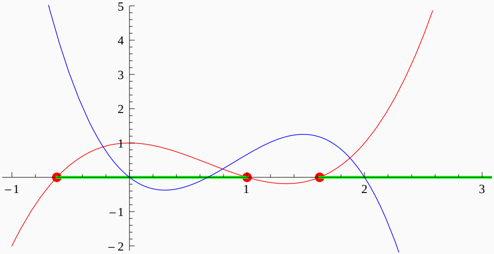
$$x^3 - 2x^2 + 1 > 0$$



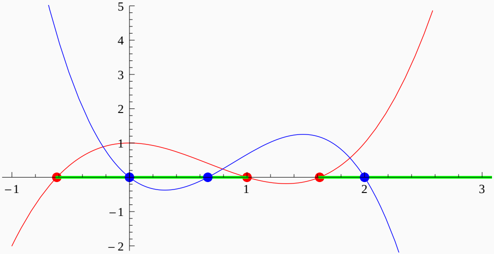
$$x^3 - 2x^2 + 1 > 0$$



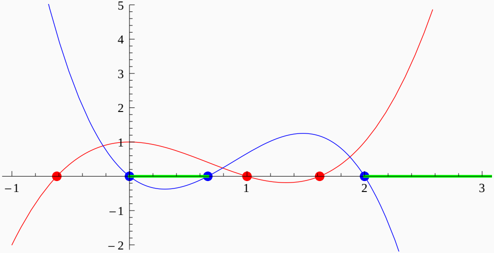
$$x^3 - 2x^2 + 1 > 0$$



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



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



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

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

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

Explain Conflicts

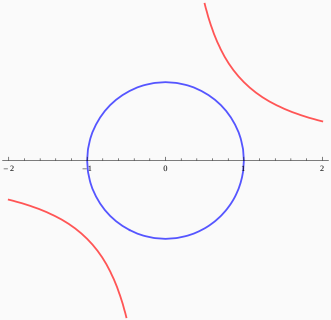
Explain conflicts using valid clausal reasons

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

Explain Conflicts

Explain conflicts using valid clausal reasons

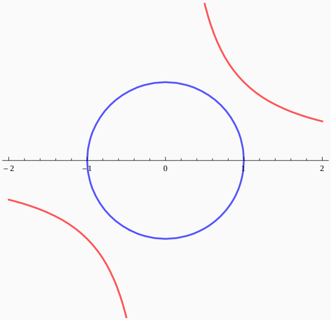
$$\overline{C_1} \vee \overline{C_2} \vee x \leq 0 \vee x \geq 1$$



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

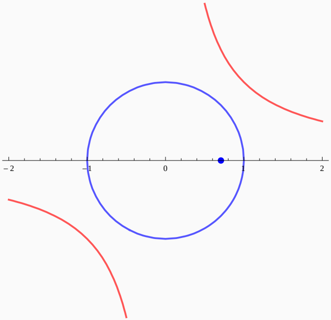
$$[]$$

NON-LINEAR REAL ARITHMETIC



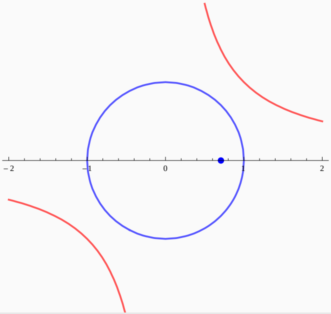
$$\overbrace{x^2 + y^2 < 1}^{C_1} \;\wedge\; \overbrace{xy > 1}^{C_2}$$

$$[C_1, C_2]$$



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

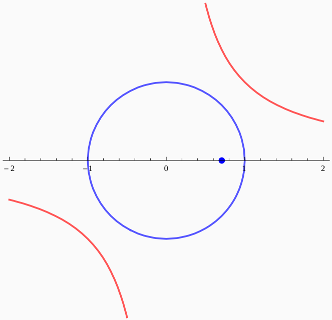
$$[C_1, C_2, x \mapsto \sqrt{2}/2]$$



Unit Constraint Reasoning

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

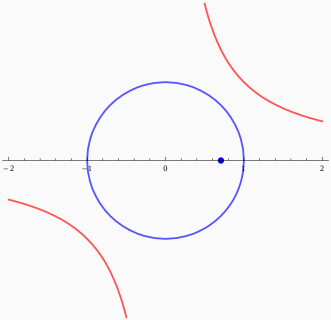
$$-2y - x + 4 < 0 \Rightarrow y > \sqrt{2}$$



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

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

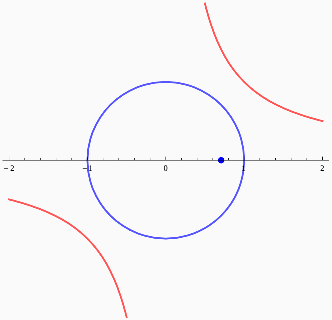
Explanation $C_1 \wedge C_2 \Rightarrow x \neq \sqrt{2}/2$



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

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

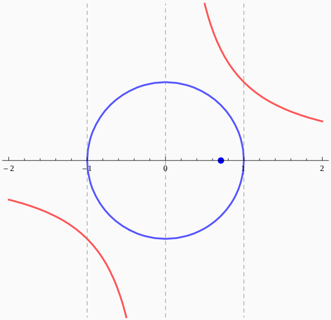
Explanation $C_1 \land C_2 \Rightarrow$



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

CAD Projection

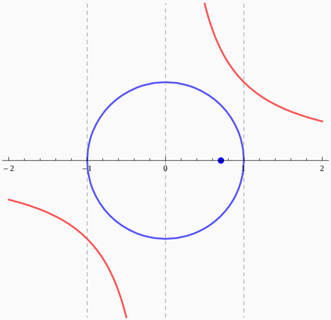
$$P = \{x, -4 + 4x^2, 1 - x^2 + x^4\}$$



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

CAD Projection

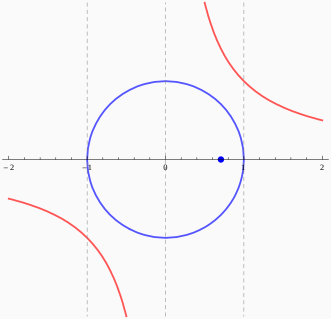
$$P = \{x, -4 + 4x^2, 1 - x^2 + x^4\}$$



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

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

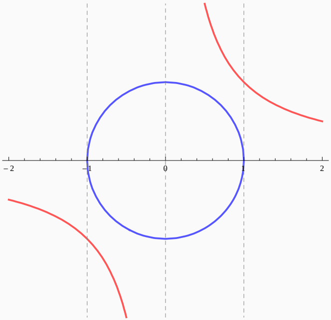
Explanation $C_1 \wedge C_2 \Rightarrow x \leq 0 \vee x \geq 1$



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

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

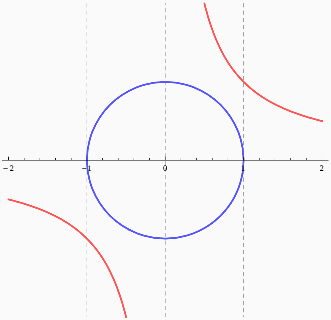
Explanation $\overline{C_1} \vee \overline{C_2} \vee x \leq 0 \vee x \geq 1$



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

$$[C_1, C_2]$$

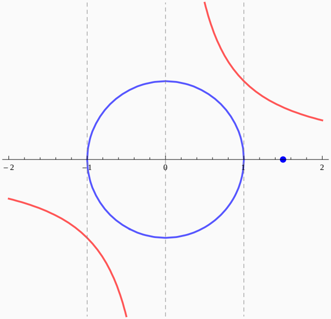
Explanation $\overline{C_1} \lor \overline{C_2} \lor x \leq 0 \lor x \geq 1$



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

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

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



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

$$[C_1, C_2, x \geq 1, x \mapsto 3/2]$$

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

EXTENDING THEORY SOLVERS TO QFFS

Def. A formula is *(un)satisfiable in* a theory T, or *T-(un)satisfiable*, if there is a (no) model of T that satisfies it

Def. A formula is *(un)satisfiable in* a theory T, or *T-(un)satisfiable*, if there is a (no) model of T that satisfies it

Note: The T-satisfiability of quantifier-free formulas is decidable iff the T-satisfiability of conjunctions/sets of literals is decidable

Def. A formula is *(un)satisfiable in* a theory T, or T-*(un)satisfiable*, if there is a (no) model of T that satisfies it

Note: The T-satisfiability of quantifier-free formulas is decidable iff the T-satisfiability of conjunctions/sets of literals is decidable

(Convert the formula in DNF and check if any of its disjuncts is T-sat)

Def. A formula is *(un)satisfiable in* a theory T, or T-*(un)satisfiable*, if there is a (no) model of T that satisfies it

Note: The T-satisfiability of quantifier-free formulas is decidable iff the T-satisfiability of conjunctions/sets of literals is decidable

(Convert the formula in DNF and check if any of its disjuncts is T-sat)

Problem: In practice, dealing with Boolean combinations of literals is as hard as in propositional logic

Def. A formula is *(un)satisfiable in* a theory T, or T*-(un)satisfiable*, if there is a (no) model of T that satisfies it

Note: The T-satisfiability of quantifier-free formulas is decidable iff the T-satisfiability of conjunctions/sets of literals is decidable

(Convert the formula in DNF and check if any of its disjuncts is T-sat)

Problem: In practice, dealing with Boolean combinations of literals is as hard as in propositional logic

Solution: Exploit propositional satisfiability technology

Two main approaches:

1. "Eager" [PRSS99, SSB02, SLB03, BGV01, BV02]

- translate into an equisatisfiable propositional formula
- feed it to any SAT solver

Notable systems: UCLID

Two main approaches:

1. "Eager" [PRSS99, SSB02, SLB03, BGV01, BV02]

- translate into an equisatisfiable propositional formula
- feed it to any SAT solver

Notable systems: UCLID

2. "Lazy" [ACG00, dMR02, BDS02, ABC+02]

- abstract the input formula to a propositional one
- feed it to a (DPLL-based) SAT solver
- use a theory decision procedure to refine the formula and guide the SAT solver

Notable systems: Barcelogic, Boolector, CVC4, MathSAT, Yices, veriT, Z3

Two main approaches:

1. "Ea

We focus on the lazy approach

- tra l formula
- fee

Notable systems: UCLID

2. "Lazy" [ACG00, dMR02, BDS02, ABC+02]

- abstract the input formula to a propositional one
- feed it to a (DPLL-based) SAT solver
- use a theory decision procedure to refine the formula and guide the SAT solver

Notable systems: Barcelogic, Boolector, CVC4, MathSAT, Yices, veriT, Z3

$$g(a) = c \quad \wedge \quad f(g(a)) \neq f(c) \;\vee\; g(a) = d \quad \wedge \quad c \neq d$$

Theory T: Equality with Uninterpreted Functions

$$g(a) = c \quad \wedge \quad f(g(a)) \neq f(c) \ \vee \ g(a) = d \quad \wedge \quad c \neq d$$

Theory T: Equality with Uninterpreted Functions

Simplest setting:

- Off-line SAT solver
- Non-incremental *theory solver* for conjunctions of equalities and disequalities
- Theory atoms (e.g., $g(a) = c$) abstracted to propositional atoms (e.g., 1)

(VERY) LAZY APPROACH FOR SMT – EXAMPLE

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

(very) lazy approach for SMT – example

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1,\ \overline{2} \vee 3,\ \overline{4}\}$ to SAT solver.

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1,\ \overline{2} \vee 3,\ \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1,\ \overline{2},\ \overline{4}\}$.
  Theory solver finds (concretization of) $\{1,\ \overline{2},\ \overline{4}\}$ unsat.

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1, \overline{2} \vee 3, \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1, \overline{2}, \overline{4}\}$.
  Theory solver finds (concretization of) $\{1, \overline{2}, \overline{4}\}$ unsat.
- Send $\{1, \overline{2} \vee 3, \overline{4}, \overline{1} \vee 2 \vee 4\}$ to SAT solver.

(Very) lazy approach for SMT – example

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1, \overline{2} \vee 3, \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1, \overline{2}, \overline{4}\}$.
  Theory solver finds (concretization of) $\{1, \overline{2}, \overline{4}\}$ unsat.
- Send $\{1, \overline{2} \vee 3, \overline{4}, \overline{1} \vee 2 \vee 4\}$ to SAT solver.
- SAT solver returns model $\{1, 3, \overline{4}\}$.
  Theory solver finds $\{1, 3, \overline{4}\}$ unsat.

(VERY) LAZY APPROACH FOR SMT – EXAMPLE

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1, \overline{2} \vee 3, \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1, \overline{2}, \overline{4}\}$.
  Theory solver finds (concretization of) $\{1, \overline{2}, \overline{4}\}$ unsat.
- Send $\{1, \overline{2} \vee 3, \overline{4}, \overline{1} \vee 2 \vee 4\}$ to SAT solver.
- SAT solver returns model $\{1, 3, \overline{4}\}$.
  Theory solver finds $\{1, 3, \overline{4}\}$ unsat.
- Send $\{1, \overline{2} \vee 3, \overline{4}, \overline{1} \vee 2, \overline{1} \vee \overline{3} \vee 4\}$ to SAT solver.

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1, \overline{2} \vee 3, \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1, \overline{2}, \overline{4}\}$.
  Theory solver finds (concretization of) $\{1, \overline{2}, \overline{4}\}$ unsat.
- Send $\{1, \overline{2} \vee 3, \overline{4}, \overline{1} \vee 2 \vee 4\}$ to SAT solver.
- SAT solver returns model $\{1, 3, \overline{4}\}$.
  Theory solver finds $\{1, 3, \overline{4}\}$ unsat.
- Send $\{1, \overline{2} \vee 3, \overline{4}, \overline{1} \vee 2, \overline{1} \vee \overline{3} \vee 4\}$ to SAT solver.
- SAT solver finds $\{1, \overline{2} \vee 3, \overline{4}, \overline{1} \vee 2 \vee 4, \overline{1} \vee \overline{3} \vee 4\}$ unsat.

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1,\ \overline{2} \vee 3,\ \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1,\ \overline{2},\ \overline{4}\}$.
  Theory solver finds (concretization of) $\{1,\ \overline{2},\ \overline{4}\}$ unsat.
- Send $\{1,\ \overline{2} \vee 3,\ \overline{4},\ \overline{1} \vee 2 \vee 4\}$ to SAT solver.
- SAT solver returns model $\{1,\ 3,\ \overline{4}\}$.
  Theory solver finds $\{1,\ 3,\ \overline{4}\}$ unsat.
- Send $\{1,\ \overline{2} \vee 3,\ \overline{4},\ \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4\}$ to SAT solver.
- SAT solver finds $\{1,\ \overline{2} \vee 3,\ \overline{4},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee \overline{3} \vee 4\}$ unsat.
  Done: the original formula is unsatisfiable in UF.

