FOUNDATIONS OF SATISFIABILITY MODULO THEORIES

SC² 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.

**Disclaimer**: 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 \textit{uniform} proof-search procedures for First Order Logic

\textbf{Limited success:} is FOL the best \textit{compromise} between \textit{expressivity} and \textit{efficiency}?

\textbf{Most recently} R&D has focused on:

\begin{itemize}
  \item addressing mostly (expressive enough) \textit{decidable fragments} of a certain logic
  \item incorporating \textbf{domain-specific} reasoning, e.g on:
    \begin{itemize}
      \item arithmetic reasoning
      \item equality
      \item data structures (arrays, lists, stacks, …)
    \end{itemize}
\end{itemize}
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):

\[
g(a) = c \quad \land \quad (f(g(a)) \neq f(c) \quad \lor \quad g(a) = d) \quad \land \quad c \neq d
\]

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

- Arithmetic
- Arrays
- UF
- Bit-Vectors

Core

- explanations
- conflicts
- lemmas
- propagations

assertions

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**

**SAT Solver**

- **DPLL**

**Core**

- **Arithmetic**
- **Arrays**
- **UF**
- **Bit-Vectors**

**assertions**

**explanations**

**conflicts**

**lemmas**

**propagations**
Core

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

\[ a \times (|b| + c) = d \land b \times (|a| + c) \neq d \land a = b \]

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

\[ a \times (|b| + c) = d \land b \times (|a| + c) \neq d \land a = b \]

is unsatisfiable, but no arithmetic reasoning is needed.

If we abstract it to

\[ \text{mul}(a, \text{add}(|b|, c)) = d \land \text{mul}(b, \text{add}(|a|, c)) \neq d \land a = b \]

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\textsuperscript{+}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\textsuperscript{+}09, ZM10, JdM12]
Used in software verification and hardware verification (for memories) \([\text{SBDL01, BNO}^+08, \text{dMB09]}\)

Two interpreted function symbols \([\_\_\_\_]\) and \(\text{store}\)

Axiomatized by:

\[
\begin{align*}
\cdot \ \forall a \forall i \forall v. \ \text{store}(a, i, v)[i] &= v \\
\cdot \ \forall a \forall i \forall j \forall v. \ i \neq j \Rightarrow \text{store}(a, i, v)[j] &= a[j]
\end{align*}
\]

Sometimes also with extensionality:

\[
\begin{align*}
\cdot \ \forall a \forall b. \ (\forall i. \ a[i] &= b[i] \Rightarrow a = b)
\end{align*}
\]

Is the following set of literals satisfiable in this theory?

\[
\begin{align*}
\text{store}(a, i, x) \neq b, \ b[i] &= y, \ \text{store}(b, i, x)[j] &= y, \ a = b, \ i = j
\end{align*}
\]
Theories of interest: Bit Vectors

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?

\[
a[1:0] \neq b[1:0] \land (a \mid b) = c \land c[0:0] = 0 \land a[1:0] + b[1:0] = 0
\]
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 $\Phi$ of literals and determines whether $\Phi$ is $T$-satisfiable.

$\Phi$ is *$T$-satisfiable* iff there is some model $M$ of $T$ such that each formula in $\Phi$ 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:

\[ x_1 = y_1 \land \cdots \land x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
Main idea: apply congruence axiom:

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

Example

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

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

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

\[
\begin{align*}
[f(x, y) = x, & \quad h(y) = g(y), \quad f(f(x, y), y) = z, \quad g(x) \neq g(z)]
\end{align*}
\]
**EUF: CONGRUENCE CLOSURE**

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

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

\[
\begin{align*}
f(x, y) &= x, \\
h(y) &= g(y), \\
f(f(x, y), y) &= z, \\
g(x) &\neq g(z)
\end{align*}
\]
EUF: CONGRUENCE CLOSURE

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

\[ [ f(x, y) = x, \ h(y) = g(y), \ f(f(x, y), y) = z, \ g(x) \neq g(z) ] \]
EUF: CONGRUENCE CLOSURE

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

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

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

\[
\begin{align*}
\& [ f(x, y) = x, \ h(y) = g(y), \ f(f(x, y), y) = z, \ g(x) \neq g(z) ]
\end{align*}
\]

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

\[[ f(x, y) = x, \ h(y) = g(y), \ f(f(x, y), y) = z, \ g(x) \neq g(z) \ \]
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. \( g(x) \neq g(z) \)
**EUF: CONGRUENCE CLOSURE**

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

\[
\begin{array}{c}
\text{[ } f(x, y) = x, \ h(y) = g(y), \ f(f(x, y), y) = z, \ g(x) \neq g(z) \text{ ]}
\end{array}
\]

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. \( g(x) \neq g(z) \)
2. \( f(f(x, y), y) = z \)
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. \( g(x) \neq g(z) \)
2. \( f(f(x, y), y) = z \)
3. \( f(x, y) = x \)
\[ \forall a, i, e : \text{store}(a, i, e)[i] = e \]
\[ \forall a, i, j, e : i \neq j \Rightarrow \text{store}(a, i, e)[j] = a[j] \]
\[ \forall a, b : a \neq b \Rightarrow \exists i : a[i] \neq b[i] \]

**Common approach:**

- UF + lemmas on demand \([BB09b, DMB09b]\)
- Use EUF as if \text{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:

\[ x - y = k \quad \rightarrow \quad x - y \leq k \land y - x \leq k \]

\[ x - y < k \quad \rightarrow \quad x - y \leq k - 1 \quad \text{(integers)} \]

\[ x - y < k \quad \rightarrow \quad x - y \leq k - \delta \quad \text{(reals)} \]
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* $\iff \exists$ negative path
1. Construct a graph from literals
2. Check if there is a negative path

**Theorem** \( \text{Literals unsatisfiable} \iff \exists \text{ negative path} \)

**Example**

\[
\begin{align*}
\l br\ x \leq 1, \ x - y \leq 2, \ y - z \leq 3, \ z - x \leq -6 \r dr
\end{align*}
\]
1. Construct a graph from literals
2. Check if there is a negative path

**Theorem** \( \text{Literals unsatisfiable} \iff \exists \text{ negative path} \)

**Example**

\[
\begin{align*}
x &\leq 1, \\
x - y &\leq 2, \\
y - z &\leq 3, \\
z - x &\leq -6
\end{align*}
\]
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 $\iff \exists$ 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 $\iff \exists$ 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* $\iff \exists$ *negative path*

**Example**

$$\begin{align*}
[ & x \leq 1, \ x - y \leq 2, \ y - z \leq 3, \ z - x \leq -6 ]
\end{align*}$$
1. Construct a graph from literals
2. Check if there is a negative path

**Theorem** *Literals unsatisfiable* $\iff \exists$ 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 $a_1x_1 + \cdots + a_nx_n \gg b$
  with $a_1$ positive and $\gg \in \{\leq, \geq\}$

$$t = b \quad \rightarrow \quad t \leq b \land t \geq b$$

- Reductions:
  - $t < b \quad \rightarrow \quad t \leq b - 1$ (integer arith.)
  - $t < b \quad \rightarrow \quad t \leq b - \delta$ (real arith.)

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 $\sum a_i x_i \cong b$ as $s \cong b$ 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

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$= a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_i$</td>
<td>$= a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n$</td>
<td>$\vdots$</td>
<td></td>
<td>$l_i \leq s_i \leq u_i$</td>
<td>$\vdots$</td>
<td></td>
</tr>
<tr>
<td>$S_m$</td>
<td>$= a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
<td>$l_j \leq x_j \leq u_j$</td>
<td>$\vdots$</td>
</tr>
</tbody>
</table>
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

\[ l_i \leq s_i \leq u_i \]

\[ \vdots \]

\[ l_j \leq x_j \leq u_j \]

\[ \vdots \]
Tableau

\[ S_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n \]

\[ \vdots \]

\[ S_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n \]

\[ \vdots \]

\[ S_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n \]

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

Bounds

\[ l_i \leq s_i \leq u_i \]

\[ \vdots \]

\[ l_j \leq x_j \leq u_j \]

\[ \vdots \]
<table>
<thead>
<tr>
<th>Tableau</th>
<th>Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n )</td>
<td>( l_i \leq s_i \leq u_i )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td></td>
</tr>
<tr>
<td>( S_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n )</td>
<td>( l_j \leq x_j \leq u_j )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td></td>
</tr>
<tr>
<td>( S_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n )</td>
<td>( \vdots )</td>
</tr>
</tbody>
</table>

- Lhs variables are basic, rhs variables are non-basic
### Tableau

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Lhs variables are basic, rhs variables are non-basic

### Bounds

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_i \leq s_i \leq u_i$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$l_j \leq x_j \leq u_j$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LINEAR RATIONAL ARITHMETIC: TABLEAU

Tableau

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

$$\vdots$$

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

$$\vdots$$

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

Bounds

$$l_i \leq s_i \leq u_i$$

$$l_j \leq x_j \leq u_j$$

- 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

\[ S_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n \]

\[ : \]

\[ S_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n \]

\[ : \]

\[ S_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n \]

Bounds

\[ l_i \leq s_i \leq u_i \]

\[ l_j \leq x_j \leq u_j \]

- 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

\[ S_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n \]

\[ \vdots \]

\[ S_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n \]

\[ \vdots \]

\[ S_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n \]

Bounds

\[ l_i \leq s_i \leq u_i \]

\[ \vdots \]

\[ l_j \leq x_j \leq u_j \]

Case 1:

\[ \cdot \, v \text{ satisfies bound on the basic variables too} \]

\[ \cdot \, \text{Satisfiable, } v \text{ is the model!} \]
### 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

\[
l_i \leq s_i \leq u_i
\]

\[
\vdots
\]

\[
l_j \leq x_j \leq u_j
\]

### 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

\[ S_1 = a_{1,1} \cdot x_1 + \cdots + a_{1,i} \cdot x_j + \cdots + a_{1,n} \cdot x_n \]

\[ \vdots \]

\[ S_i = a_{i,1} \cdot x_1 + \cdots + a_{i,i} \cdot x_j + \cdots + a_{i,n} \cdot x_n \]

\[ \vdots \]

\[ S_m = a_{m,1} \cdot x_1 + \cdots + a_{m,i} \cdot x_j + \cdots + a_{m,n} \cdot x_n \]

Bounds

\[ l_i \leq s_i \leq u_i \]

\[ l_j \leq x_j \leq u_j \]

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
LINEAR RATIONAL ARITHMETIC: EXAMPLE

\[
\begin{align*}
2y - x - 2 &\leq 0, \\
-2y - x + 4 &\leq 0
\end{align*}
\]
Linear rational arithmetic: example

Tableau

\[
\begin{align*}
    s_1 &= 2y - x \\
    s_2 &= -2y - x
\end{align*}
\]

Bounds

\[
\begin{align*}
    -\infty &\leq x \leq +\infty \\
    -\infty &\leq y \leq +\infty \\
    -\infty &\leq s_1 \leq +\infty \\
    -\infty &\leq s_2 \leq +\infty
\end{align*}
\]

Assignment

\[
\begin{align*}
    x &\rightarrow 0 \\
    y &\rightarrow 0 \\
    s_1 &\rightarrow 0 \\
    s_2 &\rightarrow 0
\end{align*}
\]
LINEAR RATIONAL ARITHMETIC: EXAMPLE

Tableau

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

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

Bounds

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

Assignment
LINEAR RATIONAL ARITHMETIC: EXAMPLE

\[ \begin{bmatrix} s_1 \leq 2, \ s_2 \leq -4 \end{bmatrix} \]

**Tableau**

\[ s_1 = 2y - x \]
\[ s_2 = -2y - x \]

**Bounds**

- \(-\infty \leq x \leq +\infty\)
- \(-\infty \leq y \leq +\infty\)
- \(-\infty \leq s_1 \leq 2\)
- \(-\infty \leq s_2 \leq +\infty\)

**Assignment**

- \(x \mapsto 0\)
- \(y \mapsto 0\)
- \(s_1 \mapsto 0\)
- \(s_2 \mapsto 0\)
LINEAR RATIONAL ARITHMETIC: EXAMPLE

Tableau

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

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

<table>
<thead>
<tr>
<th>Bounds</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-\infty \leq x \leq +\infty$</td>
<td>$x \mapsto 0$</td>
</tr>
<tr>
<td>$-\infty \leq y \leq +\infty$</td>
<td>$y \mapsto 0$</td>
</tr>
<tr>
<td>$-\infty \leq s_1 \leq 2$</td>
<td>$s_1 \mapsto 0$</td>
</tr>
<tr>
<td>$-\infty \leq s_2 \leq +\infty$</td>
<td>$s_2 \mapsto 0$</td>
</tr>
</tbody>
</table>
LINEAR RATIONAL ARITHMETIC: EXAMPLE

\[
\begin{bmatrix}
  s_1 & \leq & 2, \\
  s_2 & \leq & -4
\end{bmatrix}
\]

Tableau

\[
\begin{align*}
  s_1 &= 2y - x \\
  s_2 &= -2y - x
\end{align*}
\]

Bounds

\[
\begin{align*}
  -\infty & \leq x \leq +\infty \\
  -\infty & \leq y \leq +\infty \\
  -\infty & \leq s_1 \leq 2 \\
  -\infty & \leq s_2 \leq -4
\end{align*}
\]

Assignment

\[
\begin{align*}
  x & \mapsto 0 \\
  y & \mapsto 0 \\
  s_1 & \mapsto 0 \\
  s_2 & \mapsto 0
\end{align*}
\]
$s_2$’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**

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

[ $s_1 \leq 2$, $s_2 \leq -4$ ]
**s_2**’s bound violated

- There is slack in y
- Pivot \( s_2 \) and \( y \)

\[
\begin{bmatrix}
  s_1 \leq 2,
  s_2 \leq -4
\end{bmatrix}
\]

**Tableau**

\[
\begin{align*}
  s_1 &= 2y - x \\
  s_2 &= -2y - x
\end{align*}
\]

**Bounds**

- \(-\infty \leq x \leq +\infty\)
- \(-\infty \leq y \leq +\infty\)
- \(-\infty \leq s_1 \leq 2\)
- \(-\infty \leq s_2 \leq -4\)

**Assignment**

- \( x \mapsto 0 \)
- \( y \mapsto 0 \)
- \( s_1 \mapsto 0 \)
- \( s_2 \mapsto 0 \)
**s₂’s bound violated**

- There is slack in \( y \)
- Pivot \( s₂ \) and \( y \)

\[
\begin{align*}
  s₁ &= -s₂ - 2x \\
  y &= -\frac{1}{2}s₂ - \frac{1}{2}x
\end{align*}
\]

**Tableau**

**Bounds**

- \(-\infty \leq x \leq +\infty\)
- \(-\infty \leq y \leq +\infty\)
- \(-\infty \leq s₁ \leq 2\)
- \(-\infty \leq s₂ \leq -4\)

**Assignment**

- \( x \mapsto 0 \)
- \( y \mapsto 0 \)
- \( s₁ \mapsto 0 \)
- \( s₂ \mapsto 0 \)
s₂’s bound violated

• There is slack in y
• Pivot s₂ and y
• Update s₂ value

Tableau

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

Bounds

\[ -\infty \leq x \leq +\infty \]
\[ -\infty \leq y \leq +\infty \]
\[ -\infty \leq s_1 \leq 2 \]
\[ -\infty \leq s_2 \leq -4 \]

Assignment

\[ x \mapsto 0 \]
\[ y \mapsto 0 \]
\[ s_1 \mapsto 0 \]
\[ s_2 \mapsto 0 \]
s\(_2\)'s bound violated

- There is slack in y
- Pivot s\(_2\) and y
- Update s\(_2\) value

Tableau

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

[ \( s_1 \leq 2, \ s_2 \leq -4 \) ]

**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

**Tableau**

\[
\begin{align*}
  s_1 &= -s_2 - 2x \\
  y &= -\frac{1}{2}s_2 - \frac{1}{2}x
\end{align*}
\]

**Bounds**

\[
\begin{align*}
  -\infty &\leq x \leq +\infty \\
  -\infty &\leq y \leq +\infty \\
  -\infty &\leq s_1 \leq 2 \\
  -\infty &\leq s_2 \leq -4
\end{align*}
\]

**Assignment**

\[
\begin{align*}
  x &\mapsto 0 \\
  y &\mapsto 0 \\
  s_1 &\mapsto 0 \\
  s_2 &\mapsto -4
\end{align*}
\]
$s_2$’s bound violated

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

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

\[
\begin{bmatrix}
s_1 \leq 2, & s_2 \leq -4
\end{bmatrix}
\]

**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**

\[
x \mapsto 0
\]

\[
y \mapsto 2
\]

\[
s_1 \mapsto 4
\]

\[
s_2 \mapsto -4
\]
s₁’s bound violated

- There is slack in x

Tableau

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

[ s₁ ≤ 2, s₂ ≤ −4 ]

Bounds

- ∞ ≤ x ≤ +∞
- ∞ ≤ y ≤ +∞
- ∞ ≤ s₁ ≤ 2
- ∞ ≤ s₂ ≤ −4

Assignment

x ↦ 0
y ↦ 2
s₁ ↦ 4
s₂ ↦ −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**

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

**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**

**Tableau**

\[
\begin{align*}
  x &= -\frac{1}{2}s_1 - \frac{1}{2}s_2 \\
  y &= \frac{1}{4}s_1 - \frac{1}{4}s_2
\end{align*}
\]