Several enhancements are possible to increase efficiency:

- Check T-satisfiability only of full propositional model

Several enhancements are possible to increase efficiency:

- ~~Check T-satisfiability only of full propositional model~~
- Check T-satisfiability of partial assignment M as it grows

Several enhancements are possible to increase efficiency:

- ~~Check T-satisfiability only of full propositional model~~
- Check T-satisfiability of partial assignment M as it grows
- If M is T-unsatisfiable, add $\neg M$ as a clause

Several enhancements are possible to increase efficiency:

- ~~Check T-satisfiability only of full propositional model~~
- Check T-satisfiability of partial assignment M as it grows

- ~~If M is T-unsatisfiable, add $\neg M$ as a clause~~
- If M is T-unsatisfiable, identify a T-unsatisfiable subset M_0 of M and add $\neg M_0$ as a clause

Several enhancements are possible to increase efficiency:

- ~~Check T-satisfiability only of full propositional model~~
- Check T-satisfiability of partial assignment M as it grows

- ~~If M is T-unsatisfiable, add $\neg M$ as a clause~~
- If M is T-unsatisfiable, identify a T-unsatisfiable subset M_0 of M and add $\neg M_0$ as a clause

- If M is T-unsatisfiable, add clause and restart

Several enhancements are possible to increase efficiency:

- ~~Check T-satisfiability only of full propositional model~~
- Check T-satisfiability of partial assignment M as it grows

- ~~If M is T-unsatisfiable, add $\neg M$ as a clause~~
- If M is T-unsatisfiable, identify a T-unsatisfiable subset M_0 of M and add $\neg M_0$ as a clause

- ~~If M is T-unsatisfiable, add clause and restart~~
- If M is T-unsatisfiable, backtrack to some point where the assignment was still T-satisfiable

LAZY APPROACH – MAIN BENEFITS

- Every tool does what it is good at:
  - SAT solver takes care of Boolean information
  - Theory solver takes care of theory information

- Every tool does what it is good at:
  - SAT solver takes care of Boolean information
  - Theory solver takes care of theory information
- The theory solver works only with conjunctions of literals

- Every tool does what it is good at:
  - SAT solver takes care of Boolean information
  - Theory solver takes care of theory information
- The theory solver works only with conjunctions of literals
- Modular approach:
  - SAT and theory solvers communicate via a simple API [GHN+04]
  - SMT for a new theory only requires new theory solver
  - An off-the-shelf SAT solver can be embedded in a lazy SMT system with few new lines of code (tens)

Several variants and enhancements of lazy SMT solvers exist

They can be modeled abstractly and declaratively as *transition systems*

A transition system is a binary relation over states, induced by a set of conditional transition rules

The framework can be first developed for SAT and then extended to lazy SMT [NOT06, KG07]

An abstract framework helps one:

- skip over implementation details and unimportant control aspects
- reason formally about solvers for SAT and SMT
- model advanced features such as non-chronological bactracking, lemma learning, theory propagation, ...
- describe different strategies and prove their correctness
- compare different systems at a higher level
- get new insights for further enhancements

An abstract framework helps one:

- skip over implementation details and unimportant control aspects
- reason formally about solvers for SAT and SMT
- model advanced features such as non-chronological bactracking, lemma learning, theory propagation, …
- describe different strategies and prove their correctness
- compare different systems at a higher level
- get new insights for further enhancements

The one described next is a re-elaboration of those in [NOT06, KG07]

- Modern SAT solvers are based on the DPLL procedure [DP60, DLL62]
- DPLL tries to build incrementally a satisfying truth assignment M for a CNF formula F
- M is grown by
  - deducing the truth value of a literal from M and F, or
  - guessing a truth value
- If a wrong guess for a literal leads to an inconsistency, the procedure backtracks and tries the opposite value

States:

$$\text{fail} \quad \text{or} \quad \langle M, F \rangle$$

where

- M is a sequence of literals and *decision points* $\bullet$ denoting a partial truth *assignment*
- F is a set of clauses denoting a CNF *formula*

Def. If $M = M_0 \bullet M_1 \bullet \cdots \bullet M_n$ where each M_i contains no decision points

- M_i is *decision level i* of M
- $M^{[i]} \stackrel{\text{def}}{=} M_0 \bullet \cdots \bullet M_i$

States:

$$\text{fail} \quad \text{or} \quad \langle M, F \rangle$$

Initial state:

- $\langle (), F_0 \rangle$, where F_0 is to be checked for satisfiability

Expected final states:

- fail if F_0 is unsatisfiable
- $\langle M, G \rangle$ otherwise, where
  - G is equivalent to F_0 and
  - M satisfies G

States treated like records:

- M denotes the truth assignment component of current state
- F denotes the formula component of current state

Transition rules in *guarded assignment form* [KG07]

$$\frac{p_1 \quad \cdots \quad p_n}{[\mathsf{M} := e_1] \quad [\mathsf{F} := e_2]}$$

updating M, F or both when premises $p_1, \ldots, p_n$ all hold

Extending the assignment

Propagate $$\frac{l_1 \vee \cdots \vee l_n \vee l \in \mathsf{F} \quad \bar{l}_1, \ldots, \bar{l}_n \in \mathsf{M} \quad l, \bar{l} \notin \mathsf{M}}{\mathsf{M} := \mathsf{M}\ l}$$

Note: When convenient, treat M as a set

Note: Clauses are treated modulo ACI of $\vee$

Extending the assignment

Propagate $$\frac{l_1 \vee \cdots \vee l_n \vee l \in \mathsf{F} \quad \bar{l}_1, \ldots, \bar{l}_n \in \mathsf{M} \quad l, \bar{l} \notin \mathsf{M}}{\mathsf{M} := \mathsf{M}\ l}$$

Note: When convenient, treat M as a set

Note: Clauses are treated modulo ACI of $\vee$

Decide $$\frac{l \in \mathrm{Lit}(\mathsf{F}) \quad l, \bar{l} \notin \mathsf{M}}{\mathsf{M} := \mathsf{M} \bullet l}$$

Note: $\mathrm{Lit}(F) \stackrel{\mathrm{def}}{=} \{l \mid l \text{ literal of } F\} \cup \{\bar{l} \mid l \text{ literal of } F\}$

Repairing the assignment

$$\textbf{Fail}\quad \frac{l_1 \vee \cdots \vee l_n \in F \quad \bar{l}_1, \ldots, \bar{l}_n \in M \quad \bullet \notin M}{\text{fail}}$$

Repairing the assignment

Fail $$\frac{l_1 \vee \cdots \vee l_n \in \mathsf{F} \quad \bar{l}_1, \ldots, \bar{l}_n \in \mathsf{M} \quad \bullet \notin \mathsf{M}}{\mathsf{fail}}$$

Backtrack

$$\frac{l_1 \vee \cdots \vee l_n \in \mathsf{F} \quad \bar{l}_1, \ldots, \bar{l}_n \in \mathsf{M} \quad \mathsf{M} = M \bullet l\, N \quad \bullet \notin N}{\mathsf{M} := M\, \bar{l}}$$

Note: Last premise of **Backtrack** enforces chronological backtracking

To model conflict-driven backjumping and learning, add to states a third component C whose value is either no or a *conflict clause*

To model conflict-driven backjumping and learning, add to states a third component C whose value is either no or a *conflict clause*

States: fail or $\langle M, F, C\rangle$

Initial state:

- $\langle (), F_0, \text{no}\rangle$, where F_0 is to be checked for satisfiability

Expected final states:

- fail if F_0 is unsatisfiable
- $\langle M, G, \text{no}\rangle$ otherwise, where
  - G is equivalent to F_0 and
  - M satisfies G

Replace Backtrack with

Replace Backtrack with

Conflict $$\frac{C = \text{no} \quad l_1 \vee \cdots \vee l_n \in F \quad \bar{l}_1, \ldots, \bar{l}_n \in M}{C := l_1 \vee \cdots \vee l_n}$$

Explain $$\frac{C = l \vee D \quad l_1 \vee \cdots \vee l_n \vee \bar{l} \in F \quad \bar{l}_1, \ldots, \bar{l}_n \prec_M \bar{l}}{C := l_1 \vee \cdots \vee l_n \vee D}$$

Backjump $$\frac{C = l_1 \vee \cdots \vee l_n \vee l \quad \text{lev}\,\bar{l}_1, \ldots, \text{lev}\,\bar{l}_n \leq \ i < \text{lev}\,\bar{l}}{C := \text{no} \quad M := M^{[i]}\, l}$$

Note: $l \prec_M l'$ if l occurs before l' in M
$\text{lev}\, l = i$ iff l occurs in decision level i of M

Replace **Backtrack** with

Conflict $$\frac{\mathsf{C} = \mathsf{no} \quad l_1 \vee \cdots \vee l_n \in \mathsf{F} \quad \bar{l}_1, \ldots, \bar{l}_n \in \mathsf{M}}{\mathsf{C} := l_1 \vee \cdots \vee l_n}$$

Explain $$\frac{\mathsf{C} = l \vee D \quad l_1 \vee \cdots \vee l_n \vee \bar{l} \in \mathsf{F} \quad \bar{l}_1, \ldots, \bar{l}_n \prec_{\mathsf{M}} \bar{l}}{\mathsf{C} := l_1 \vee \cdots \vee l_n \vee D}$$

Backjump $$\frac{\mathsf{C} = l_1 \vee \cdots \vee l_n \vee l \quad \mathsf{lev}\,\bar{l}_1, \ldots, \mathsf{lev}\,\bar{l}_n \leq \; i < \mathsf{lev}\,\bar{l}}{\mathsf{C} := \mathsf{no} \quad \mathsf{M} := \mathsf{M}^{[i]}\, l}$$

Maintain invariant: $\mathsf{F} \models_{\mathrm{p}} \mathsf{C}$ and $\mathsf{M} \models_{\mathrm{p}} \neg\mathsf{C}$ when $\mathsf{C} \neq \mathsf{no}$

Note: $\models_{\mathrm{p}}$ denotes propositional entailment

Modify Fail to

Modify **Fail** to

$$\textbf{Fail}\ \frac{C \neq \mathrm{no} \quad \bullet \notin M}{\mathrm{fail}}$$

EXECUTION EXAMPLE

$$F := \{1,\ \ \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ \overline{1} \lor \overline{5} \lor 7,\ \ \overline{2} \lor \overline{5} \lor 6 \lor \overline{7}\}$$

M	F	C	rule
	F	no	

EXECUTION EXAMPLE

$F := \{1,\ \ \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ \overline{1} \vee \overline{5} \vee 7,\ \ \overline{2} \vee \overline{5} \vee 6 \vee \overline{7}\}$

M	F	C	rule
	F	no	
1	F	no	by Propagate

EXECUTION EXAMPLE

$$F := \{1,\ \ \bar{1} \vee 2,\ \ \bar{3} \vee 4,\ \ \bar{5} \vee \bar{6},\ \ \bar{1} \vee \bar{5} \vee 7,\ \ \bar{2} \vee \bar{5} \vee 6 \vee \bar{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by Propagate
1 2	F	no	by Propagate

EXECUTION EXAMPLE

$$F := \{1,\ \ \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ \overline{1} \vee \overline{5} \vee 7,\ \ \overline{2} \vee \overline{5} \vee 6 \vee \overline{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by **Propagate**
1 2	F	no	by **Propagate**
1 2 $\bullet$ 3	F	no	by **Decide**

EXECUTION EXAMPLE

$$F := \{1,\ \ \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ \overline{1} \lor \overline{5} \lor 7,\ \ \overline{2} \lor \overline{5} \lor 6 \lor \overline{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by Propagate
$1\ 2$	F	no	by Propagate
$1\ 2 \bullet 3$	F	no	by Decide
$1\ 2 \bullet 3\ 4$	F	no	by Propagate

$$F := \{1,\ \ \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ \overline{1} \vee \overline{5} \vee 7,\ \ \overline{2} \vee \overline{5} \vee 6 \vee \overline{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by Propagate
$1\ 2$	F	no	by Propagate
$1\ 2 \bullet 3$	F	no	by Decide
$1\ 2 \bullet 3\ 4$	F	no	by Propagate
$1\ 2 \bullet 3\ 4 \bullet 5$	F	no	by Decide

EXECUTION EXAMPLE

$$F := \{1,\ \ \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ \overline{1} \vee \overline{5} \vee 7,\ \ \overline{2} \vee \overline{5} \vee 6 \vee \overline{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by Propagate
$1\ 2$	F	no	by Propagate
$1\ 2 \bullet 3$	F	no	by Decide
$1\ 2 \bullet 3\ 4$	F	no	by Propagate
$1\ 2 \bullet 3\ 4 \bullet 5$	F	no	by Decide
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}$	F	no	by Propagate

EXECUTION EXAMPLE

$$F := \{1,\ \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ \overline{1} \vee \overline{5} \vee 7,\ \overline{2} \vee \overline{5} \vee 6 \vee \overline{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by Propagate
$1\ 2$	F	no	by Propagate
$1\ 2 \bullet 3$	F	no	by Decide
$1\ 2 \bullet 3\ 4$	F	no	by Propagate
$1\ 2 \bullet 3\ 4 \bullet 5$	F	no	by Decide
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}$	F	no	by Propagate
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	no	by Propagate

$$F := \{1,\ \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ \overline{1} \vee \overline{5} \vee 7,\ \overline{2} \vee \overline{5} \vee 6 \vee \overline{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by **Propagate**
$1\ 2$	F	no	by **Propagate**
$1\ 2 \bullet 3$	F	no	by **Decide**
$1\ 2 \bullet 3\ 4$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5$	F	no	by **Decide**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	$\overline{2} \vee \overline{5} \vee 6 \vee \overline{7}$	by **Conflict**

$$F := \{1,\ \bar{1} \vee 2,\ \bar{3} \vee 4,\ \bar{5} \vee \bar{6},\ \bar{1} \vee \bar{5} \vee 7,\ \bar{2} \vee \bar{5} \vee 6 \vee \bar{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by **Propagate**
$1\ 2$	F	no	by **Propagate**
$1\ 2 \bullet 3$	F	no	by **Decide**
$1\ 2 \bullet 3\ 4$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5$	F	no	by **Decide**
$1\ 2 \bullet 3\ 4 \bullet 5\ \bar{6}$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5\ \bar{6}\ 7$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5\ \bar{6}\ 7$	F	$\bar{2} \vee \bar{5} \vee 6 \vee \bar{7}$	by **Conflict**
$1\ 2 \bullet 3\ 4 \bullet 5\ \bar{6}\ 7$	F	$1 \vee \bar{2} \vee \bar{5} \vee 6$	by **Explain** with $\bar{1} \vee \bar{5} \vee 7$

$$F := \{1,\ \overline{1} \lor 2,\ \overline{3} \lor 4,\ \overline{5} \lor \overline{6},\ \overline{1} \lor \overline{5} \lor 7,\ \overline{2} \lor \overline{5} \lor 6 \lor \overline{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by **Propagate**
$1\ 2$	F	no	by **Propagate**
$1\ 2 \bullet 3$	F	no	by **Decide**
$1\ 2 \bullet 3\ 4$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5$	F	no	by **Decide**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	$\overline{2} \lor \overline{5} \lor 6 \lor \overline{7}$	by **Conflict**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	$1 \lor \overline{2} \lor \overline{5} \lor 6$	by **Explain** with $\overline{1} \lor \overline{5} \lor 7$
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	$1 \lor \overline{2} \lor \overline{5}$	by **Explain** with $\overline{5} \lor \overline{6}$

$$F := \{1,\ \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ \overline{1} \vee \overline{5} \vee 7,\ \overline{2} \vee \overline{5} \vee 6 \vee \overline{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by **Propagate**
$1\ 2$	F	no	by **Propagate**
$1\ 2 \bullet 3$	F	no	by **Decide**
$1\ 2 \bullet 3\ 4$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5$	F	no	by **Decide**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	$\overline{2} \vee \overline{5} \vee 6 \vee \overline{7}$	by **Conflict**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	$1 \vee \overline{2} \vee \overline{5} \vee 6$	by **Explain** with $\overline{1} \vee \overline{5} \vee 7$
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	$1 \vee \overline{2} \vee \overline{5}$	by **Explain** with $\overline{5} \vee \overline{6}$
$1\ 2\ \overline{5}$	F	no	by **Backjump**