**Bounds**

\[
\begin{align*}
  -\infty &\leq x \leq +\infty \\
  -\infty &\leq y \leq +\infty \\
  -\infty &\leq s_1 \leq 2 \\
  -\infty &\leq s_2 \leq -4
\end{align*}
\]

**Assignment**

\[
\begin{align*}
  x &\mapsto 0 \\
  y &\mapsto 2 \\
  s_1 &\mapsto 4 \\
  s_2 &\mapsto -4
\end{align*}
\]

**Notes**

- **s_1**’s bound violated
- There is slack in \( x \)
- Pivot \( s_1 \) and \( x \)

\([ s_1 \leq 2, \ s_2 \leq -4 \]
$s_1$'s bound violated

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

**Tableau**

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

**Bounds**

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

**Assignment**

$$x \mapsto 0$$
$$y \mapsto 2$$
$$s_1 \mapsto 4$$
$$s_2 \mapsto -4$$
**s_1**’s bound violated

- There is slack in \( x \)
- Pivot \( s_1 \) and \( x \)
- Update \( s_1 \) value

**Tableau**

\[
\begin{align*}
    x &= -\frac{1}{2} s_1 - \frac{1}{2} s_2 \\
    y &= \frac{1}{4} s_1 - \frac{1}{4} s_2
\end{align*}
\]

**Bounds**

\(-\infty \leq x \leq +\infty\)  \(x \mapsto 0\)

\(-\infty \leq y \leq +\infty\)  \(y \mapsto 2\)

\(-\infty \leq s_1 \leq 2\)  \(s_1 \mapsto 2\)

\(-\infty \leq s_2 \leq -4\)  \(s_2 \mapsto -4\)
$s_1$’s bound violated

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

Tableau

\[
\begin{align*}
x &= - \frac{1}{2} s_1 - \frac{1}{2} s_2 \\
y &= \frac{1}{4} s_1 - \frac{1}{4} s_2
\end{align*}
\]

Bounds

| $-\infty \leq x \leq +\infty$ | $x \mapsto 0$ |
| $-\infty \leq y \leq +\infty$ | $y \mapsto 2$ |
| $-\infty \leq s_1 \leq 2$ | $s_1 \mapsto 2$ |
| $-\infty \leq s_2 \leq -4$ | $s_2 \mapsto -4$ |
**Linear Rational Arithmetic: Example**

$s_1$’s bound violated

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

**Tableau**

\[
\begin{align*}
x &= -\frac{1}{2}s_1 - \frac{1}{2}s_2 \\
y &= \frac{1}{4}s_1 - \frac{1}{4}s_2
\end{align*}
\]

**Bounds**

\[
\begin{align*}
-\infty &\leq x \leq +\infty \\
-\infty &\leq y \leq +\infty \\
-\infty &\leq s_1 \leq 2 \\
-\infty &\leq s_2 \leq -4
\end{align*}
\]

**Assignment**

\[
\begin{align*}
x &\mapsto 1 \\
y &\mapsto \frac{3}{2} \\
s_1 &\mapsto 2 \\
s_2 &\mapsto -4
\end{align*}
\]
LINEAR RATIONAL ARITHMETIC: EXAMPLE

\[ \begin{array}{c}
\text{Tableau} \\
\text{Bounds} \\
\text{Assignment}
\end{array} \]

\begin{align*}
x &= -\frac{1}{2} s_1 - \frac{1}{2} s_2 \\
y &= \frac{1}{4} s_1 - \frac{1}{4} s_2
\end{align*}

\[\begin{align*}
-\infty &\leq x \leq +\infty \\
-\infty &\leq y \leq +\infty \\
-\infty &\leq s_1 \leq 2 \\
-\infty &\leq s_2 \leq -4
\end{align*}\]

\[\begin{align*}
x &\mapsto 1 \\
y &\mapsto \frac{3}{2} \\
s_1 &\mapsto 2 \\
s_2 &\mapsto -4 \]
LINEAR INTEGER ARITHMETIC

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 $\nu$ is integral, then SAT (lucky case)
- Otherwise, refine:
  - Add branch-and-bound lemmas: $x \leq \lfloor \nu(x) \rfloor \lor x \geq \lceil \nu(x) \rceil$
  - Add cutting plane lemmas: new implied inequality falsified by $\nu$
- 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: $x \leq \lfloor v(x) \rfloor$ $\lor$ $x \geq \lceil v(x) \rceil$
  - Add cutting plane lemmas: new implied inequality falsified by $v$
- Additionally solve integer equalities
- Not guaranteed to terminate

Alternatives [JdM13, BSW15] not yet mature
Non-linear Arithmetic

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

\( f \) is in \( \mathbb{Z}[y, x] \), \( a_i \) are in \( \mathbb{Z}[y] \)

Examples

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

Polynomial Constraints

\[ f(x, y) > 0 \land g(x) < 0 \]
\[ p_1 > 0 \lor (p_2 = 0 \land p_3 < 0) \quad p_1, p_2, p_3 \in \mathbb{Z}[x_1, \ldots, x_n] \]

**Projection (Saturation)**

Project polynomials using a projection \( P \)

\[ \{p_1, p_2, p_3\} \mapsto \{p_1, p_2, p_3, p_4, \ldots, p_n\} \]

**Lifting (Model construction)**

For each variable \( x_k \)
1. Isolate roots of \( p_i(\alpha, x_k) \)
2. Choose a cell \( C \) and assign \( x_k \mapsto \alpha_k \in C \), continue
3. If no more cells, backtrack
NON-LINEAR REAL ARITHMETIC

Model Construction

Build partial model by assigning variables to values

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

Unit Reasoning

Reason about unit constraints

\[C_1 \equiv (x^2 + y^2 < 1) \quad C_2 \equiv (xy > 1)\]

Explain Conflicts

Explain conflicts using valid clausal reasons

\[
\overline{C_1} \lor \overline{C_2} \lor x \leq 0 \lor x \geq 1
\]
Unit Reasoning

Reason about unit constraints

\[ C_1 \equiv (x^2 + y^2 < 1) \quad C_2 \equiv (xy > 1) \]
\( x^3 - 2x^2 + 1 > 0 \)
\[ x^3 - 2x^2 + 1 > 0 \]
\[ x^3 - 2x^2 + 1 > 0 \]
\[ x^3 - 2x^2 + 1 > 0 \]
$x^3 - 2x^2 + 1 > 0$  
$-3x^3 + 8x^2 - 4x > 0$
\[ x^3 - 2x^2 + 1 > 0 \quad \text{and} \quad -3x^3 + 8x^2 - 4x > 0 \]
\[ x^3 - 2x^2 + 1 > 0 \quad -3x^3 + 8x^2 - 4x > 0 \]
Model Construction

Build partial model by assigning variables to values

\[\ldots, C_1, C_2, \ldots, x \mapsto \sqrt{2}/2, \ldots\]\n
Unit Reasoning

Reason about unit constraints

\[C_1 \equiv (x^2 + y^2 < 1) \quad C_2 \equiv (xy > 1)\]

Explain Conflicts

Explain conflicts using valid clausal reasons

\[\overline{C_1} \lor \overline{C_2} \lor x \leq 0 \lor x \geq 1\]
Explain Conflicts

Explain conflicts using valid clausal reasons

\[ \overline{C_1} \lor \overline{C_2} \lor x \leq 0 \lor x \geq 1 \]
\[
\begin{align*}
\text{Explanation: } & C_1 = x^2 + y^2 < 1 \land C_2 = xy > 1 \\
& \left\{ x \neq 2 \right\}
\end{align*}
\]
Non-linear real arithmetic

Explanation:

\[ C_1 \land C_2 \]

\[ C_1: x^2 + y^2 < 1 \]

\[ C_2: xy > 1 \]
\[ x^2 + y^2 < 1 \quad \land \quad xy > 1 \]

\[ [C_1, C_2, x \mapsto \sqrt{2}/2] \]
Unit Constraint Reasoning

\[ x^2 + y^2 < 1 \Rightarrow -\sqrt{3}/2 < y < \sqrt{3}/2 \]
\[-2y - x + 4 < 0 \Rightarrow y > \sqrt{2} \]
\[ C_1 \wedge C_2 \Rightarrow x \neq \sqrt{2}/2 \]

Explanation: \( C_1 \wedge C_2 \Rightarrow x \neq \sqrt{2}/2 \)
\[ x^2 + y^2 < 1 \quad \land \quad xy > 1 \]

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

Explanation: \( C_1 \land C_2 \Rightarrow \)
CAD Projection

\[ P = \{ x, -4 + 4x^2, 1 - x^2 + x^4 \} \]
NON-LINEAR REAL ARITHMETIC

CAD Projection

\[ P = \{ x, -4 + 4x^2, 1 - x^2 + x^4 \} \]
NON-LINEAR REAL ARITHMETIC

Explanation $C_1 \land C_2 \Rightarrow x \leq 0 \lor x \geq 1$
Explanation $\overline{C_1} \lor \overline{C_2} \lor x \leq 0 \lor x \geq 1$
$x^2 + y^2 < 1 \land xy > 1$

$[C_1, C_2]$

Explanation $\overline{C_1} \lor \overline{C_2} \lor x \leq 0 \lor x \geq 1$
NON-LINEAR REAL ARITHMETIC

\[
\begin{align*}
\begin{cases}
C_1 & \quad x^2 + y^2 < 1 \\
C_2 & \quad xy > 1
\end{cases}
\end{align*}
\]

\[\text{Explanation } C_1 \lor C_2 \lor x \leq 0 \lor x \geq 1\]
\[C_1\vphantom{\begin{array}{c} x^2 + y^2 < 1 \wedge \end{array}}\]
\[
x^2 + y^2 < 1 \quad \wedge \quad xy > 1
\]

\[\overline{C_1} \lor \overline{C_2} \lor x \leq 0 \lor x \geq 1\]

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

We focus on the lazy approach.
\( g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \)

Theory \( T \): Equality with Uninterpreted Functions
\[(\text{very}) \text{ LAZY APPROACH FOR SMT – EXAMPLE}\]

\[
g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d
\]

**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)
\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]
\[
g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d
\]

1. \[g(a) = c\]
2. \[f(g(a)) \neq f(c)\]
3. \[g(a) = d\]
4. \[c \neq d\]

• Send \(\{1, 2 \lor 3, 4\}\) to SAT solver.
\( g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \)

1. Send \( \{1, \overline{2} \lor 3, \overline{4}\} \) to SAT solver.
2. SAT solver returns model \( \{1, \overline{2}, \overline{4}\} \).
   Theory solver finds (concretization of) \( \{1, \overline{2}, \overline{4}\} \) unsat.
\[
g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d
\]

1. Send \{1, \overline{2} \lor 3, \overline{4}\} to SAT solver.
2. SAT solver returns model \{1, \overline{2}, \overline{4}\}.
   Theory solver finds (concretization of) \{1, \overline{2}, \overline{4}\} unsat.
3. Send \{1, \overline{2} \lor 3, \overline{4}, \overline{1} \lor 2 \lor 4\} to SAT solver.
\[
g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d
\]

1. Send \{1, \overline{2} \lor 3, \overline{4}\} to SAT solver.
2. SAT solver returns model \{1, \overline{2}, \overline{4}\}.
   Theory solver finds (concretization of) \{1, \overline{2}, \overline{4}\} unsat.
3. Send \{1, \overline{2} \lor 3, \overline{4}, \overline{1} \lor 2 \lor 4\} to SAT solver.
4. SAT solver returns model \{1, 3, \overline{4}\}.
   Theory solver finds \{1, 3, \overline{4}\} unsat.
(VERY) LAZY APPROACH FOR SMT – EXAMPLE

\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

1. Send \( \{1, \overline{2}, 3, \overline{4}\} \) to SAT solver.
2. SAT solver returns model \( \{1, \overline{2}, \overline{4}\} \).
   Theory solver finds (concretization of) \( \{1, \overline{2}, \overline{4}\} \) unsat.
3. Send \( \{1, \overline{2}, 3, \overline{4}, \overline{1} \lor 2 \lor 4\} \) to SAT solver.
4. SAT solver returns model \( \{1, 3, \overline{4}\} \).
   Theory solver finds \( \{1, 3, \overline{4}\} \) unsat.
5. Send \( \{1, \overline{2} \lor 3, \overline{4}, \overline{1} \lor 2, \overline{1} \lor 3 \lor 4\} \) to SAT solver.
\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

1. Send \( \{1, \overline{2} \lor 3, \overline{4}\} \) to SAT solver.
2. SAT solver returns model \( \{1, \overline{2}, \overline{4}\} \).
   Theory solver finds (concretization of) \( \{1, \overline{2}, \overline{4}\} \) unsat.
3. Send \( \{1, \overline{2} \lor 3, \overline{4}, \overline{1} \lor 2 \lor 4\} \) to SAT solver.
4. SAT solver returns model \( \{1, 3, \overline{4}\} \).
   Theory solver finds \( \{1, 3, \overline{4}\} \) unsat.
5. Send \( \{1, \overline{2} \lor 3, \overline{4}, \overline{1} \lor 2, \overline{1} \lor \overline{3} \lor 4\} \) to SAT solver.
6. SAT solver finds \( \{1, \overline{2} \lor 3, \overline{4}, \overline{1} \lor 2 \lor 4, \overline{1} \lor \overline{3} \lor 4\} \) unsat.

Done: the original formula is unsatisfiable in UF.
(VERY) LAZY APPROACH FOR SMT – EXAMPLE

\[
\begin{align*}
g(a) &= c & \wedge & f(g(a)) \neq f(c) & \lor & g(a) = d & \wedge & c \neq d \\
1 & & & 2 & & 3 & & 4
\end{align*}
\]

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

fail  or  $\langle M, F \rangle$

where

- $M$ is a sequence of literals and decision points 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 $M$
- $M[i] \overset{\text{def}}{=} M_0 \bullet \cdots \bullet M_i$
States:

fail or $\langle M, F \rangle$

Initial state:

- $\langle \emptyset, F_0 \rangle$, 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]

\[
\begin{array}{c}
p_1 \quad \cdots \quad p_n \\
\hline
[M := e_1] \quad [F := e_2]
\end{array}
\]

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

**Propagate**

\[ l_1 \lor \cdots \lor l_n \lor l \in F \quad \overline{l}_1, \ldots, \overline{l}_n \in M \quad l, \overline{l} \notin M \]

\[ M := M l \]

**Note:** When convenient, treat \( M \) as a set

**Note:** Clauses are treated modulo ACI of \( \lor \)
Extending the assignment

**Propagate**

\[
\begin{align*}
l_1 \lor \cdots \lor l_n \lor l & \in F \\
l_1, \ldots, l_n & \in M \\
l, \neg l & \notin M
\end{align*}
\]

\[
M := M \! \cup \! l
\]

**Note:** When convenient, treat \( M \) as a set

**Note:** Clauses are treated modulo ACI of \( \lor \)

**Decide**

\[
\begin{align*}
l & \in \text{Lit}(F) \\
l, \neg l & \notin M
\end{align*}
\]

\[
M := M \! \cdot \! l
\]

**Note:** \( \text{Lit}(F) := \{ l \mid l \text{ literal of } F \} \cup \{ \neg l \mid l \text{ literal of } F \} \)
Repairing the assignment

\[
\text{Fail} \quad \frac{l_1 \lor \cdots \lor l_n \in F \quad \bar{l}_1, \ldots, \bar{l}_n \in M \quad \bullet \notin M}{\text{fail}}
\]
Reparing the assignment

**Fail**

\[ l_1 \lor \cdots \lor l_n \in F \quad \bar{l}_1, \ldots, \bar{l}_n \in M \quad \bullet \notin M \]

fail

**Backtrack**

\[ l_1 \lor \cdots \lor l_n \in F \quad \bar{l}_1, \ldots, \bar{l}_n \in M \quad M = M \bullet l \quad N \quad \bullet \notin N \]

\[ M := M \bar{l} \]

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

**Expected final states:**
- fail if $F_0$ is unsatisfiable
- $\langle M, G, no \rangle$ otherwise, where
  - $G$ is equivalent to $F_0$ and
  - $M$ satisfies $G$
Replace **Backtrack** with

\[
\begin{align*}
C &= \text{no} \\
\text{lin}_1 \land \ldots \land \text{lin}_2 F_1 \land \ldots \land F_N \\
C &= \text{lin}_1 \land \ldots \land \text{lin}_2 \\
M_C &= \text{lin}_1 \\
\text{Backjump} &= \text{lev}_1 \ldots \text{lev}_n \ldots \text{lev}_i < \text{lev}_j \Rightarrow M_C = M[j]
\end{align*}
\]

Maintain invariant:

\[
F_j = pC \text{ and } M_j = pC \quad \text{when } C \neq \text{no}
\]

*Note:* \(j = p\) denotes propositional entailment
Replace **Backtrack** with

**Conflict** \[ C = \text{no} \quad l_1 \lor \cdots \lor l_n \in F \quad \bar{l}_1, \ldots, \bar{l}_n \in M \]
\[ C := l_1 \lor \cdots \lor l_n \]