EXECUTION EXAMPLE

$$F := \{1,\ \ \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ \overline{1} \vee \overline{5} \vee 7,\ \ \overline{2} \vee \overline{5} \vee 6 \vee \overline{7}\}$$

M	F	C	rule
	F	no	
1	F	no	by **Propagate**
$1\ 2$	F	no	by **Propagate**
$1\ 2 \bullet 3$	F	no	by **Decide**
$1\ 2 \bullet 3\ 4$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5$	F	no	by **Decide**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	no	by **Propagate**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	$\overline{2} \vee \overline{5} \vee 6 \vee \overline{7}$	by **Conflict**
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	$1 \vee \overline{2} \vee \overline{5} \vee 6$	by **Explain** with $\overline{1} \vee \overline{5} \vee 7$
$1\ 2 \bullet 3\ 4 \bullet 5\ \overline{6}\ 7$	F	$1 \vee \overline{2} \vee \overline{5}$	by **Explain** with $\overline{5} \vee \overline{6}$
$1\ 2\ \overline{5}$	F	no	by **Backjump**
$1\ 2\ \overline{5} \bullet 3$	F	no	by **Decide**
$\ldots$			

Also add

$$\textbf{Learn}\quad \frac{F \models_{\mathrm{p}} C \qquad C \notin F}{F := F \cup \{C\}}$$

$$\textbf{Forget}\quad \frac{C = \mathrm{no} \qquad F = G \cup \{C\} \qquad G \models_{\mathrm{p}} C}{F := G}$$

$$\textbf{Restart}\quad \frac{}{M := M^{[0]} \qquad C := \mathrm{no}}$$

Note: Learn can be applied to any clause stored in C when $C \neq \mathrm{no}$

At the core, current CDCL SAT solvers are implementations of the transition system with rules

Propagate, Decide,

Conflict, Explain, Backjump,

Learn, Forget, Restart

At the core, current CDCL SAT solvers are implementations of the transition system with rules

Propagate, **Decide**,

Conflict, **Explain**, **Backjump**,

Learn, **Forget**, **Restart**

Basic DPLL $\stackrel{\text{def}}{=}$

{ **Propagate**, **Decide**, **Conflict**, **Explain**, **Backjump** }

DPLL $\stackrel{\text{def}}{=}$ Basic DPLL + { **Learn**, **Forget**, **Restart** }

Some terminology:

Irreducible state: state for which no Basic DPLL rules apply

Execution: sequence of transitions allowed by the rules and starting with $M = \emptyset$ and $C = no$

Exhausted execution: execution ending in an irreducible state

Some terminology:

Irreducible state: state for which no Basic DPLL rules apply

Execution: sequence of transitions allowed by the rules and starting with $M = \emptyset$ and $C = \mathrm{no}$

Exhausted execution: execution ending in an irreducible state

Proposition (Strong Termination) Every execution in Basic DPLL is finite.

Note: This is not so immediate, because of **Backjump**.

Some terminology:

Irreducible state: state for which no Basic DPLL rules apply

Execution: sequence of transitions allowed by the rules and starting with $M = \emptyset$ and $C = \text{no}$

Exhausted execution: execution ending in an irreducible state

Proposition (Strong Termination) Every execution in Basic DPLL is finite.

Lemma Every exhausted execution ends with either $C = \text{no}$ or fail.

Some terminology:

Irreducible state: state for which no Basic DPLL rules apply

Execution: sequence of transitions allowed by the rules and starting with $\mathsf{M} = \emptyset$ and $\mathsf{C} = \mathsf{no}$

Exhausted execution: execution ending in an irreducible state

Proposition (Soundness) For every exhausted execution starting with $\mathsf{F} = F_0$ and ending with fail, the clause set F_0 is unsatisfiable.

Proposition (Completeness) For every exhausted execution starting with $\mathsf{F} = F_0$ and ending with $\mathsf{C} = \mathsf{no}$, the clause set F_0 is satisfied by M.

- Applying
  - one Basic DPLL rule between each two **Learn** applications and
  - **Restart** less and less often

  ensures termination

- Applying
  - one Basic DPLL rule between each two **Learn** applications and
  - **Restart** less and less often

  ensures termination

- A common basic strategy applies the rules with the following priorities:
  1. If $n > 0$ conflicts have been found so far,
     increase n and apply **Restart**

- Applying
  - one Basic DPLL rule between each two **Learn** applications and
  - **Restart** less and less often

  ensures termination

- A common basic strategy applies the rules with the following priorities:
  1. If $n > 0$ conflicts have been found so far,
     increase n and apply **Restart**
  2. If a clause is falsified by M, apply **Conflict**

- Applying
  - one Basic DPLL rule between each two **Learn** applications and
  - **Restart** less and less often

  ensures termination

- A common basic strategy applies the rules with the following priorities:
  1. If $n > 0$ conflicts have been found so far, increase n and apply **Restart**
  2. If a clause is falsified by M, apply **Conflict**
  3. Keep applying **Explain** until **Backjump** is applicable

- Applying
  - one Basic DPLL rule between each two **Learn** applications and
  - **Restart** less and less often

  ensures termination

- A common basic strategy applies the rules with the following priorities:
  1. If $n > 0$ conflicts have been found so far,
     increase n and apply **Restart**
  2. If a clause is falsified by M, apply **Conflict**
  3. Keep applying **Explain** until **Backjump** is applicable
  4. Apply **Learn**

- Applying
  - one Basic DPLL rule between each two **Learn** applications and
  - **Restart** less and less often

  ensures termination

- A common basic strategy applies the rules with the following priorities:
  1. If $n > 0$ conflicts have been found so far,
     increase n and apply **Restart**
  2. If a clause is falsified by M, apply **Conflict**
  3. Keep applying **Explain** until **Backjump** is applicable
  4. Apply **Learn**
  5. Apply **Backjump**

- Applying
  - one Basic DPLL rule between each two **Learn** applications and
  - **Restart** less and less often

  ensures termination

- A common basic strategy applies the rules with the following priorities:
  1. If $n > 0$ conflicts have been found so far,
     increase n and apply **Restart**
  2. If a clause is falsified by M, apply **Conflict**
  3. Keep applying **Explain** until **Backjump** is applicable
  4. Apply **Learn**
  5. Apply **Backjump**
  6. Apply **Propagate** to completion

- Applying
  - one Basic DPLL rule between each two **Learn** applications and
  - **Restart** less and less often

  ensures termination

- A common basic strategy applies the rules with the following priorities:
  1. If $n > 0$ conflicts have been found so far,
     increase n and apply **Restart**
  2. If a clause is falsified by M, apply **Conflict**
  3. Keep applying **Explain** until **Backjump** is applicable
  4. Apply **Learn**
  5. Apply **Backjump**
  6. Apply **Propagate** to completion
  7. Apply **Decide**

Same states and transitions but

- F contains quantifier-free clauses in some theory T
- M is a sequence of theory literals and decision points
- the DPLL system is augmented with rules

  T-**Conflict**, T-**Propagate**, T-**Explain**

- maintains invariant: $F \models_T C$ and $M \models_{\mathrm{p}} \neg C$ when $C \neq \mathrm{no}$

Def. $F \models_T G$ iff every model of T that satisfies F satisfies G as well

Fix a theory T

$$T\text{-}\mathbf{Conflict}\ \frac{C = \mathrm{no} \quad l_1, \ldots, l_n \in M \quad l_1, \ldots, l_n \models_T \bot}{C := \bar{l}_1 \vee \cdots \vee \bar{l}_n}$$

Fix a theory T

$$T\text{-Conflict}\quad \frac{\mathsf{C} = \mathsf{no} \quad l_1, \ldots, l_n \in \mathsf{M} \quad l_1, \ldots, l_n \models_T \bot}{\mathsf{C} := \bar{l}_1 \vee \cdots \vee \bar{l}_n}$$

$$T\text{-Propagate}\quad \frac{l \in \mathrm{Lit}(\mathsf{F}) \quad \mathsf{M} \models_T l \quad l, \bar{l} \notin \mathsf{M}}{\mathsf{M} := \mathsf{M}\ l}$$

Fix a theory T

$$T\text{-Conflict}\quad \frac{C = \mathrm{no} \quad l_1, \ldots, l_n \in M \quad l_1, \ldots, l_n \models_T \bot}{C := \bar{l}_1 \vee \cdots \vee \bar{l}_n}$$

$$T\text{-Propagate}\quad \frac{l \in \mathrm{Lit}(F) \quad M \models_T l \quad l, \bar{l} \notin M}{M := M\ l}$$

$$T\text{-Explain}\quad \frac{C = l \vee D \quad \bar{l}_1, \ldots, \bar{l}_n \models_T \bar{l} \quad \bar{l}_1, \ldots, \bar{l}_n \prec_M \bar{l}}{C := l_1 \vee \cdots \vee l_n \vee D}$$

Note: $\bot$ = empty clause

Note: $\models_T$ decided by theory solver

T-**Conflict** is enough to model the naive integration of SAT solvers and theory solvers seen in the earlier UF example

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\,\overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^{+}$

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\,\overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^+$
$1\,\overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^{+}$
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2 \vee 4$	by ***T*-Conflict**

MODELING THE VERY LAZY THEORY APPROACH

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\,\overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^{+}$
$1\,\overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\,\overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2 \vee 4$	by ***T*-Conflict**
$1\,\overline{4} \bullet \overline{2}$	$,\ \overline{1} \vee 2 \vee 4$	$\overline{1} \vee 2 \vee 4$	by **Learn**

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\,\overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^+$
$1\,\overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\,\overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2 \vee 4$	by ***T*-Conflict**
$1\,\overline{4} \bullet \overline{2}$	$,\ \overline{1} \vee 2 \vee 4$	$\overline{1} \vee 2 \vee 4$	by **Learn**
$1\,\overline{4}$	$,\ \overline{1} \vee 2 \vee 4$	no	by **Restart**

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^+$
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2 \vee 4$	by ***T*-Conflict**
$1\ \overline{4} \bullet \overline{2}$	$,\ \overline{1} \vee 2 \vee 4$	$\overline{1} \vee 2 \vee 4$	by **Learn**
$1\ \overline{4}$	$,\ \overline{1} \vee 2 \vee 4$	no	by **Restart**
$1\ \overline{4}\ 2\ 3$	$,\ \overline{1} \vee 2 \vee 4$	no	by **Propagate**$^+$

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\,\overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^+$
$1\,\overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\,\overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2 \vee 4$	by ***T*-Conflict**
$1\,\overline{4} \bullet \overline{2}$	$,\ \overline{1} \vee 2 \vee 4$	$\overline{1} \vee 2 \vee 4$	by **Learn**
$1\,\overline{4}$	$,\ \overline{1} \vee 2 \vee 4$	no	by **Restart**
$1\,\overline{4}\,2\,3$	$,\ \overline{1} \vee 2 \vee 4$	no	by **Propagate**$^+$
$1\,\overline{4}\,2\,3$	$,\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee \overline{3} \vee 4$	$\overline{1} \vee \overline{3} \vee 4$	by ***T*-Conflict**, **Learn**

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^{+}$
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2 \vee 4$	by ***T*-Conflict**
$1\ \overline{4} \bullet \overline{2}$	$,\ \overline{1} \vee 2 \vee 4$	$\overline{1} \vee 2 \vee 4$	by **Learn**
$1\ \overline{4}$	$,\ \overline{1} \vee 2 \vee 4$	no	by **Restart**
$1\ \overline{4}\ 2\ 3$	$,\ \overline{1} \vee 2 \vee 4$	no	by **Propagate**$^{+}$
$1\ \overline{4}\ 2\ 3$	$,\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee \overline{3} \vee 4$	$\overline{1} \vee \overline{3} \vee 4$	by ***T*-Conflict**, **Learn**
fail			by **Fail**

The very lazy approach can be improved considerably with

- An *on-line* SAT engine,
  which can accept new input clauses on the fly

The very lazy approach can be improved considerably with

- An *on-line* SAT engine,
  which can accept new input clauses on the fly
- an *incremental and explicating* T-solver,
  which can

The very lazy approach can be improved considerably with

- An *on-line* SAT engine,
  which can accept new input clauses on the fly
- an *incremental and explicating* T-solver,
  which can
  1. check the T-satisfiability of M as it is extended and

The very lazy approach can be improved considerably with

- An *on-line* SAT engine,
  which can accept new input clauses on the fly
- an *incremental and explicating* T-solver,
  which can
  1. check the T-satisfiability of M as it is extended and
  2. identify a small T-unsatisfiable subset of M once M becomes T-unsatisfiable

A BETTER LAZY APPROACH

$$\underbrace{g(a) = c}_{1} \quad \land \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \lor \underbrace{g(a) = d}_{3} \quad \land \quad \underbrace{c \neq d}_{\overline{4}}$$

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by Propagate^{+}

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\,\overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^{+}$
$1\,\overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\,\overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^{+}$
$1\,\overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\,\overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2$	by ***T*-Conflict**

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^{+}$
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2$	by ***T*-Conflict**
$1\ \overline{4}\ 2$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Backjump**

A BETTER LAZY APPROACH

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^{+}$
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2$	by ***T*-Conflict**
$1\ \overline{4}\ 2$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Backjump**
$1\ \overline{4}\ 2\ 3$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^+$
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2$	by ***T*-Conflict**
$1\ \overline{4}\ 2$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Backjump**
$1\ \overline{4}\ 2\ 3$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**
$1\ \overline{4}\ 2\ 3$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee \overline{3} \vee 4$	by ***T*-Conflict**

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^+$
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Decide**
$1\ \overline{4} \bullet \overline{2}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee 2$	by ***T*-Conflict**
$1\ \overline{4}\ 2$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Backjump**
$1\ \overline{4}\ 2\ 3$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**
$1\ \overline{4}\ 2\ 3$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{1} \vee \overline{3} \vee 4$	by ***T*-Conflict**
fail			by **Fail**

Ignoring **Restart** (for simplicity), a common strategy is to apply the rules using the following priorities:

1. If a clause is falsified by the current assignment M, apply **Conflict**
2. If M is $\mathcal{T}$-unsatisfiable, apply **$\mathcal{T}$-Conflict**
3. Apply **Fail** or **Explain**+**Learn**+**Backjump** as appropriate
4. Apply **Propagate**
5. Apply **Decide**

Ignoring **Restart** (for simplicity), a common strategy is to apply the rules using the following priorities:

1. If a clause is falsified by the current assignment M, apply **Conflict**
2. If M is T-unsatisfiable, apply **T-Conflict**
3. Apply **Fail** or **Explain**+**Learn**+**Backjump** as appropriate
4. Apply **Propagate**
5. Apply **Decide**

Note: Depending on the cost of checking the T-satisfiability of M, Step (2) can be applied with lower frequency or priority

With *T*-**Conflict** as the only theory rule, the theory solver is used just to validate the choices of the SAT engine

With *T*-**Conflict** as the only theory rule, the theory solver is used just to validate the choices of the SAT engine

With *T*-**Propagate** and *T*-**Explain**, it can also be used to guide the engine's search [Tin02]

T-**Propagate** $$\frac{l \in \mathrm{Lit}(F) \quad M \models_T l \quad l, \bar{l} \notin M}{M := M\ l}$$

T-**Explain** $$\frac{C = l \vee D \quad \bar{l}_1, \ldots, \bar{l}_n \models_T \bar{l} \quad \bar{l}_1, \ldots, \bar{l}_n \prec_M \bar{l}}{C := l_1 \vee \cdots \vee l_n \vee D}$$

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by Propagate$^+$

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by Propagate$^+$
$1\ \overline{4}\ 2$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by T-Propagate ($1 \models_T 2$)

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^{+}$
$1\ \overline{4}\ 2$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by ***T*-Propagate** $(1 \models_T 2)$
$1\ \overline{4}\ 2\ \overline{3}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by ***T*-Propagate** $(1,\ \overline{4} \models_T \overline{3})$

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^+$
$1\ \overline{4}\ 2$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by T-**Propagate** ($1 \models_T 2$)
$1\ \overline{4}\ 2\ \overline{3}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by T-**Propagate** ($1,\ \overline{4} \models_T \overline{3}$)
$1\ \overline{4}\ 2\ \overline{3}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{2} \vee 3$	by **Conflict**

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

M	F	C	rule
	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	
$1\ \overline{4}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by **Propagate**$^+$
$1\ \overline{4}\ 2$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by ***T*-Propagate** $(1 \models_T 2)$
$1\ \overline{4}\ 2\ \overline{3}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	no	by ***T*-Propagate** $(1,\ \overline{4} \models_T \overline{3})$
$1\ \overline{4}\ 2\ \overline{3}$	$1,\ \overline{2} \vee 3,\ \overline{4}$	$\overline{2} \vee 3$	by **Conflict**
fail			by **Fail**

Note: *T*-propagation eliminates search altogether in this case
no applications of **Decide** are needed

At the core, current lazy SMT solvers are implementations of the transition system with rules

(1) **Propagate**, **Decide**, **Conflict**, **Explain**, **Backjump**, **Fail**

(2) T-**Conflict**, T-**Propagate**, T-**Explain**

(3) **Learn**, **Forget**, **Restart**

At the core, current lazy SMT solvers are implementations of the transition system with rules

(1) **Propagate**, **Decide**, **Conflict**, **Explain**, **Backjump**, **Fail**

(2) **T-Conflict**, **T-Propagate**, **T-Explain**

(3) **Learn**, **Forget**, **Restart**

Basic DPLL Modulo Theories $\stackrel{\text{def}}{=}$ (1) + (2)

DPLL Modulo Theories $\stackrel{\text{def}}{=}$ (1) + (2) + (3)

Updated terminology:

Irreducible state: state to which no Basic DPLL MT rules apply

Execution: sequence of transitions allowed by the rules and starting with $M = \emptyset$ and $C = no$

Exhausted execution: execution ending in an irreducible state

Updated terminology:

Irreducible state: state to which no Basic DPLL MT rules apply

Execution: sequence of transitions allowed by the rules and starting with $M = \emptyset$ and $C = no$

Exhausted execution: execution ending in an irreducible state

Proposition (Termination) Every execution in which

(a) **Learn**/**Forget** are applied only finitely many times and

(b) **Restart** is applied with increased periodicity

is finite.

Lemma Every exhausted execution ends with either $C = no$ or fail.

Updated terminology:

Irreducible state: state to which no Basic DPLL MT rules apply

Execution: sequence of transitions allowed by the rules and starting with $\mathsf{M} = \emptyset$ and $\mathsf{C} = \mathsf{no}$

Exhausted execution: execution ending in an irreducible state

Proposition (Soundness) For every exhausted execution starting with $\mathsf{F} = F_0$ and ending with fail, the clause set F_0 is T-unsatisfiable.

Proposition (Completeness) For every exhausted execution starting with $\mathsf{F} = F_0$ and ending with $\mathsf{C} = \mathsf{no}$, F_0 is T-satisfiable; specifically, M is T-satisfiable and $\mathsf{M} \models_{\mathrm{p}} F_0$.

The approach formalized so far can be implemented with a simple architecture named DPLL(T) [GHN+04, NOT06]

DPLL(T) = DPLL(X) engine + T-solver

The approach formalized so far can be implemented with a simple architecture named DPLL(T) [GHN+04, NOT06]

$$\text{DPLL}(T) = \text{DPLL}(X) \text{ engine} + T\text{-solver}$$

DPLL(X):

- Very similar to a SAT solver, enumerates Boolean models
- Not allowed: pure literal, blocked literal detection, ...
- Required: incremental addition of clauses
- Desirable: partial model detection

The approach formalized so far can be implemented with a simple architecture named DPLL(T) [GHN+04, NOT06]

$$\text{DPLL}(T) = \text{DPLL}(X) \text{ engine} + T\text{-solver}$$

T-solver:

- Checks the T-satisfiability of conjunctions of literals
- Computes theory propagations
- Produces explanations of T-unsatisfiability/propagation
- Must be incremental and backtrackable

For certain theories, determining that a set M is T-unsatisfiable requires reasoning by cases.

For certain theories, determining that a set M is T-unsatisfiable requires reasoning by cases.

Example: T = the theory of arrays.

$$M = \{\, \underbrace{r(w(a,i,x),j) \neq x}_{1},\ \underbrace{r(w(a,i,x),j) \neq r(a,j)}_{2} \,\}$$

For certain theories, determining that a set M is T-unsatisfiable requires reasoning by cases.

Example: T = the theory of arrays.

$$M = \{\, \underbrace{r(w(a,i,x),j) \neq x}_{1},\ \underbrace{r(w(a,i,x),j) \neq r(a,j)}_{2} \,\}$$

$i = j$) Then, $r(w(a,i,x),j) = x$. Contradiction with 1.

For certain theories, determining that a set M is T-unsatisfiable requires reasoning by cases.

Example: T = the theory of arrays.

$$M = \{\, \underbrace{r(w(a,i,x),j) \neq x}_{1},\ \underbrace{r(w(a,i,x),j) \neq r(a,j)}_{2} \,\}$$

$i = j$) Then, $r(w(a,i,x),j) = x$. Contradiction with 1.

$i \neq j$) Then, $r(w(a,i,x),j) = r(a,j)$. Contradiction with 2.

For certain theories, determining that a set M is T-unsatisfiable requires reasoning by cases.

Example: T = the theory of arrays.

$$M = \{\, \underbrace{r(w(a,i,x),j) \neq x}_{1},\ \underbrace{r(w(a,i,x),j) \neq r(a,j)}_{2} \,\}$$

$i = j$) Then, $r(w(a,i,x),j) = x$. Contradiction with 1.

$i \neq j$) Then, $r(w(a,i,x),j) = r(a,j)$. Contradiction with 2.

Conclusion: M is T-unsatisfiable

A *complete* T-solver reasons by cases via (internal) case splitting and backtracking mechanisms

A *complete* T-solver reasons by cases via (internal) case splitting and backtracking mechanisms

An alternative is to lift case splitting and backtracking from the T-solver to the SAT engine

A *complete* T-solver reasons by cases via (internal) case splitting and backtracking mechanisms

An alternative is to lift case splitting and backtracking from the T-solver to the SAT engine

Basic idea: encode case splits as sets of clauses and send them as needed to the SAT engine for it to split on them [BNOT06]

A *complete* T-solver reasons by cases via (internal) case splitting and backtracking mechanisms

An alternative is to lift case splitting and backtracking from the T-solver to the SAT engine

Basic idea: encode case splits as sets of clauses and send them as needed to the SAT engine for it to split on them [BNOT06]

Possible benefits:

- All case-splitting is coordinated by the SAT engine
- Only have to implement case-splitting infrastructure in one place
- Can learn a wider class of lemmas

Basic idea: encode case splits as a set of clauses and send them as needed to the SAT engine for it to split on them

Basic idea: encode case splits as a set of clauses and send them as needed to the SAT engine for it to split on them

Basic Scenario:

$$M = \{\ldots,\ s = \underbrace{r(w(a,i,t),j)}_{s'},\ \ldots\}$$

- Main SMT module: "Is M T-unsatisfiable?"

Basic idea: encode case splits as a set of clauses and send them as needed to the SAT engine for it to split on them

Basic Scenario:

$$M = \{\ldots,\ s = \underbrace{r(w(a,i,t),j)}_{s'},\ \ldots\}$$

- Main SMT module: "Is M T-unsatisfiable?"
- T-solver: "I do not know yet, but it will help me if you consider these *theory lemmas*:

$$s = s' \land i = j \Rightarrow s = t, \quad s = s' \land i \neq j \Rightarrow s = r(a,j) \text{ ''}$$

To model the generation of theory lemmas for case splits, add the rule

T-Learn

$$\frac{\models_T \exists \mathbf{v}(l_1 \vee \cdots \vee l_n) \quad l_1, \ldots, l_n \in L_{\mathrm{S}} \quad \mathbf{v} \text{ vars not in F}}{\mathrm{F} := \mathrm{F} \cup \{l_1 \vee \cdots \vee l_n\}}$$

where L_{S} is a finite set of literals dependent on the initial set of clauses (see [BNOT06] for a formal definition of L_{S})

To model the generation of theory lemmas for case splits, add the rule

T-Learn

$$\frac{\models_T \exists \mathbf{v}(l_1 \vee \cdots \vee l_n) \quad l_1, \ldots, l_n \in L_{\mathrm{S}} \quad \mathbf{v} \text{ vars not in F}}{\mathrm{F} := \mathrm{F} \cup \{l_1 \vee \cdots \vee l_n\}}$$

where L_{S} is a finite set of literals dependent on the initial set of clauses (see [BNOT06] for a formal definition of L_{S})

Note: For many theories with a theory solver, there exists an appropriate finite L_{S} for every input F

The set L_{S} does not need to be computed explicitly

Now we can relax the requirement on the theory solver:

When $M \models_{\mathrm{p}} F$, it must either

- *determine whether $M \models_{T} \perp$ or*
- *generate a new clause by T-**Learn** containing at least one literal of L_{S} undefined in M*

Now we can relax the requirement on the theory solver:

When $M \models_p F$, it must either

- *determine whether $M \models_T \bot$ or*
- *generate a new clause by T-**Learn** containing at least one literal of L_S undefined in M*

The T-solver is required to determine whether $M \models_T \bot$ only if all literals in L_S are defined in M

Now we can relax the requirement on the theory solver:

When $M \models_{\mathrm{p}} F$, it must either

- *determine whether $M \models_T \bot$ or*
- *generate a new clause by T-**Learn** containing at least one literal of L_{S} undefined in M*

The T-solver is required to determine whether $M \models_T \bot$ only if all literals in L_{S} are defined in M

Note: In practice, to determine if $M \models_T \bot$, the T-solver only needs a small subset of L_{S} to be defined in M

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by Propagate$^+$

EXAMPLE — THEORY OF FINITE SETS

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by Propagate$^+$
$x = y \cup z \;\bullet\; y = \emptyset$	F	by Decide

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by Propagate$^+$
$x = y \cup z \ \bullet\ y = \emptyset$	F	by Decide
$x = y \cup z \ \bullet\ y = \emptyset\ x \neq z$	F	by Propagate

EXAMPLE — THEORY OF FINITE SETS

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by **Propagate$^+$**
$x = y \cup z \bullet y = \emptyset$	F	by **Decide**
$x = y \cup z \bullet y = \emptyset\ x \neq z$	F	by **Propagate**
$x = y \cup z \bullet y = \emptyset\ x \neq z$	$F, (x = z \vee e \in x \vee e \in z),$	by ***T*-Learn**

EXAMPLE — THEORY OF FINITE SETS

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by Propagate$^+$
$x = y \cup z \ \bullet \ y = \emptyset$	F	by Decide
$x = y \cup z \ \bullet \ y = \emptyset \ x \neq z$	F	by Propagate
$x = y \cup z \ \bullet \ y = \emptyset \ x \neq z$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by T-Learn

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by Propagate$^+$
$x = y \cup z \;\bullet\; y = \emptyset$	F	by Decide
$x = y \cup z \;\bullet\; y = \emptyset\; x \neq z$	F	by Propagate
$x = y \cup z \;\bullet\; y = \emptyset\; x \neq z$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by T-Learn
$x = y \cup z \;\bullet\; y = \emptyset\; x \neq z \;\bullet\; e \in x$	$F, (x = z \vee e \in x \vee e \in z),$	by Decide

Example — Theory of finite sets

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by Propagate$^+$
$x = y \cup z \bullet y = \emptyset$	F	by Decide
$x = y \cup z \bullet y = \emptyset \; x \neq z$	F	by Propagate
$x = y \cup z \bullet y = \emptyset \; x \neq z$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by T-Learn
$x = y \cup z \bullet y = \emptyset \; x \neq z \bullet e \in x$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by Decide