**Explain** \[ C = l \lor D \quad l_1 \lor \cdots \lor l_n \lor \bar{l} \in F \quad \bar{l}_1, \ldots, \bar{l}_n \prec_M \bar{l} \]
\[ C := l_1 \lor \cdots \lor l_n \lor D \]

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

**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

\[
C = \text{no} \quad l_1 \lor \cdots \lor l_n \in F \quad \bar{l}_1, \ldots, \bar{l}_n \in M \\
C := l_1 \lor \cdots \lor l_n
\]

### Explain

\[
C = l \lor D \quad l_1 \lor \cdots \lor l_n \lor \bar{l} \in F \quad \bar{l}_1, \ldots, \bar{l}_n \prec_M \bar{l} \\
C := l_1 \lor \cdots \lor l_n \lor D
\]

### Backjump

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

Maintain invariant: \( F \models_p C \) and \( M \models_p \neg C \) when \( C \neq \text{no} \)

**Note:** \( \models_p \) denotes propositional entailment
Modify *Fail* to
Modify *Fail* to

\[
\text{Fail} \quad \frac{C \neq \text{no} \quad \bullet \notin M}{\text{fail}}
\]
\[ F := \{1, \overline{1} \lor 2, \overline{3} \lor 4, \overline{5} \lor 6, \overline{1} \lor \overline{5} \lor 7, \overline{2} \lor \overline{5} \lor 6 \lor 7\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(F)</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>
**EXECUTION EXAMPLE**

\[ F := \{1, \ 1 \lor 2, \ 3 \lor 4, \ 5 \lor 6, \ 1 \lor 5 \lor 7, \ 2 \lor 5 \lor 6 \lor 7\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>yes</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

\[ 1 \lor 2 \lor 5 \lor 6 \lor 7, \ 2 \lor 5 \lor 6 \lor 7 \]

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

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>(F)</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>
\[ F := \{1, \overline{1} \lor 2, \overline{3} \lor 4, \overline{5} \lor \overline{6}, \overline{1} \lor 5 \lor 7, \overline{2} \lor 5 \lor 6 \lor 7\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12</td>
<td>F</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12 \cdot 3</td>
<td>F</td>
<td>no</td>
<td>by Decide</td>
</tr>
</tbody>
</table>
\[ F := \{1, \overline{1} \lor 2, \overline{3} \lor 4, \overline{5} \lor 6, \overline{1} \lor \overline{5} \lor 7, \overline{2} \lor \overline{5} \lor 6 \lor 7\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( F )</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3</td>
<td>( F )</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
</tbody>
</table>

**M F C rule**

**F no by Propagate**

**F no by Decide**

**F no by Propagate**

**F no by Explain with**

**F no by Backjump**

**F no by Explain with**
\[ F := \{1, \ \overline{1} \lor 2, \ \overline{3} \lor 4, \ \overline{5} \lor 6, \ \overline{1} \lor \overline{5} \lor 7, \ \overline{2} \lor \overline{5} \lor 6 \lor \overline{7}\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>F</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 \cdot 3</td>
<td>F</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 \cdot 3 4</td>
<td>F</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 \cdot 3 4 \cdot 5</td>
<td>F</td>
<td>no</td>
<td>by Decide</td>
</tr>
</tbody>
</table>
\( F := \{1, \overline{1} \lor 2, \overline{3} \lor 4, \overline{5} \lor 6, \overline{1} \lor \overline{5} \lor 7, \overline{2} \lor \overline{5} \lor 6 \lor 7\} \)

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( F )</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 • 3</td>
<td>( F )</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 • 3 4</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 • 3 4 • 5</td>
<td>( F )</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 • 3 4 • 5 • 6</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
</tbody>
</table>
\[ F := \{1, \overline{1} \lor 2, \overline{3} \lor 4, \overline{5} \lor \overline{6}, \overline{1} \lor \overline{5} \lor 7, \overline{2} \lor \overline{5} \lor 6 \lor 7\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( F )</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12 * 3</td>
<td>( F )</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>12 * 3 4</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12 * 3 4 * 5</td>
<td>( F )</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>12 * 3 4 * 5 6</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12 * 3 4 * 5 6 7</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
</tbody>
</table>
\[ F := \{1, \overline{1} \lor 2, \overline{3} \lor 4, \overline{5} \lor \overline{6}, \overline{1} \lor \overline{5} \lor 7, \overline{2} \lor \overline{5} \lor 6 \lor \overline{7}\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3</td>
<td>( F )</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 ( \bullet ) 5</td>
<td>( F )</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 ( \bullet ) 5 ( \overline{6} )</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 ( \bullet ) 5 ( \overline{6} ) ( \overline{7} )</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 ( \bullet ) 5 ( \overline{6} ) ( \overline{7} )</td>
<td>( \overline{2} \lor \overline{5} \lor 6 \lor \overline{7} )</td>
<td>by Conflict</td>
<td></td>
</tr>
</tbody>
</table>
\[ F := \{1, \bar{1} \lor 2, \bar{3} \lor 4, \bar{5} \lor \bar{6}, \bar{1} \lor \bar{5} \lor 7, \bar{2} \lor \bar{5} \lor 6 \lor \bar{7}\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(F)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 * 3</td>
<td>(F)</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 * 3 4</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 * 3 4 * 5</td>
<td>(F)</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 * 3 4 * 5 * \bar{6}</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 * 3 4 * 5 * \bar{6} * \bar{7}</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 * 3 4 * 5 * \bar{6} * \bar{7}</td>
<td>(\bar{2} \lor \bar{5} \lor 6 \lor \bar{7})</td>
<td>by Conflict</td>
<td></td>
</tr>
<tr>
<td>1 2 * 3 4 * 5 * \bar{6} * \bar{7}</td>
<td>(1 \lor \bar{2} \lor \bar{5} \lor 6)</td>
<td>by Explain with (\bar{1} \lor \bar{5} \lor 7)</td>
<td></td>
</tr>
</tbody>
</table>
\[ F := \{1, \ \overline{1} \lor 2, \ \overline{3} \lor 4, \ \overline{5} \lor 6, \ \overline{1} \lor \overline{5} \lor 7, \ \overline{2} \lor \overline{5} \lor 6 \lor 7\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(F)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 (\cdot) 3</td>
<td>(F)</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 (\cdot) 3 4</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 (\cdot) 3 4 (\cdot) 5</td>
<td>(F)</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 (\cdot) 3 4 (\cdot) 5 (\cdot) 6</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 (\cdot) 3 4 (\cdot) 5 (\cdot) 6 (\cdot) 7</td>
<td>(F)</td>
<td>(\overline{2} \lor \overline{5} \lor 6 \lor 7)</td>
<td>by Conflict</td>
</tr>
<tr>
<td>1 2 (\cdot) 3 4 (\cdot) 5 (\overline{6}) (\cdot) 7</td>
<td>(F)</td>
<td>(1 \lor \overline{2} \lor \overline{5} \lor 6)</td>
<td>by Explain with (\overline{1} \lor \overline{5} \lor 7)</td>
</tr>
<tr>
<td>1 2 (\cdot) 3 4 (\cdot) 5 (\overline{6}) (\cdot) 7</td>
<td>(F)</td>
<td>(1 \lor \overline{2} \lor \overline{5})</td>
<td>by Explain with (\overline{5} \lor 6)</td>
</tr>
</tbody>
</table>
**EXECUTION EXAMPLE**

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

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12 3</td>
<td>(F)</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>12 3 4</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12 3 4 5</td>
<td>(F)</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>12 3 4 5 6</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12 3 4 5 6 7</td>
<td>(F)</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>12 3 4 5 6 7 (\overline{F})</td>
<td>(\overline{2} \vee \overline{5} \vee 6 \vee 7)</td>
<td>by Conflict</td>
<td></td>
</tr>
<tr>
<td>12 3 4 5 6 7</td>
<td>(\overline{F})</td>
<td>(1 \vee \overline{2} \vee \overline{5} \vee 6)</td>
<td>by Explain with (\overline{1} \vee \overline{5} \vee 7)</td>
</tr>
<tr>
<td>12 3 4 5 6 7</td>
<td>(\overline{F})</td>
<td>(1 \vee \overline{2} \vee \overline{5})</td>
<td>by Explain with (\overline{5} \vee \overline{6})</td>
</tr>
<tr>
<td>12 5</td>
<td>(F)</td>
<td>no</td>
<td>by Backjump</td>
</tr>
</tbody>
</table>
**EXECUTION EXAMPLE**