EXAMPLE — THEORY OF FINITE SETS

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by Propagate$^+$
$x = y \cup z \;\bullet\; y = \emptyset$	F	by Decide
$x = y \cup z \;\bullet\; y = \emptyset\; x \neq z$	F	by Propagate
$x = y \cup z \;\bullet\; y = \emptyset\; x \neq z$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by T-Learn
$x = y \cup z \;\bullet\; y = \emptyset\; x \neq z \;\bullet\; e \in x$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by Decide
$x = y \cup z \;\bullet\; y = \emptyset\; x \neq z \;\bullet\; e \in x\; e \notin z$	$F, (x = z \vee e \in x \vee e \in z),$	by Propagate

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by Propagate$^+$
$x = y \cup z \bullet y = \emptyset$	F	by Decide
$x = y \cup z \bullet y = \emptyset\ x \neq z$	F	by Propagate
$x = y \cup z \bullet y = \emptyset\ x \neq z$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by T-Learn
$x = y \cup z \bullet y = \emptyset\ x \neq z \bullet e \in x$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by Decide
$x = y \cup z \bullet y = \emptyset\ x \neq z \bullet e \in x\ e \notin z$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by Propagate

EXAMPLE — THEORY OF FINITE SETS

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by Propagate$^+$
$x = y \cup z \bullet y = \emptyset$	F	by Decide
$x = y \cup z \bullet y = \emptyset\, x \neq z$	F	by Propagate
$x = y \cup z \bullet y = \emptyset\, x \neq z$	$F, (x = z \vee e \in x \vee e \in z), (x = z \vee e \notin x \vee e \notin z)$	by T-Learn
$x = y \cup z \bullet y = \emptyset\, x \neq z \bullet e \in x$	$F, (x = z \vee e \in x \vee e \in z), (x = z \vee e \notin x \vee e \notin z)$	by Decide
$x = y \cup z \bullet y = \emptyset\, x \neq z \bullet e \in x\, e \notin z$	$F, (x = z \vee e \in x \vee e \in z), (x = z \vee e \notin x \vee e \notin z)$	by Propagate

T-solver can make the following deductions at this point:

$$e \in x \;\cdots\; \Rightarrow e \in y \cup z \;\cdots\; \Rightarrow e \in y \;\cdots\; \Rightarrow e \in \emptyset \;\Rightarrow \perp$$

EXAMPLE — THEORY OF FINITE SETS

$$F: \quad x = y \cup z \quad \wedge \quad y \neq \emptyset \vee x \neq z$$

M	F	rule
$x = y \cup z$	F	by **Propagate**$^+$
$x = y \cup z \ \bullet \ y = \emptyset$	F	by **Decide**
$x = y \cup z \ \bullet \ y = \emptyset \ x \neq z$	F	by **Propagate**
$x = y \cup z \ \bullet \ y = \emptyset \ x \neq z$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by ***T*-Learn**
$x = y \cup z \ \bullet \ y = \emptyset \ x \neq z \ \bullet \ e \in x$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by **Decide**
$x = y \cup z \ \bullet \ y = \emptyset \ x \neq z \ \bullet \ e \in x \ e \notin z$	$F, (x = z \vee e \in x \vee e \in z),$ $(x = z \vee e \notin x \vee e \notin z)$	by **Propagate**

T-solver can make the following deductions at this point:

$$e \in x \quad \cdots \quad \Rightarrow \quad e \in y \cup z \quad \cdots \quad \Rightarrow \quad e \in y \quad \cdots \quad \Rightarrow \quad e \in \emptyset \quad \Rightarrow \quad \perp$$

This enables an application of ***T*-Conflict** with clause

$$x \neq y \cup z \ \vee \ y \neq \emptyset \ \vee \ x = z \ \vee \ e \notin x \ \vee \ e \in z$$

Correctness results can be extended to the new rule.

Soundness: The new T-**Learn** rule maintains satisfiability of the clause set.

Completeness: As long as the theory solver can decide $M \models_T \bot$ when all literals in L_S are determined, the system is still complete.

Termination: The system terminates under the same conditions as before. Roughly:

- Any lemma is (re)learned only finitely many times
- **Restart** is applied with increased periodicity

COMBINING THEORIES AND THEIR SOLVERS

Many applications give rise to mixed-theory formulas like:

$$a \approx b + 2 \wedge A = \mathrm{store}(B, a+1, 4) \wedge$$
$$A[b+3] = 2 \vee f(a-1) \neq f(b+1)$$

Many applications give rise to mixed-theory formulas like:

$$a \approx b + 2 \,\wedge\, A = \mathrm{store}(B, a+1, 4) \,\wedge\\ A[b+3] = 2 \,\vee\, f(a-1) \neq f(b+1)$$

Solving that formula requires reasoning over

- the theory of linear arithmetic (T_{LA})
- the theory of arrays (T_{A})
- the theory of uninterpreted functions (T_{UF})

Many applications give rise to mixed-theory formulas like:

$$a \approx b + 2 \wedge A = \mathrm{store}(B, a + 1, 4) \wedge$$
$$A[b + 3] = 2 \vee f(a - 1) \neq f(b + 1)$$

Solving that formula requires reasoning over

- the theory of linear arithmetic (T_{LA})
- the theory of arrays (T_{A})
- the theory of uninterpreted functions (T_{UF})

Question: Given solvers for each theory, can we combine them modularly into one for $T_{\mathrm{LA}} \cup T_{\mathrm{A}} \cup T_{\mathrm{UF}}$?

NEED FOR COMBINING THEORIES AND SOLVERS

Many applications give rise to mixed-theory formulas like:

$$a \approx b + 2 \;\wedge\; A = \mathrm{store}(B, a+1, 4) \;\wedge$$
$$A[b+3] = 2 \;\vee\; f(a-1) \neq f(b+1)$$

Solving that formula requires reasoning over

- the theory of linear arithmetic (T_{LA})
- the theory of arrays (T_{A})
- the theory of uninterpreted functions (T_{UF})

Question: Given solvers for each theory, can we combine them modularly into one for $T_{\mathrm{LA}} \cup T_{\mathrm{A}} \cup T_{\mathrm{UF}}$?

Under certain conditions, we can do it with the Nelson-Oppen combination method [NO79, Opp80]

Consider the following set of literals over $T_{\mathrm{LRA}} \cup T_{\mathrm{UF}}$ (T_{LRA}, linear real arithmetic):

$$
\begin{aligned}
f(f(x) - f(y)) &= a \\
f(0) &> a + 2 \\
x &= y
\end{aligned}
$$

Consider the following set of literals over $T_{\mathrm{LRA}} \cup T_{\mathrm{UF}}$ (T_{LRA}, linear real arithmetic):

$$\begin{aligned} f(f(x) - f(y)) &= a \\ f(0) &> a + 2 \\ x &= y \end{aligned}$$

First step: *purify* literals so that each belongs to a single theory

Consider the following set of literals over $T_{\mathrm{LRA}} \cup T_{\mathrm{UF}}$ (T_{LRA}, linear real arithmetic):

$$\begin{array}{rcl} f(f(x) - f(y)) & = & a \\ f(0) & > & a + 2 \\ x & = & y \end{array}$$

First step: *purify* literals so that each belongs to a single theory

$$f(f(x) - f(y)) = a \implies \begin{array}{r} f(e_1) = a \\ e_1 = f(x) - f(y) \end{array} \implies \begin{array}{r} f(e_1) = a \\ e_1 = e_2 - e_3 \\ e_2 = f(x) \\ e_3 = f(y) \end{array}$$

Consider the following set of literals over $T_{\mathrm{LRA}} \cup T_{\mathrm{UF}}$ (T_{LRA}, linear real arithmetic):

$$\begin{array}{rcl} f(f(x) - f(y)) & = & a \\ f(0) & > & a + 2 \\ x & = & y \end{array}$$

First step: *purify* literals so that each belongs to a single theory

$$f(0) > a + 2 \quad \Longrightarrow \quad \begin{array}{c} f(e_4) > a + 2 \\ e_4 = 0 \end{array} \quad \Longrightarrow \quad \begin{array}{l} f(e_4) = e_5 \\ e_4 = 0 \\ e_5 > a + 2 \end{array}$$

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	
$x = y$	

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	
$x = y$	

$$L_1 \models_{\mathrm{UF}} e_2 = e_3$$

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	

$$L_2 \models_{\mathrm{LRA}} e_1 = e_4$$

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	
$e_1 = e_4$	

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	
$e_1 = e_4$	

$$L_1 \models_{\mathrm{UF}} a = e_5$$

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	$a = e_5$
$e_1 = e_4$	

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	$a = e_5$
$e_1 = e_4$	

Third step: check for satisfiability locally

$L_1 \not\models_{\mathrm{UF}} \bot$

$L_2 \models_{\mathrm{LRA}} \bot$

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	$a = e_5$
$e_1 = e_4$	

Third step: check for satisfiability locally

$L_1 \not\models_{\mathrm{UF}} \perp$

$L_2 \models_{\mathrm{LRA}} \perp$

Report unsatisfiable

Consider the following unsatisfiable set of literals over $T_{\mathrm{LIA}} \cup T_{\mathrm{UF}}$ (T_{LIA}, linear integer arithmetic):

$$\begin{aligned} 1 \leq \quad x \quad & \leq 2 \\ f(1) \quad = \quad & a \\ f(2) \quad = \quad & f(1) + 3 \\ a \quad = \quad & b + 2 \end{aligned}$$

Consider the following unsatisfiable set of literals over $T_{\mathrm{LIA}} \cup T_{\mathrm{UF}}$ (T_{LIA}, linear integer arithmetic):

$$\begin{aligned} 1 \leq \quad x \quad & \leq 2 \\ f(1) \quad = \quad & a \\ f(2) \quad = \quad & f(1) + 3 \\ a \quad = \quad & b + 2 \end{aligned}$$

First step: *purify* literals so that each belongs to a single theory

Consider the following unsatisfiable set of literals over $T_{\mathrm{LIA}} \cup T_{\mathrm{UF}}$ (T_{LIA}, linear integer arithmetic):

$$\begin{aligned} 1 \leq \quad & x & \leq 2 \\ f(1) \quad & = & a \\ f(2) \quad & = & f(1) + 3 \\ a \quad & = & b + 2 \end{aligned}$$

First step: *purify* literals so that each belongs to a single theory

$$f(1) = a \quad \Longrightarrow \quad \begin{aligned} f(e_1) &= a \\ e_1 &= 1 \end{aligned}$$

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Consider the following unsatisfiable set of literals over $T_{\mathrm{LIA}} \cup T_{\mathrm{UF}}$ (T_{LIA}, linear integer arithmetic):

$$\begin{array}{rcl} 1 \leq & x & \leq 2 \\ f(1) & = & a \\ f(2) & = & f(1) + 3 \\ a & = & b + 2 \end{array}$$

First step: *purify* literals so that each belongs to a single theory

$$f(2) = f(1) + 3 \quad \Longrightarrow \quad \begin{array}{r} e_2 = 2 \\ f(e_2) = e_3 \\ f(e_1) = e_4 \\ e_3 = e_4 + 3 \end{array}$$

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

No more entailed equalities, but $L_1 \models_{\mathrm{LIA}} x = e_1 \lor x = e_2$

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

Consider each case of $x = e_1 \vee x = e_2$ separately

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

Case 1) $x = e_1$

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	$x = e_1$
$e_3 = e_4 + 3$	
$a = e_4$	
$x = e_1$	

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	$x = e_1$
$e_3 = e_4 + 3$	
$a = e_4$	
$x = e_1$	

$L_2 \models_{\mathrm{UF}} a = b$, which entails $\perp$ when sent to L_1

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

Case 2) $x = e_2$

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	$x = e_2$
$e_3 = e_4 + 3$	
$a = e_4$	
$x = e_2$	

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	$x = e_2$
$e_3 = e_4 + 3$	
$a = e_4$	
$x = e_2$	

$L_2 \models_{\mathrm{UF}} e_3 = b$, which entails $\perp$ when sent to L_1

- For $i = 1, 2$, let T_i be a first-order theory of *signature* Σ_i (set of function and predicate symbols in T_i other than $=$)
- Let $T = T_1 \cup T_2$
- Let $\mathcal{C}$ be a finite set of *free* constants (i.e., not in $\Sigma_1 \cup \Sigma_2$)

- For $i = 1, 2$, let T_i be a first-order theory of *signature* Σ_i (set of function and predicate symbols in T_i other than $=$)
- Let $T = T_1 \cup T_2$
- Let $\mathcal{C}$ be a finite set of *free* constants (i.e., not in $\Sigma_1 \cup \Sigma_2$)

We consider only input problems of the form

$$L_1 \cup L_2$$

where each L_i is a finite set of *ground* (i.e., variable-free) $(\Sigma_i \cup \mathcal{C})$-literals

- For $i = 1, 2$, let T_i be a first-order theory of *signature* Σ_i (set of function and predicate symbols in T_i other than $=$)
- Let $T = T_1 \cup T_2$
- Let $\mathcal{C}$ be a finite set of *free* constants (i.e., not in $\Sigma_1 \cup \Sigma_2$)

We consider only input problems of the form

$$L_1 \cup L_2$$

where each L_i is a finite set of *ground* (i.e., variable-free) $(\Sigma_i \cup \mathcal{C})$-literals

Note: Because of purification, there is no loss of generality in considering only ground $(\Sigma_i \cup \mathcal{C})$-literals

Bare-bones, non-deterministic, non-incremental version

[Opp80, Rin96, TH96]:

Bare-bones, non-deterministic, non-incremental version [Opp80, Rin96, TH96]:

Input: $L_1 \cup L_2$ with L_i finite set of ground $(\Sigma_i \cup \mathcal{C})$-literals
Output: sat or unsat

Bare-bones, non-deterministic, non-incremental version

[Opp80, Rin96, TH96]:

Input: $L_1 \cup L_2$ with L_i finite set of ground $(\Sigma_i \cup \mathcal{C})$-literals
Output: **sat** or **unsat**

1. Guess an *arrangement* A, i.e., a set of equalities and disequalities over $\mathcal{C}$ such that

$$c = d \in A \quad \text{or} \quad c \neq d \in A \quad \text{for all } c, d \in \mathcal{C}$$

Bare-bones, non-deterministic, non-incremental version [Opp80, Rin96, TH96]:

Input: $L_1 \cup L_2$ with L_i finite set of ground $(\Sigma_i \cup \mathcal{C})$-literals
Output: **sat** or **unsat**

1. Guess an *arrangement* A, i.e., a set of equalities and disequalities over $\mathcal{C}$ such that

$$c = d \in A \quad \text{or} \quad c \neq d \in A \quad \text{for all } c, d \in \mathcal{C}$$

2. If $L_i \cup A$ is T_i-unsatisfiable for $i = 1$ or $i = 2$, return unsat

Bare-bones, non-deterministic, non-incremental version

[Opp80, Rin96, TH96]:

Input: $L_1 \cup L_2$ with L_i finite set of ground $(\Sigma_i \cup \mathcal{C})$-literals
Output: **sat** or **unsat**

1. Guess an *arrangement* A, i.e., a set of equalities and disequalities over $\mathcal{C}$ such that

$$c = d \in A \quad \text{or} \quad c \neq d \in A \quad \text{for all } c, d \in \mathcal{C}$$

2. If $L_i \cup A$ is T_i-unsatisfiable for $i = 1$ or $i = 2$, return unsat
3. Otherwise, return **sat**

Proposition (Termination) The method is terminating.

(Trivially, because there is only a finite number of arrangements to guess)

Proposition (Termination) The method is terminating.

(Trivially, because there is only a finite number of arrangements to guess)

Proposition (Soundness) If the method returns **unsat** for every arrangement, the input is $(T_1 \cup T_2)$-unsatisfiable.

(Because satisfiability in $(T_1 \cup T_2)$ is always preserved)

Proposition (Termination) The method is terminating.

(Trivially, because there is only a finite number of arrangements to guess)

Proposition (Soundness) If the method returns **unsat** for every arrangement, the input is $(T_1 \cup T_2)$-unsatisfiable.

(Because satisfiability in $(T_1 \cup T_2)$ is always preserved)

Proposition (Completeness) If $\Sigma_1 \cap \Sigma_2 = \emptyset$ and T_1 and T_2 are stably infinite, when the method returns **sat** for some arrangement, the input is $(T_1 \cup T_2)$-is satisfiable.

Def. A theory T is *stably infinite* iff every quantifier-free T-satisfiable formula is satisfiable in an infinite model of T

Def. A theory T is *stably infinite* iff every quantifier-free T-satisfiable formula is satisfiable in an infinite model of T

Many interesting theories are stably infinite:

- Theories of an infinite structure (e.g., integer arithmetic)
- Complete theories with an infinite model (e.g., theory of dense linear orders, theory of lists)
- Convex theories (e.g., EUF, linear real arithmetic)

Def. A theory T is *stably infinite* iff every quantifier-free T-satisfiable formula is satisfiable in an infinite model of T

Many interesting theories are stably infinite:

- Theories of an infinite structure (e.g., integer arithmetic)
- Complete theories with an infinite model (e.g., theory of dense linear orders, theory of lists)
- Convex theories (e.g., EUF, linear real arithmetic)

Def. A theory T is *convex* iff, for any set L of literals

$L \models_T s_1 = t_1 \vee \cdots \vee s_n = t_n \implies L \models_T s_i = t_i$ for some i

Note: With convex theories, arrangements do not need to be guessed, theycan be computed by (theory) propagation

Def. A theory T is *stably infinite* iff every quantifier-free T-satisfiable formula is satisfiable in an infinite model of T

Def. A theory T is *stably infinite* iff every quantifier-free T-satisfiable formula is satisfiable in an infinite model of T

Other interesting theories are not stably infinite:

- Theories of a finite structure (e.g., theory of bit vectors of finite size, arithmetic modulo n)
- Theories with models of bounded cardinality (e.g., theory of strings of bounded length)
- Some equational/Horn theories

Def. A theory T is *stably infinite* iff every quantifier-free T-satisfiable formula is satisfiable in an infinite model of T

Other interesting theories are not stably infinite:

- Theories of a finite structure (e.g., theory of bit vectors of finite size, arithmetic modulo n)
- Theories with models of bounded cardinality (e.g., theory of strings of bounded length)
- Some equational/Horn theories

The Nelson-Oppen method has been extended to some classes of non-stably infinite theories [TZ05, RRZ05, JB10]

Let $T_1, \ldots, T_n$ be theories with respective solvers $S_1, \ldots, S_n$

How can we integrate all of them cooperatively into a single SMT solver for $T = T_1 \cup \cdots \cup T_n$?

Let $T_1, \ldots, T_n$ be theories with respective solvers $S_1, \ldots, S_n$

How can we integrate all of them cooperatively into a single SMT solver for $T = T_1 \cup \cdots \cup T_n$?

Quick Solution:

1. Combine $S_1, \ldots, S_n$ with Nelson-Oppen into a theory solver for T
2. Build a DPLL(T) solver as usual

Let $T_1, \ldots, T_n$ be theories with respective solvers $S_1, \ldots, S_n$

How can we integrate all of them cooperatively into a single SMT solver for $T = T_1 \cup \cdots \cup T_n$?

Better Solution [Bar02, BBC+05b, BNOT06]:

1. Extend DPLL(T) to DPLL($T_1, \ldots, T_n$)
2. Lift Nelson-Oppen to the DPLL($X_1, \ldots, X_n$) level
3. Build a DPLL($T_1, \ldots, T_n$) solver

- Let $n = 2$, for simplicity
- Let T_i be of signature Σ_i for $i = 1, 2$, with $\Sigma_1 \cap \Sigma_2 = \emptyset$
- Let $\mathcal{C}$ be a set of free constants
- Assume wlog that each input literal has signature $(\Sigma_1 \cup \mathcal{C})$ or $(\Sigma_2 \cup \mathcal{C})$ (no *mixed* literals)
- Let $\mathrm{M}|_i \stackrel{\text{def}}{=} \{(\Sigma_i \cup \mathcal{C})\text{-literals of M and their complement}\}$
- Let $\mathrm{I}(\mathrm{M}) \stackrel{\text{def}}{=} \{c = d \mid c, d \text{ occur in } \mathcal{C}, \mathrm{M}|_1 \text{ and } \mathrm{M}|_2\} \cup$
  $\{c \neq d \mid c, d \text{ occur in } \mathcal{C}, \mathrm{M}|_1 \text{ and } \mathrm{M}|_2\}$
  (*interface literals*)

Propagate, Conflict, Explain, Backjump, Fail (unchanged)

Propagate, **Conflict**, **Explain**, **Backjump**, **Fail** (unchanged)

Decide $$\frac{l \in \mathrm{Lit}(F) \cup \mathrm{I}(M) \quad l, \bar{l} \notin M}{M := M \bullet l}$$

Only change: decide on interface equalities as well

Propagate, **Conflict**, **Explain**, **Backjump**, **Fail** (unchanged)

Decide $$\frac{l \in \mathrm{Lit}(F) \cup \mathrm{I}(M) \quad l, \bar{l} \notin M}{M := M \bullet l}$$

Only change: decide on interface equalities as well

***T*-Propagate** $$\frac{l \in \mathrm{Lit}(F) \cup \mathrm{I}(M) \quad i \in \{1, 2\} \quad M \models_{T_i} l \quad l, \bar{l} \notin M}{M := M\ l}$$

Only change: propagate interface equalities as well, but reason locally in each T_i

T-Conflict

$$\frac{C = \mathsf{no} \quad l_1, \dots, l_n \in M \quad l_1, \dots, l_n \models_{T_i} \bot \quad i \in \{1,2\}}{C := \bar{l}_1 \vee \cdots \vee \bar{l}_n}$$

T-Explain

$$\frac{C = l \vee D \quad \bar{l}_1, \dots, \bar{l}_n \models_{T_i} \bar{l} \quad i \in \{1,2\} \quad \bar{l}_1, \dots, \bar{l}_n \prec_M \bar{l}}{C := l_1 \vee \cdots \vee l_n \vee D}$$

Only change: reason locally in each T_i

T-Conflict

$$\frac{\mathsf{C} = \mathsf{no} \quad l_1, \ldots, l_n \in \mathsf{M} \quad l_1, \ldots, l_n \models_{T_i} \bot \quad i \in \{1, 2\}}{\mathsf{C} := \bar{l}_1 \vee \cdots \vee \bar{l}_n}$$

T-Explain

$$\frac{\mathsf{C} = l \vee D \quad \bar{l}_1, \ldots, \bar{l}_n \models_{T_i} \bar{l} \quad i \in \{1, 2\} \quad \bar{l}_1, \ldots, \bar{l}_n \prec_{\mathsf{M}} \bar{l}}{\mathsf{C} := l_1 \vee \cdots \vee l_n \vee D}$$

Only change: reason locally in each T_i

$\mathbf{I}$-Learn

$$\frac{\models_{T_i} l_1 \vee \cdots \vee l_n \quad l_1, \ldots, l_n \in \mathsf{M}|_i \cup \mathbf{I}(\mathsf{M}) \quad i \in \{1, 2\}}{\mathsf{F} := \mathsf{F} \cup \{l_1 \vee \cdots \vee l_n\}}$$

New rule: for entailed disjunctions of interface literals

EXAMPLE — CONVEX THEORIES

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = e_2}^{1} \wedge \overbrace{f(y) = e_3}^{2} \wedge \overbrace{f(e_4) = e_5}^{3} \wedge \overbrace{x = y}^{4} \wedge$$

$$\underbrace{e_2 - e_3 = e_1}_{5} \wedge \underbrace{e_4 = 0}_{6} \wedge \underbrace{e_5 > a + 2}_{7}$$

$$\underbrace{e_2 = e_3}_{8} \quad \underbrace{e_1 = e_4}_{9} \quad \underbrace{a = e_5}_{10}$$

EXAMPLE — CONVEX THEORIES

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = e_2}^{1} \wedge \overbrace{f(y) = e_3}^{2} \wedge \overbrace{f(e_4) = e_5}^{3} \wedge \overbrace{x = y}^{4} \wedge$$
$$\underbrace{e_2 - e_3 = e_1}_{5} \wedge \underbrace{e_4 = 0}_{6} \wedge \underbrace{e_5 > a + 2}_{7}$$

$$\underbrace{e_2 = e_3}_{8} \quad \underbrace{e_1 = e_4}_{9} \quad \underbrace{a = e_5}_{10}$$

M	F	C	rule
	F	no	

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = e_2}^{1} \wedge \overbrace{f(y) = e_3}^{2} \wedge \overbrace{f(e_4) = e_5}^{3} \wedge \overbrace{x = y}^{4} \wedge$$
$$\underbrace{e_2 - e_3 = e_1}_{5} \wedge \underbrace{e_4 = 0}_{6} \wedge \underbrace{e_5 > a + 2}_{7}$$

$$\underbrace{e_2 = e_3}_{8} \quad \underbrace{e_1 = e_4}_{9} \quad \underbrace{a = e_5}_{10}$$

M	F	C	rule
	F	no	
0 1 2 3 4 5 6 7	F	no	by **Propagate**$^+$

$$F := \overbrace{f(e_1) = a}^{0} \land \overbrace{f(x) = e_2}^{1} \land \overbrace{f(y) = e_3}^{2} \land \overbrace{f(e_4) = e_5}^{3} \land \overbrace{x = y}^{4} \land$$
$$\underbrace{e_2 - e_3 = e_1}_{5} \land \underbrace{e_4 = 0}_{6} \land \underbrace{e_5 > a + 2}_{7}$$

$$\underbrace{e_2 = e_3}_{8} \quad \underbrace{e_1 = e_4}_{9} \quad \underbrace{a = e_5}_{10}$$

M	F	C	rule
	F	no	
0 1 2 3 4 5 6 7	F	no	by **Propagate**$^+$
0 1 2 3 4 5 6 7 8	F	no	by T-**Propagate** (1, 2, 4 $\models_{\mathrm{UF}}$ 8)

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = e_2}^{1} \wedge \overbrace{f(y) = e_3}^{2} \wedge \overbrace{f(e_4) = e_5}^{3} \wedge \overbrace{x = y}^{4} \wedge$$
$$\underbrace{e_2 - e_3 = e_1}_{5} \wedge \underbrace{e_4 = 0}_{6} \wedge \underbrace{e_5 > a + 2}_{7}$$

$$\underbrace{e_2 = e_3}_{8} \quad \underbrace{e_1 = e_4}_{9} \quad \underbrace{a = e_5}_{10}$$

M	F	C	rule
	F	no	
0 1 2 3 4 5 6 7	F	no	by **Propagate**$^{+}$
0 1 2 3 4 5 6 7 8	F	no	by T-**Propagate** ($1, 2, 4 \models_{\text{UF}} 8$)
0 1 2 3 4 5 6 7 8 9	F	no	by T-**Propagate** ($5, 6, 8 \models_{\text{LRA}} 9$)

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = e_2}^{1} \wedge \overbrace{f(y) = e_3}^{2} \wedge \overbrace{f(e_4) = e_5}^{3} \wedge \overbrace{x = y}^{4} \wedge$$
$$\underbrace{e_2 - e_3 = e_1}_{5} \wedge \underbrace{e_4 = 0}_{6} \wedge \underbrace{e_5 > a + 2}_{7}$$

$$\underbrace{e_2 = e_3}_{8} \quad \underbrace{e_1 = e_4}_{9} \quad \underbrace{a = e_5}_{10}$$

M	F	C	rule
	F	no	
0 1 2 3 4 5 6 7	F	no	by **Propagate**$^+$
0 1 2 3 4 5 6 7 8	F	no	by T-**Propagate** ($1, 2, 4 \models_{\mathrm{UF}} 8$)
0 1 2 3 4 5 6 7 8 9	F	no	by T-**Propagate** ($5, 6, 8 \models_{\mathrm{LRA}} 9$)
0 1 2 3 4 5 6 7 8 9 10	F	no	by T-**Propagate** ($0, 3, 9 \models_{\mathrm{UF}} 10$)

$$F := \overbrace{f(e_1) = a}^{0} \land \overbrace{f(x) = e_2}^{1} \land \overbrace{f(y) = e_3}^{2} \land \overbrace{f(e_4) = e_5}^{3} \land \overbrace{x = y}^{4} \land$$
$$\underbrace{e_2 - e_3 = e_1}_{5} \land \underbrace{e_4 = 0}_{6} \land \underbrace{e_5 > a + 2}_{7}$$

$$\underbrace{e_2 = e_3}_{8} \quad \underbrace{e_1 = e_4}_{9} \quad \underbrace{a = e_5}_{10}$$

M	F	C	rule
	F	no	
0 1 2 3 4 5 6 7	F	no	by **Propagate**$^{+}$
0 1 2 3 4 5 6 7 8	F	no	by T-**Propagate** ($1, 2, 4 \models_{\mathrm{UF}} 8$)
0 1 2 3 4 5 6 7 8 9	F	no	by T-**Propagate** ($5, 6, 8 \models_{\mathrm{LRA}} 9$)
0 1 2 3 4 5 6 7 8 9 10	F	no	by T-**Propagate** ($0, 3, 9 \models_{\mathrm{UF}} 10$)
0 1 2 3 4 5 6 7 8 9 10	F	$\overline{7} \lor \overline{10}$	by T-**Conflict** ($7, 10 \models_{\mathrm{LRA}} \bot$)