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

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3</td>
<td>( F )</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 ( \bullet ) 5</td>
<td>( F )</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 ( \bullet ) 5 ( \overline{6} )</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 ( \bullet ) 5 ( \overline{6} ) ( \overline{7} )</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 ( \bullet ) 5 ( \overline{6} ) ( \overline{7} )</td>
<td>( \overline{2} \lor \overline{5} \lor 6 \lor 7 )</td>
<td>by Conflict</td>
<td></td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 ( \bullet ) 5 ( \overline{6} ) ( \overline{7} )</td>
<td>( 1 \lor \overline{2} \lor \overline{5} \lor 6 )</td>
<td>by Explain with ( \overline{1} \lor \overline{5} \lor 7 )</td>
<td></td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 ( \bullet ) 5 ( \overline{6} ) ( \overline{7} )</td>
<td>( 1 \lor \overline{2} \lor 5 )</td>
<td>by Explain with ( 5 \lor 6 )</td>
<td></td>
</tr>
<tr>
<td>1 2 ( \overline{5} )</td>
<td>( F )</td>
<td>no</td>
<td>by Backjump</td>
</tr>
<tr>
<td>1 2 ( \overline{5} ) ( \bullet ) 3</td>
<td>( F )</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Also add

Learn: \[ \frac{F \models_p C \quad C \not\in F}{F := F \cup \{C\}} \]

Forget: \[ \frac{C = \text{no} \quad F = G \cup \{C\} \quad G \models_p C}{F := G} \]

Restart: \[ M := M^{[0]} \quad C := \text{no} \]

Note: Learn can be applied to any clause stored in C when C \neq \text{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**

\[\text{Basic DPLL} \overset{\text{def}}{=} \{ \text{Propagate, Decide, Conflict, Explain, Backjump} \}\]

\[\text{DPLL} \overset{\text{def}}{=} \text{Basic DPLL} + \{ \text{Learn, Forget, Restart}\}\]
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 $C = \text{no}$

*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 $C = no$

*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 $C = no$ 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 $M = \emptyset$ and $C = \text{no}$

*Exhausted execution:* execution ending in an irreducible state

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

**Proposition** (Completeness) For every exhausted execution starting with $F = F_0$ and ending with $C = \text{no}$, 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
FROM SAT TO SMT

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 $M \models_p \neg C$ when $C \neq \text{no}$

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

$T$-Conflict: $\frac{C = \text{no} \quad l_1, \ldots, l_n \in M \quad l_1, \ldots, l_n \models_T \bot}{C := \overline{l}_1 \lor \cdots \lor \overline{l}_n}$
Fix a theory $T$

$T$-Conflict: \[ C = \text{no} \quad l_1, \ldots, l_n \in M \quad l_1, \ldots, l_n \models_T \bot \]
\[ C := \overline{l}_1 \lor \cdots \lor \overline{l}_n \]

$T$-Propagate: \[ l \in \text{Lit}(F) \quad M \models_T l \quad l, \overline{l} \notin M \]
\[ M := M \cup \{l\} \]
SMT-LEVEL RULES

Fix a theory $T$

$T$-Conflict: $C = \text{no } l_1, \ldots, l_n \in M \quad l_1, \ldots, l_n \models_T \bot$

$C := \bar{l}_1 \lor \cdots \lor \bar{l}_n$

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

$M := M \ l$

$T$-Explain: $C = l \lor D \quad \bar{l}_1, \ldots, \bar{l}_n \models_T \bar{l} \quad \bar{l}_1, \ldots, \bar{l}_n \prec_M \bar{l}$

$C := l_1 \lor \cdots \lor l_n \lor D$

Note: $\bot = \text{empty clause}$

Note: $\models_T \text{ decided by theory solver}$
T-Conflict is enough to model the naive integration of SAT solvers and theory solvers seen in the earlier UF example.
MODELING THE VERY LAZY THEORY APPROACH

\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]
\[ g(a) = c \land f(g(a)) \neq f(c) \lor g(a) = d \land c \neq d \]

\[\begin{array}{c|c|c|c|c}
 M & F & C & \text{rule} \\
1 & \bar{2} \lor 3 & \bar{4} & \text{no} \\
\end{array}\]
MODELING THE VERY LAZY THEORY APPROACH

\[
g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1, 2 ∨ 3, 4</td>
<td>no</td>
</tr>
<tr>
<td>1 4</td>
<td>1 4</td>
<td>1, 2 ∨ 3, 4</td>
<td>no, by Propagate+</td>
</tr>
</tbody>
</table>
\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1, \overline{2} \lor 3, \overline{4}</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 \overline{4}</td>
<td>1, \overline{2} \lor 3, \overline{4}</td>
<td>no</td>
<td>by Propagate^+</td>
<td></td>
</tr>
<tr>
<td>1 \overline{4} \bullet 2</td>
<td>1, \overline{2} \lor 3, \overline{4}</td>
<td>no</td>
<td>by Decide</td>
<td></td>
</tr>
</tbody>
</table>
MODELING THE VERY LAZY THEORY APPROACH

\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>no</td>
<td>by Propagate+</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>no</td>
<td>by Decide</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>( \overline{1} \lor 2 \lor 4 )</td>
<td>by ( T )-Conflict</td>
<td></td>
</tr>
</tbody>
</table>
MODELING THE VERY LAZY THEORY APPROACH

\[
g(a) = c \land f(g(a)) \neq f(c) \lor g(a) = d \land c \neq d
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>\bar{2}</td>
<td>3</td>
<td>\bar{4}</td>
</tr>
<tr>
<td>1 \bar{4}</td>
<td>\bar{2}</td>
<td>3</td>
<td>\bar{4}</td>
</tr>
<tr>
<td>1 \bar{4} \cdot \bar{2}</td>
<td>\bar{2}</td>
<td>3</td>
<td>\bar{4}</td>
</tr>
<tr>
<td>1 \bar{4} \cdot \bar{2}</td>
<td>\bar{4}</td>
<td>3</td>
<td>\bar{4}</td>
</tr>
<tr>
<td>1 \bar{4} \cdot \bar{2}</td>
<td>, \bar{1} \lor 2 \lor 4</td>
<td>\bar{1} \lor 2 \lor 4</td>
<td>by Learn</td>
</tr>
</tbody>
</table>
MODELING THE VERY LAZY THEORY APPROACH

\[
g(a) = c \land f(g(a)) \neq f(c) \lor g(a) = d \land c \neq d
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1, 2 ∨ 3, 4</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, 2 ∨ 3, 4</td>
<td>no by Propagate^+</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, 2 ∨ 3, 4</td>
<td>no by Decide</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, 2 ∨ 3, 4</td>
<td>1 ∨ 2 ∨ 4 by T-Conflict</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, 2 ∨ 3, 4</td>
<td>1 ∨ 2 ∨ 4 by Learn</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, 2 ∨ 3, 4</td>
<td>no by Restart</td>
</tr>
</tbody>
</table>
MODELING THE VERY LAZY THEORY APPROACH

\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ( \overline{4} )</td>
<td>1, 2 \lor 3, 4</td>
<td>no</td>
<td>by Propagate$^+$</td>
</tr>
<tr>
<td>1 ( \overline{4} ) ( \overline{2} )</td>
<td>1, 2 \lor 3, 4</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 ( \overline{4} ) ( \overline{2} ) ( \overline{2} )</td>
<td>1, 2 \lor 3, 4</td>
<td>no</td>
<td>by $T$-Conflict</td>
</tr>
<tr>
<td>1 ( \overline{4} ) ( \overline{2} ) ( \overline{2} )</td>
<td>( \overline{1} ) \lor 2 \lor 4</td>
<td>by Learn</td>
<td></td>
</tr>
<tr>
<td>1 ( \overline{4} ) ( \overline{2} ) ( \overline{2} )</td>
<td>( \overline{1} ) \lor 2 \lor 4</td>
<td>no</td>
<td>by Restart</td>
</tr>
<tr>
<td>1 ( \overline{4} ) 2 3</td>
<td>( \overline{1} ) \lor 2 \lor 4</td>
<td>no</td>
<td>by Propagate$^+$</td>
</tr>
</tbody>
</table>
MODELING THE VERY LAZY THEORY APPROACH