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = e_2}^{1} \wedge \overbrace{f(y) = e_3}^{2} \wedge \overbrace{f(e_4) = e_5}^{3} \wedge \overbrace{x = y}^{4} \wedge$$
$$\underbrace{e_2 - e_3 = e_1}_{5} \wedge \underbrace{e_4 = 0}_{6} \wedge \underbrace{e_5 > a + 2}_{7}$$

$$\underbrace{e_2 = e_3}_{8} \quad \underbrace{e_1 = e_4}_{9} \quad \underbrace{a = e_5}_{10}$$

M	F	C	rule
	F	no	
0 1 2 3 4 5 6 7	F	no	by **Propagate**$^+$
0 1 2 3 4 5 6 7 8	F	no	by **T-Propagate** ($1, 2, 4 \models_{\mathrm{UF}} 8$)
0 1 2 3 4 5 6 7 8 9	F	no	by **T-Propagate** ($5, 6, 8 \models_{\mathrm{LRA}} 9$)
0 1 2 3 4 5 6 7 8 9 10	F	no	by **T-Propagate** ($0, 3, 9 \models_{\mathrm{UF}} 10$)
0 1 2 3 4 5 6 7 8 9 10	F	$\overline{7} \vee \overline{10}$	by **T-Conflict** ($7, 10 \models_{\mathrm{LRA}} \perp$)
fail			by **Fail**

EXAMPLE — NON-CONVEX THEORIES

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = b}^{1} \wedge \overbrace{f(e_2) = e_3}^{2} \wedge \overbrace{f(e_1) = e_4}^{3} \wedge$$
$$\underbrace{1 \leq x}_{4} \wedge \underbrace{x \leq 2}_{5} \wedge \underbrace{e_1 = 1}_{6} \wedge \underbrace{a = b + 2}_{7} \wedge \underbrace{e_2 = 2}_{8} \wedge \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

$$F := \overbrace{f(e_1) = a}^{0} \land \overbrace{f(x) = b}^{1} \land \overbrace{f(e_2) = e_3}^{2} \land \overbrace{f(e_1) = e_4}^{3} \land$$
$$\underbrace{1 \leq x}_{4} \land \underbrace{x \leq 2}_{5} \land \underbrace{e_1 = 1}_{6} \land \underbrace{a = b + 2}_{7} \land \underbrace{e_2 = 2}_{8} \land \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	

EXAMPLE — NON-CONVEX THEORIES

$$F := \overbrace{f(e_1) = a}^{0} \land \overbrace{f(x) = b}^{1} \land \overbrace{f(e_2) = e_3}^{2} \land \overbrace{f(e_1) = e_4}^{3} \land$$
$$\underbrace{1 \leq x}_{4} \land \underbrace{x \leq 2}_{5} \land \underbrace{e_1 = 1}_{6} \land \underbrace{a = b + 2}_{7} \land \underbrace{e_2 = 2}_{8} \land \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$

EXAMPLE — NON-CONVEX THEORIES

$$F := \overbrace{f(e_1) = a}^{0} \land \overbrace{f(x) = b}^{1} \land \overbrace{f(e_2) = e_3}^{2} \land \overbrace{f(e_1) = e_4}^{3} \land$$
$$\underbrace{1 \leq x}_{4} \land \underbrace{x \leq 2}_{5} \land \underbrace{e_1 = 1}_{6} \land \underbrace{a = b + 2}_{7} \land \underbrace{e_2 = 2}_{8} \land \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$
$0 \cdots 9$ 10	F	no	by T-**Propagate** $(0, 3 \models_{\mathrm{UF}} 10)$

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = b}^{1} \wedge \overbrace{f(e_2) = e_3}^{2} \wedge \overbrace{f(e_1) = e_4}^{3} \wedge$$
$$\underbrace{1 \leq x}_{4} \wedge \underbrace{x \leq 2}_{5} \wedge \underbrace{e_1 = 1}_{6} \wedge \underbrace{a = b + 2}_{7} \wedge \underbrace{e_2 = 2}_{8} \wedge \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$
$0 \cdots 9\ 10$	F	no	by T-**Propagate** ($0, 3 \models_{\mathrm{UF}} 10$)
$0 \cdots 9\ 10$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **I-Learn** ($\models_{\mathrm{LIA}} \overline{4} \vee \overline{5} \vee 11 \vee 12$)

$$F := \overbrace{f(e_1) = a}^{0} \land \overbrace{f(x) = b}^{1} \land \overbrace{f(e_2) = e_3}^{2} \land \overbrace{f(e_1) = e_4}^{3} \land$$
$$\underbrace{1 \leq x}_{4} \land \underbrace{x \leq 2}_{5} \land \underbrace{e_1 = 1}_{6} \land \underbrace{a = b + 2}_{7} \land \underbrace{e_2 = 2}_{8} \land \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$
$0 \cdots 9\ 10$	F	no	by T-**Propagate** $(0, 3 \models_{\mathrm{UF}} 10)$
$0 \cdots 9\ 10$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	no	by **I-Learn** $(\models_{\mathrm{LIA}} \overline{4} \lor \overline{5} \lor 11 \lor 12)$
$0 \cdots 9\ 10 \bullet 11$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	no	by **Decide**

$$F := \overbrace{f(e_1) = a}^{0} \land \overbrace{f(x) = b}^{1} \land \overbrace{f(e_2) = e_3}^{2} \land \overbrace{f(e_1) = e_4}^{3} \land$$
$$\underbrace{1 \leq x}_{4} \land \underbrace{x \leq 2}_{5} \land \underbrace{e_1 = 1}_{6} \land \underbrace{a = b + 2}_{7} \land \underbrace{e_2 = 2}_{8} \land \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$
$0 \cdots 9\ 10$	F	no	by **T-Propagate** $(0, 3 \models_{\mathrm{UF}} 10)$
$0 \cdots 9\ 10$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	no	by **I-Learn** $(\models_{\mathrm{LIA}} \overline{4} \lor \overline{5} \lor 11 \lor 12)$
$0 \cdots 9\ 10 \bullet 11$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	no	by **Decide**
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	no	by **T-Propagate** $(0, 1, 11 \models_{\mathrm{UF}} 13)$

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = b}^{1} \wedge \overbrace{f(e_2) = e_3}^{2} \wedge \overbrace{f(e_1) = e_4}^{3} \wedge$$
$$\underbrace{1 \leq x}_{4} \wedge \underbrace{x \leq 2}_{5} \wedge \underbrace{e_1 = 1}_{6} \wedge \underbrace{a = b + 2}_{7} \wedge \underbrace{e_2 = 2}_{8} \wedge \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$
$0 \cdots 9\ 10$	F	no	by **T-Propagate** $(0,\ 3 \models_{\mathrm{UF}} 10)$
$0 \cdots 9\ 10$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **I-Learn** $(\models_{\mathrm{LIA}} \overline{4} \vee \overline{5} \vee 11 \vee 12)$
$0 \cdots 9\ 10 \bullet 11$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Decide**
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **T-Propagate** $(0,\ 1,\ 11 \models_{\mathrm{UF}} 13)$
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	$\overline{7} \vee \overline{13}$	by **T-Conflict** $(7,\ 13 \models_{\mathrm{UF}} \perp)$

EXAMPLE — NON-CONVEX THEORIES

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = b}^{1} \wedge \overbrace{f(e_2) = e_3}^{2} \wedge \overbrace{f(e_1) = e_4}^{3} \wedge$$
$$\underbrace{1 \leq x}_{4} \wedge \underbrace{x \leq 2}_{5} \wedge \underbrace{e_1 = 1}_{6} \wedge \underbrace{a = b + 2}_{7} \wedge \underbrace{e_2 = 2}_{8} \wedge \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$
$0 \cdots 9\ 10$	F	no	by **T-Propagate** ($0, 3 \models_{\mathrm{UF}} 10$)
$0 \cdots 9\ 10$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **I-Learn** ($\models_{\mathrm{LIA}} \overline{4} \vee \overline{5} \vee 11 \vee 12$)
$0 \cdots 9\ 10 \bullet 11$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Decide**
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **T-Propagate** ($0, 1, 11 \models_{\mathrm{UF}} 13$)
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	$\overline{7} \vee \overline{13}$	by **T-Conflict** ($7, 13 \models_{\mathrm{UF}} \perp$)
$0 \cdots 9\ 10\ \overline{13}$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Backjump**

EXAMPLE — NON-CONVEX THEORIES

$$F := \overbrace{f(e_1) = a}^{0} \land \overbrace{f(x) = b}^{1} \land \overbrace{f(e_2) = e_3}^{2} \land \overbrace{f(e_1) = e_4}^{3} \land$$
$$\underbrace{1 \leq x}_{4} \land \underbrace{x \leq 2}_{5} \land \underbrace{e_1 = 1}_{6} \land \underbrace{a = b + 2}_{7} \land \underbrace{e_2 = 2}_{8} \land \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$
$0 \cdots 9\ 10$	F	no	by **T-Propagate** ($0,\ 3 \models_{\mathrm{UF}} 10$)
$0 \cdots 9\ 10$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	no	by **I-Learn** ($\models_{\mathrm{LIA}} \overline{4} \lor \overline{5} \lor 11 \lor 12$)
$0 \cdots 9\ 10 \bullet 11$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	no	by **Decide**
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	no	by **T-Propagate** ($0,\ 1,\ 11 \models_{\mathrm{UF}} 13$)
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	$\overline{7} \lor \overline{13}$	by **T-Conflict** ($7,\ 13 \models_{\mathrm{UF}} \bot$)
$0 \cdots 9\ 10\ \overline{13}$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	no	by **Backjump**
$0 \cdots 9\ 10\ \overline{13}\ \overline{11}$	$F,\ \overline{4} \lor \overline{5} \lor 11 \lor 12$	no	by **T-Propagate** ($0,\ 1,\ \overline{13} \models_{\mathrm{UF}} \overline{11}$)

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = b}^{1} \wedge \overbrace{f(e_2) = e_3}^{2} \wedge \overbrace{f(e_1) = e_4}^{3} \wedge$$
$$\underbrace{1 \leq x}_{4} \wedge \underbrace{x \leq 2}_{5} \wedge \underbrace{e_1 = 1}_{6} \wedge \underbrace{a = b + 2}_{7} \wedge \underbrace{e_2 = 2}_{8} \wedge \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$
$0 \cdots 9\ 10$	F	no	by **T-Propagate** $(0, 3 \models_{\mathrm{UF}} 10)$
$0 \cdots 9\ 10$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **I-Learn** $(\models_{\mathrm{LIA}} \overline{4} \vee \overline{5} \vee 11 \vee 12)$
$0 \cdots 9\ 10 \bullet 11$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Decide**
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **T-Propagate** $(0, 1, 11 \models_{\mathrm{UF}} 13)$
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	$\overline{7} \vee \overline{13}$	by **T-Conflict** $(7, 13 \models_{\mathrm{UF}} \bot)$
$0 \cdots 9\ 10\ \overline{13}$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Backjump**
$0 \cdots 9\ 10\ \overline{13}\ \overline{11}$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **T-Propagate** $(0, 1, \overline{13} \models_{\mathrm{UF}} \overline{11})$
$0 \cdots 9\ 10\ \overline{13}\ \overline{11}\ 12$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Propagate**

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = b}^{1} \wedge \overbrace{f(e_2) = e_3}^{2} \wedge \overbrace{f(e_1) = e_4}^{3} \wedge$$
$$\underbrace{1 \leq x}_{4} \wedge \underbrace{x \leq 2}_{5} \wedge \underbrace{e_1 = 1}_{6} \wedge \underbrace{a = b + 2}_{7} \wedge \underbrace{e_2 = 2}_{8} \wedge \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$
$0 \cdots 9\ 10$	F	no	by **T-Propagate** ($0,\ 3 \models_{\mathrm{UF}} 10$)
$0 \cdots 9\ 10$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **I-Learn** ($\models_{\mathrm{LIA}} \overline{4} \vee \overline{5} \vee 11 \vee 12$)
$0 \cdots 9\ 10 \bullet 11$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Decide**
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **T-Propagate** ($0,\ 1,\ 11 \models_{\mathrm{UF}} 13$)
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	$\overline{7} \vee \overline{13}$	by **T-Conflict** ($7,\ 13 \models_{\mathrm{UF}} \bot$)
$0 \cdots 9\ 10\ \overline{13}$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Backjump**
$0 \cdots 9\ 10\ \overline{13}\ \overline{11}$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **T-Propagate** ($0,\ 1,\ \overline{13} \models_{\mathrm{UF}} \overline{11}$)
$0 \cdots 9\ 10\ \overline{13}\ \overline{11}\ 12$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Propagate**
$\cdots$			(exercise)

$$F := \overbrace{f(e_1) = a}^{0} \wedge \overbrace{f(x) = b}^{1} \wedge \overbrace{f(e_2) = e_3}^{2} \wedge \overbrace{f(e_1) = e_4}^{3} \wedge$$
$$\underbrace{1 \leq x}_{4} \wedge \underbrace{x \leq 2}_{5} \wedge \underbrace{e_1 = 1}_{6} \wedge \underbrace{a = b + 2}_{7} \wedge \underbrace{e_2 = 2}_{8} \wedge \underbrace{e_3 = e_4 + 3}_{9}$$

$$\underbrace{a = e_4}_{10} \quad \underbrace{x = e_1}_{11} \quad \underbrace{x = e_2}_{12} \quad \underbrace{a = b}_{13}$$

M	F	C	rule
	F	no	
$0 \cdots 9$	F	no	by **Propagate**$^+$
$0 \cdots 9\ 10$	F	no	by **T-Propagate** ($0, 3 \models_{\mathrm{UF}} 10$)
$0 \cdots 9\ 10$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **I-Learn** ($\models_{\mathrm{LIA}} \overline{4} \vee \overline{5} \vee 11 \vee 12$)
$0 \cdots 9\ 10 \bullet 11$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Decide**
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **T-Propagate** ($0, 1, 11 \models_{\mathrm{UF}} 13$)
$0 \cdots 9\ 10 \bullet 11\ 13$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	$\overline{7} \vee \overline{13}$	by **T-Conflict** ($7, 13 \models_{\mathrm{UF}} \bot$)
$0 \cdots 9\ 10\ \overline{13}$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Backjump**
$0 \cdots 9\ 10\ \overline{13}\ \overline{11}$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **T-Propagate** ($0, 1, \overline{13} \models_{\mathrm{UF}} \overline{11}$)
$0 \cdots 9\ 10\ \overline{13}\ \overline{11}\ 12$	$F,\ \overline{4} \vee \overline{5} \vee 11 \vee 12$	no	by **Propagate**
$\cdots$			(exercise)
fail	$\cdots$	$\cdots$	by **Fail**

REFERENCES

1. C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli. **Satisfiability Modulo Theories**. In Handbook of Satisfiability. IOS Press, 2009.
2. C. Barrett and C. Tinelli. **Satisfiability Modulo Theories**. In Handbook of Model Checking. Springer, 2017.
3. R. Sebastiani. **Lazy Satisfiability Modulo Theories**. Journal on Satisfiability, Boolean Modeling and Computation 3:141-224, 2007.
4. R. Nieuwenhuis, A. Oliveras, and C. Tinelli. **Solving SAT and SAT Modulo Theories: From an abstract Davis-Putnam-Logemann- Loveland procedure to DPLL(T)**. Journal of the ACM, 53(6):937-977, 2006.
5. S. Krstić and A. Goel. **Architecting Solvers for SAT Modulo Theories: Nelson-Oppen with DPLL**. In Proceeding of the Symposium on Frontiers of Combining Systems (FroCoS'07). Volume 4720 of LNCS. Springer, 2007.