\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 4</td>
<td>1, 2 v 3, 4</td>
<td>no</td>
<td>by Propagate⁺</td>
</tr>
<tr>
<td>1 4 • 2</td>
<td>1, 2 v 3, 4</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>1 4 • 2</td>
<td>1, 2 v 3, 4</td>
<td>1 v 2 v 4</td>
<td>by T-Conflict</td>
</tr>
<tr>
<td>1 4</td>
<td>1, 2 v 3, 4</td>
<td>1 v 2 v 4</td>
<td>by Learn</td>
</tr>
<tr>
<td>1 4 2 3</td>
<td>1 v 2 v 4</td>
<td>no</td>
<td>by Restart</td>
</tr>
<tr>
<td>1 4 2 3</td>
<td>1 v 2 v 4, 1 v 3 v 4</td>
<td>1 v 3 v 4</td>
<td>by T-Conflict, Learn</td>
</tr>
</tbody>
</table>
MODELING THE VERY LAZY THEORY APPROACH

\[
g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2 ∨ 3, 4</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1, 2 ∨ 3, 4</td>
<td>no</td>
<td>no by Propagate⁺</td>
<td></td>
</tr>
<tr>
<td>1, 2 ∨ 3, 4</td>
<td>no</td>
<td>by Decide</td>
<td></td>
</tr>
<tr>
<td>1, 2 ∨ 3, 4</td>
<td>1 ∨ 2 ∨ 4</td>
<td>by T-Conflict</td>
<td></td>
</tr>
<tr>
<td>1, 2 ∨ 3, 4</td>
<td>1 ∨ 2 ∨ 4</td>
<td>by Learn</td>
<td></td>
</tr>
<tr>
<td>1, 2 ∨ 3, 4</td>
<td>no</td>
<td>by Restart</td>
<td></td>
</tr>
<tr>
<td>1, 2 ∨ 3, 4</td>
<td>no</td>
<td>by Propagate⁺</td>
<td></td>
</tr>
<tr>
<td>1, 2 ∨ 3, 4</td>
<td>1 ∨ 2 ∨ 4</td>
<td>by T-Conflict, Learn</td>
<td></td>
</tr>
<tr>
<td>fail</td>
<td>1 ∨ 3 ∨ 4</td>
<td>by Fail</td>
<td></td>
</tr>
</tbody>
</table>
A BETTER LAZY APPROACH

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
A BETTER LAZY APPROACH

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

\[ g(a) = c \quad \land \quad \{ f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \} \]

\[ 1 \quad \overline{2} \quad 3 \quad \overline{4} \]
A BETTER LAZY APPROACH

\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2 v 3, 4</td>
<td>no</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ M \quad F \quad C \quad \text{rule} \]

\[ 1, 2 \lor 3, 4 \quad \text{no} \]
A BETTER LAZY APPROACH

\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>F</td>
<td>C</td>
<td>rule</td>
</tr>
<tr>
<td>1</td>
<td>2 \lor 3</td>
<td>4</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, 2 \lor 3</td>
<td>4</td>
</tr>
</tbody>
</table>
A BETTER LAZY APPROACH

\[
g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 \lor 3</td>
<td>4</td>
<td>no</td>
</tr>
<tr>
<td>1 4</td>
<td>1</td>
<td>2 \lor 3</td>
<td>4</td>
</tr>
<tr>
<td>1 4 \cdot 2</td>
<td>1</td>
<td>2 \lor 3</td>
<td>4</td>
</tr>
</tbody>
</table>
A BETTER LAZY APPROACH

\[
\begin{align*}
g(a) &= c & \land & \quad f(g(a)) \neq f(c) & \lor & \quad g(a) = d & \land & \quad c \neq d \\
\hline
1 & \quad \bar{4} & \quad 1, \bar{2} \lor 3, \bar{4} & \quad \text{no} & & & & \\
1 & \quad \bar{4} & \quad 1, \bar{2} \lor 3, \bar{4} & \quad \text{no} & \quad \text{by Propagate}^+ & & & \\
1 & \quad \bar{4} & \quad \bar{2} \lor 3, \bar{4} & \quad \text{no} & \quad \text{by Decide} & & & \\
1 & \quad \bar{4} & \quad \bar{2} \lor 3, \bar{4} & \quad \bar{1} \lor 2 & \quad \text{by T-Conflict} & & &
\end{align*}
\]
A BETTER LAZY APPROACH

\[
g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d
\]

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>1, (\overline{2} \lor 3, \overline{4})</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>(\overline{4})</td>
<td></td>
<td>1, (\overline{2} \lor 3, \overline{4})</td>
<td>no by Propagate(^+)</td>
</tr>
<tr>
<td>1</td>
<td>(\overline{4} \cdot \overline{2})</td>
<td></td>
<td>1, (\overline{2} \lor 3, \overline{4})</td>
<td>no by Decide</td>
</tr>
<tr>
<td>1</td>
<td>(\overline{4} \cdot \overline{2})</td>
<td>(\overline{2})</td>
<td>1, (\overline{2} \lor 3, \overline{4})</td>
<td>(\overline{1} \lor 2) by (T)-Conflict</td>
</tr>
<tr>
<td>1</td>
<td>(\overline{4})</td>
<td>(\overline{2})</td>
<td>1, (\overline{2} \lor 3, \overline{4})</td>
<td>no by Backjump</td>
</tr>
</tbody>
</table>
A BETTER LAZY APPROACH

\[ g(a) = c \land f(g(a)) \neq f(c) \lor g(a) = d \land c \neq d \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1, 2 \lor 3, 4</td>
<td>no</td>
</tr>
<tr>
<td>1, 4</td>
<td>2</td>
<td>1, 2 \lor 3, 4</td>
<td>no by Propagate^+</td>
</tr>
<tr>
<td>1 4</td>
<td>2</td>
<td>1, 2 \lor 3, 4</td>
<td>no by Decide</td>
</tr>
<tr>
<td>1 4</td>
<td>2</td>
<td>1, 2 \lor 3, 4</td>
<td>( \overline{1} \lor 2 ) by T-Conflict</td>
</tr>
<tr>
<td>1 4</td>
<td>2</td>
<td>1, 2 \lor 3, 4</td>
<td>no by Backjump</td>
</tr>
<tr>
<td>1 4</td>
<td>2</td>
<td>1, 2 \lor 3, 4</td>
<td>no by Propagate</td>
</tr>
</tbody>
</table>
A BETTER LAZY APPROACH

\[
g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1, 2 \lor 3, 4</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>4 1</td>
<td>1, 2 \lor 3, 4</td>
<td>no by Propagate$^+$</td>
</tr>
<tr>
<td>1</td>
<td>4 2</td>
<td>1, 2 \lor 3, 4</td>
<td>\bar{1} \lor 2 by T-Conflict</td>
</tr>
<tr>
<td>1</td>
<td>4 2 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4 2 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4 2 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4 2 3</td>
<td>1, 2 \lor 3, 4</td>
<td>\bar{1} \lor 3 \lor 4 by T-Conflict</td>
</tr>
</tbody>
</table>
A BETTER LAZY APPROACH

\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>( \overline{1} \lor 2 )</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>( \overline{1} \lor \overline{3} \lor 4 )</td>
</tr>
<tr>
<td>fail</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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**
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 \textit{T-Conflict} as the \textit{only theory rule}, the theory solver is used just to \textbf{validate} the choices of the SAT engine.
With \textbf{\textit{T-Conflict}} as the \textbf{only theory rule}, the theory solver is used just to \textbf{validate} the choices of the SAT engine.

With \textbf{\textit{T-Propagate}} and \textbf{\textit{T-Explain}}, it can also be used to \textbf{guide} the engine’s search [Tin02]

\begin{align*}
\textbf{\textit{T-Propagate}} \quad & l \in \text{Lit}(F) \quad M \models_T l \quad l, \bar{l} \notin M \\
& M := M \ l
\end{align*}

\begin{align*}
\textbf{\textit{T-Explain}} \quad & C = l \lor 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 \lor \cdots \lor l_n \lor D
\end{align*}
\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]
\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

\[ \begin{array}{c|c|c|c}
M & F & C & \text{rule} \\
\hline
1, \bar{2} \lor 3, \bar{4} & & & \text{no} \\
\end{array} \]
\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

1, 2 \lor 3, 4

1 4 1, 2 \lor 3, 4 no by Propagate\(^+\)
\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\overline{1}) (\overline{4})</td>
<td>(\overline{1}) (\overline{2}) \lor (\overline{3}) (\overline{4})</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>(\overline{1}) (\overline{4})</td>
<td>(\overline{1}) (\overline{2}) \lor (\overline{3}) (\overline{4})</td>
<td>no</td>
<td>by Propagate</td>
</tr>
<tr>
<td>(\overline{1}) (\overline{4}) (\overline{2})</td>
<td>(\overline{1}) (\overline{2}) \lor (\overline{3}) (\overline{4})</td>
<td>no</td>
<td>by (T)-Propagate (1 \models_T 2)</td>
</tr>
</tbody>
</table>
Theory propagation example