References

Gilles Audemard, Piergiorgio Bertoli, Alessandro Cimatti, Artur Korniłowicz, and Roberto Sebastiani.
A SAT-based approach for solving formulas over boolean and linear mathematical propositions.
In Andrei Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, volume 2392 of *Lecture Notes in Artificial Intelligence*, pages 195–210. Springer, 2002.

Alessandro Armando, Claudio Castellini, and Enrico Giunchiglia.
SAT-based procedures for temporal reasoning.
In S. Biundo and M. Fox, editors, *Proceedings of the 5th European Conference on Planning (Durham, UK)*, volume 1809 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2000.

Alessandro Armando, Jacopo Mantovani, and Lorenzo Platania.
Bounded model checking of software using SMT solvers instead of SAT solvers.
In *Proceedings of the 13th International SPIN Workshop on Model Checking of Software (SPIN'06)*, volume 3925 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2006.

Clark W. Barrett.
Checking Validity of Quantifier-Free Formulas in Combinations of First-Order Theories.
PhD dissertation, Department of Computer Science, Stanford University, Stanford, CA, Sep 2002.

R. Brummayer and A. Biere.
Boolector: An Efficient SMT Solver for Bit-Vectors and Arrays.
In S. Kowalewski and A. Philippou, editors, *15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'05*, volume 5505 of *Lecture Notes in Computer Science*, pages 174–177. Springer, 2009.

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.

M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Schulz, and R. Sebastiani.
An incremental and layered procedure for the satisfiability of linear arithmetic logic.
In *Tools and Algorithms for the Construction and Analysis of Systems, 11th Int. Conf., (TACAS)*, volume 3440 of *Lecture Notes in Computer Science*, pages 317–333, 2005.

Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Tommi Junttila, Silvio Ranise, Roberto Sebastiani, and Peter van Rossu.
Efficient satisfiability modulo theories via delayed theory combination.
In K.Etessami and S. Rajamani, editors, *Proceedings of the 17th International Conference on Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, pages 335–349. Springer, 2005.

Roberto Bruttomesso, Alessandro Cimatti, Anders Franzén, Alberto Griggio, Ziyad Hanna, Alexander Nadel, Amit Palti, and Roberto Sebastiani.
A lazy and layered SMT(BV) solver for hard industrial verification problems.
In Werner Damm and Holger Hermanns, editors, *Proceedings of the* 19^{th} *International Conference on Computer Aided Verification*, volume 4590 of *Lecture Notes in Computer Science*, pages 547–560. Springer-Verlag, July 2007.

J. R. Burch and D. L. Dill.
Automatic verification of pipelined microprocessor control.
In *Procs. 6th Int. Conf. Computer Aided Verification (CAV)*, LNCS 818, pages 68–80, 1994.

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 W. Barrett, David L. Dill, and Aaron Stump.
Checking satisfiability of first-order formulas by incremental translation to SAT.
In J. C. Godskesen, editor, *Proceedings of the International Conference on Computer-Aided Verification*, Lecture Notes in Computer Science, 2002.

R. E. Bryant, S. M. German, and M. N. Velev.
Processor Verification Using Efficient Reductions of the Logic of Uninterpreted Functions to Propositional Logic.
ACM Transactions on Computational Logic, TOCL, 2(1):93–134, 2001.

C. Borralleras, S. Lucas, R. Navarro-Marset, E. Rodríguez-Carbonell, and A. Rubio.
Solving Non-linear Polynomial Arithmetic via SAT Modulo Linear Arithmetic.
In R. A. Schmidt, editor, *22nd International Conference on Automated Deduction , CADE-22*, volume 5663 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2009.

M. Bofill, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio.
A Write-Based Solver for SAT Modulo the Theory of Arrays.
In *Formal Methods in Computer-Aided Design, FMCAD*, pages 1–8, 2008.

Miquel Bofill, Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio.
The Barcelogic SMT solver.
In *Computer-aided Verification (CAV)*, volume 5123 of *Lecture Notes in Computer Science*, pages 294–298. Springer, 2008.

Clark Barrett, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli.
Splitting on demand in sat modulo theories.
In M. Hermann and A. Voronkov, editors, *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'06), Phnom Penh, Cambodia*, volume 4246 of *Lecture Notes in Computer Science*, pages 512–526. Springer, 2006.

Kshitij Bansal, Andrew Reynolds, Clark Barrett, and Cesare Tinelli.
A new decision procedure for finite sets and cardinality constraints in SMT.
In Nicola Olivetti and Ashish Tiwari, editors, *Proceedings of the 8th International Joint Conference on Automated Reasoning, Coimbra, Portugal*, volume 9706 of *Lecture Notes in Computer Science*, pages 82–98. Springer International Publishing, 2016.

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.

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

R. E. Bryant and M. N. Velev.
Boolean Satisfiability with Transitivity Constraints.
ACM Transactions on Computational Logic, TOCL, 3(4):604–627, 2002.

S. Cotton and O. Maler.
Fast and Flexible Difference Constraint Propagation for DPLL(T).
In A. Biere and C. P. Gomes, editors, *9th International Conference on Theory and Applications of Satisfiability Testing, SAT'06*, volume 4121 of *Lecture Notes in Computer Science*, pages 170–183. Springer, 2006.

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

Bruno Dutertre and Leonardo de Moura.
A Fast Linear-Arithmetic Solver for DPLL(T).
In T. Ball and R. B. Jones, editors, *18th International Conference on Computer Aided Verification, CAV'06*, volume 4144 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2006.

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.

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

L. de Moura and N. Bjørner.
Generalized, efficient array decision procedures.
In *9th International Conference on Formal Methods in Computer-Aided Design, FMCAD 2009*, pages 45–52. IEEE, 2009.

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

L. de Moura and H. Rueß.
Lemmas on Demand for Satisfiability Solvers.
In *5th International Conference on Theory and Applications of Satisfiability Testing, SAT'02*, pages 244–251, 2002.

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.

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

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

C. Flanagan, K. R. M Leino, M. Lillibridge, G. Nelson, and J. B. Saxe.
Extended static checking for Java.
In *Proc. ACM Conference on Programming Language Design and Implementation*, pages 234–245, June 2002.

Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli.
DPLL(T): Fast decision procedures.
In R. Alur and D. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification, CAV'04 (Boston, Massachusetts)*, volume 3114 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2004.

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.

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

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.

Liana Hadarean, Clark Barrett, Dejan Jovanović, Cesare Tinelli, and Kshitij Bansal.
A tale of two solvers: Eager and lazy approaches to bit-vectors.
In Armin Biere and Roderick Bloem, editors, *Proceedings of the 26th International Conference on Computer Aided Verification (CAV '14)*, volume 8559 of *Lecture Notes in Computer Science*, pages 680–695. Springer, July 2014.

George Hagen and Cesare Tinelli.
Scaling up the formal verification of Lustre programs with SMT-based techniques.
In A. Cimatti and R. Jones, editors, *Proceedings of the 8th International Conference on Formal Methods in Computer-Aided Design (FMCAV'08), Portland, Oregon*, pages 109–117. IEEE, 2008.

Dejan Jovanović and Clark Barrett.
Polite theories revisited.
In Chris Fermüller and Andrei Voronkov, editors, *Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 6397 of *Lecture Notes in Computer Science*, pages 402–416. Springer-Verlag, 2010.

Dejan Jovanović and Leonardo de Moura.
Solving Non-linear Arithmetic.
In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *6th International Joint Conference on Automated Reasoning (IJCAR '12)*, volume 7364 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012.

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

Sava Krstić and Amit Goel.
Architecting solvers for SAT modulo theories: Nelson-Oppen with DPLL.
In B. Konev and F. Wolter, editors, *Proceeding of the Symposium on Frontiers of Combining Systems (Liverpool, England)*, volume 4720 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2007.

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.

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.

Shuvendu K. Lahiri and Madanlal Musuvathi.
An Efficient Decision Procedure for UTVPI Constraints.
In B. Gramlich, editor, *5th International Workshop on Frontiers of Combining Systems, FroCos'05*, volume 3717 of *Lecture Notes in Computer Science*, pages 168–183. Springer, 2005.

S. K. Lahiri, R. Nieuwenhuis, and A. Oliveras.
SMT Techniques for Fast Predicate Abstraction.
In T. Ball and R. B. Jones, editors, *18th International Conference on Computer Aided Verification, CAV'06*, volume 4144 of *Lecture Notes in Computer Science*, pages 413–426. Springer, 2006.

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.

Baoluo Meng, Andrew Reynolds, Cesare Tinelli, and Clark Barrett.
Relational constraint solving in SMT.
In Leonardo de Moura, editor, *Proceedings of the 26th International Conference on Automated Deduction*, volume 10395 of *Lecture Notes in Computer Science*, pages 148–165. Springer, 2017.

Greg Nelson and Derek C. Oppen.
Simplification by cooperating decision procedures.
ACM Trans. on Programming Languages and Systems, 1(2):245–257, October 1979.

Greg Nelson and Derek C. Oppen.
Fast decision procedures based on congruence closure.
Journal of the ACM, 27(2):356–364, 1980.

Robert Nieuwenhuis and Albert Oliveras.
DPLL(T) with Exhaustive Theory Propagation and its Application to Difference Logic.
In Kousha Etessami and Sriram K. Rajamani, editors, *Proceedings of the 17th International Conference on Computer Aided Verification, CAV'05 (Edimburgh, Scotland)*, volume 3576 of *Lecture Notes in Computer Science*, pages 321–334. Springer, July 2005.

R. Nieuwenhuis and A. Oliveras.
Fast Congruence Closure and Extensions.
Information and Computation, IC, 2005(4):557–580, 2007.

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.
Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T).
Journal of the ACM, 53(6):937–977, November 2006.

Derek C. Oppen.
Complexity, convexity and combinations of theories.
Theoretical Computer Science, 12:291–302, 1980.

Christos H Papadimitriou.
On the complexity of integer programming.
Journal of the ACM (JACM), 28(4):765–768, 1981.

A. Pnueli, Y. Rodeh, O. Shtrichman, and M. Siegel.
Deciding Equality Formulas by Small Domains Instantiations.
In N. Halbwachs and D. Peled, editors, *11th International Conference on Computer Aided Verification, CAV'99*, volume 1633 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 1999.

Andrew Reynolds and Jasmin Christian Blanchette.
A decision procedure for (co)datatypes in SMT solvers.
Journal of Automated Reasoning, 58(3):341–362, 2016.

Christophe Ringeissen.
Cooperation of decision procedures for the satisfiability problem.
In F. Baader and K.U. Schulz, editors, *Frontiers of Combining Systems: Proceedings of the 1st International Workshop, Munich (Germany)*, Applied Logic, pages 121–140. Kluwer Academic Publishers, March 1996.

Silvio Ranise, Christophe Ringeissen, and Calogero G. Zarba.
Combining data structures with nonstably infinite theories using many-sorted logic.
In B. Gramlich, editor, *Proceedings of the Workshop on Frontiers of Combining Systems*, volume 3717 of *Lecture Notes in Computer Science*, pages 48–64. Springer, 2005.

A. Stump, C. W. Barrett, D. L. Dill, and J. R. Levitt.
A Decision Procedure for an Extensional Theory of Arrays.
In *16th Annual IEEE Symposium on Logic in Computer Science, LICS'01*, pages 29–37. IEEE Computer Society, 2001.

Sanjit A. Seshia, Shuvendu K. Lahiri, and Randal E. Bryant.
A hybrid SAT-based decision procedure for separation logic with uninterpreted functions.
In *Proc. 40th Design Automation Conference*, pages 425–430. ACM Press, 2003.

O. Strichman, S. A. Seshia, and R. E. Bryant.
Deciding Separation Formulas with SAT.
In E. Brinksma and K. G. Larsen, editors, *14th International Conference on Computer Aided Verification, CAV'02*, volume 2404 of *Lecture Notes in Computer Science*, pages 209–222. Springer, 2002.

N. Tillmann and J. de Halleux.
Pex-White Box Test Generation for .NET.
In B. Beckert and R. Hähnle, editors, *2nd International Conference on Tests and Proofs, TAP'08*, volume 4966 of *Lecture Notes in Computer Science*, pages 134–153. Springer, 2008.

Cesare Tinelli and Mehdi T. Harandi.
A new correctness proof of the Nelson–Oppen combination procedure.
In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems: Proceedings of the 1st International Workshop (Munich, Germany)*, Applied Logic, pages 103–120. Kluwer Academic Publishers, March 1996.

C. Tinelli.
A DPLL-based calculus for ground satisfiability modulo theories.
In G. Ianni and S. Flesca, editors, *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (Cosenza, Italy)*, volume 2424 of *Lecture Notes in Artificial Intelligence*. Springer, 2002.

Cesare Tinelli and Calogero Zarba.
Combining nonstably infinite theories.
Journal of Automated Reasoning, 34(3):209–238, April 2005.

C. Wang, F. Ivancic, M. K. Ganai, and A. Gupta.
Deciding Separation Logic Formulae by SAT and Incremental Negative Cycle Elimination.
In G. Sutcliffe and A. Voronkov, editors, *12h International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'05*, volume 3835 of *Lecture Notes in Computer Science*, pages 322–336. Springer, 2005.

Harald Zankl and Aart Middeldorp.
Satisfiability of Non-linear (Ir)rational Arithmetic.
In Edmund M. Clarke and Andrei Voronkov, editors, *16th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'10*, volume 6355 of *Lecture Notes in Computer Science*, pages 481–500. Springer, 2010.

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.