\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 ∨ 3, 4</td>
<td>no</td>
<td>by Propagate +</td>
</tr>
<tr>
<td>1</td>
<td>2 ∨ 3, 4</td>
<td>no</td>
<td>by T-Propagate (1 \models_T 2)</td>
</tr>
<tr>
<td>1</td>
<td>2 ∨ 3, 4</td>
<td>no</td>
<td>by T-Propagate (1, 4 \models_T 3)</td>
</tr>
</tbody>
</table>
\( g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \)

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1, ( \overline{2} \lor 3, \overline{4} )</td>
<td>no by Propagate(^+)</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>no by ( T )-Propagate (1 ( \models ) 2)</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2, 3</td>
<td>no by ( T )-Propagate (1, ( \overline{4} ) ( \models ) ( \overline{3} ))</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2, ( \overline{3} )</td>
<td>no by Conflict</td>
</tr>
</tbody>
</table>
\[ g(a) = c \quad \land \quad f(g(a)) \neq f(c) \quad \lor \quad g(a) = d \quad \land \quad c \neq d \]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>F</td>
<td>C</td>
<td>rule</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2 \lor 3, 4</td>
<td>no</td>
<td>by Propagate$^+$</td>
<td></td>
</tr>
<tr>
<td>1 4</td>
<td>1, 2 \lor 3, 4</td>
<td>no</td>
<td>by $T$-Propagate (1 $\models_T$ 2)</td>
<td></td>
</tr>
<tr>
<td>1 4 2</td>
<td>1, 2 \lor 3, 4</td>
<td>no</td>
<td>by $T$-Propagate (1, 4 $\models_T$ 3)</td>
<td></td>
</tr>
<tr>
<td>1 4 2 3</td>
<td>1, 2 \lor 3, 4</td>
<td>no</td>
<td>by Conflict</td>
<td></td>
</tr>
<tr>
<td>fail</td>
<td></td>
<td></td>
<td>by Fail</td>
<td></td>
</tr>
</tbody>
</table>

**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 $\overset{\text{def}}{=} (1) + (2)$

DPLL Modulo Theories $\overset{\text{def}}{=} (1) + (2) + (3)$
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 $C = \text{no}$

**Exhausted execution:** execution ending in an irreducible state

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

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

**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 \( C = \text{no} \) 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 \( M = \emptyset \) and \( C = \text{no} \)

*Exhausted execution:* execution ending in an irreducible state

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

**Proposition** (Completeness) For every exhausted execution starting with \( F = F_0 \) and ending with \( C = \text{no} \), \( F_0 \) is \( T \)-satisfiable; specifically, \( M \) is \( T \)-satisfiable and \( M \models_p F_0 \).
The approach formalized so far can be implemented with a simple architecture named $\text{DPLL}(T)$ \cite{ghn04,not06}

$$\text{DPLL}(T) = \text{DPLL}(X) \text{ engine + } T\text{-solver}$$
The approach formalized so far can be implemented with a simple architecture named $\text{DPLL}(T)$ \cite{ghn+04,not06}

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

$\text{DPLL}(X)$:

- 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 $\text{DPLL}(T)$ [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.

$$M = \{ r(w(a, i, x), j) \neq x, \ r(w(a, i, x), j) \neq r(a, j) \}$$
For certain theories, determining that a set \( M \) is \( T \)-unsatisfiable requires reasoning by cases.

**Example:** \( T \) = the theory of arrays.

\[
M = \{ r(w(a, i, x), j) \neq x, \ r(w(a, i, x), j) \neq r(a, j) \} \]

\( i = j \) 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.

\[ M = \{ r(w(a, i, x), j) \neq x, \quad r(w(a, i, x), j) \neq r(a, j) \} \]

$i = j$) 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.

\[ M = \{ r(w(a, i, x), j) \neq x, \quad r(w(a, i, x), j) \neq r(a, j) \} \]

- $i = j$) 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:**

\[ M = \{ \ldots, s = r(w(a, i, t), j), \ldots \} \]

- 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:

\[ M = \{\ldots, s = r(w(a, i, t), j), \ldots\} \]

- 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:
  \[ s = s' \land i = j \Rightarrow s = t, \quad s = s' \land i \neq j \Rightarrow s = r(a, j) \]”
To model the generation of theory lemmas for case splits, add the rule

\[ T{-}\text{Learn} \]

\[
\vdash_T \exists \mathbf{v}(l_1 \lor \cdots \lor l_n) \quad l_1, \ldots, l_n \in L_S \quad \mathbf{v} \text{ vars not in } F
\]

\[ F := F \cup \{l_1 \lor \cdots \lor l_n\} \]

where \( L_S \) is a finite set of literals dependent on the initial set of clauses (see \( \text{[BNOT06]} \) for a formal definition of \( L_S \)).
To model the generation of theory lemmas for case splits, add the rule

$$\vdash_T \exists v (l_1 \lor \cdots \lor l_n) \quad l_1, \ldots, l_n \in L_S \quad v \text{ vars not in } F$$

$$F := F \cup \{l_1 \lor \cdots \lor l_n\}$$

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 $M \models_p F$, 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 $M$
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 \( T\text{-Learn} \) 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 $M \models_p F$, 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 $M$
\[ F : \ x = y \cup z \land y \neq \emptyset \lor x \neq z \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y \cup z )</td>
<td>( F )</td>
<td>by Propagate(^+)</td>
</tr>
</tbody>
</table>
$F: \ x = y \cup z \ \land \ y \neq \emptyset \lor x \neq z$

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = y \cup z$</td>
<td>$F$</td>
<td>by Propagate$^+$</td>
</tr>
<tr>
<td>$x = y \cup z \land y = \emptyset$</td>
<td>$F$</td>
<td>by Decide</td>
</tr>
</tbody>
</table>
**EXAMPLE — THEORY OF FINITE SETS**

\[ F : \quad x = y \cup z \land y \neq \emptyset \lor x \neq z \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y \cup z )</td>
<td>( F )</td>
<td>by Propagate^+</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset )</td>
<td>( F )</td>
<td>by Decide</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x = z )</td>
<td>( F )</td>
<td>by Propagate</td>
</tr>
</tbody>
</table>
**EXAMPLE — THEORY OF FINITE SETS**

\( F : \ x = y \cup z \land y \neq \emptyset \lor x \neq z \)

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y \cup z )</td>
<td>( F )</td>
<td>by \text{Propagate}^+</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset )</td>
<td>( F )</td>
<td>by \text{Decide}</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z )</td>
<td>( F )</td>
<td>by \text{Propagate}</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z )</td>
<td>( F, (x = z \lor e \in x \lor e \in z), )</td>
<td>by \text{T-Learn}</td>
</tr>
</tbody>
</table>
**Example — Theory of Finite Sets**

\[ F : \quad x = y \cup z \quad \land \quad y \neq \emptyset \lor x \neq z \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x = y \cup z)</td>
<td>(F)</td>
<td>by Propagate$^+$</td>
</tr>
<tr>
<td>(x = y \cup z \land y = \emptyset)</td>
<td>(F)</td>
<td>by Decide</td>
</tr>
<tr>
<td>(x = y \cup z \land y = \emptyset \land x \neq z)</td>
<td>(F)</td>
<td>by Propagate</td>
</tr>
<tr>
<td>(x = y \cup z \land y = \emptyset \land x \neq z)</td>
<td>(F, (x = z \lor e \in x \lor e \in z), (x = z \lor e \notin x \lor e \notin z))</td>
<td>by T-Learn</td>
</tr>
</tbody>
</table>
**Example — Theory of Finite Sets**

\[ F : x = y \cup z \land y \neq \emptyset \lor x \neq z \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y \cup z )</td>
<td>( F )</td>
<td>Propagate(^+)</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset )</td>
<td>( F )</td>
<td>Decide</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z )</td>
<td>( F )</td>
<td>Propagate</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z )</td>
<td>( F, (x = z \lor e \in x \land e \in z), )</td>
<td>T-Learn</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z \land e \in x )</td>
<td>( F, (x = z \lor e \in x \land e \in z), )</td>
<td>Decide</td>
</tr>
</tbody>
</table>
**F**:  \( x = y \cup z \land y \neq \emptyset \lor x \neq z \)

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y \cup z )</td>
<td>( F )</td>
<td>by Propagate$^+$</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset )</td>
<td>( F )</td>
<td>by Decide</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z )</td>
<td>( F )</td>
<td>by Propagate</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z \land e \in x )</td>
<td>( F, (x = z \lor e \in x \lor e \in z), )</td>
<td>by T-Learn</td>
</tr>
<tr>
<td></td>
<td>( (x = z \lor e \notin x \lor e \notin z) )</td>
<td></td>
</tr>
</tbody>
</table>

This enables an application of T-Conflict with clause
\[ F : x = y \cup z \land y \neq \emptyset \lor x \neq z \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y \cup z )</td>
<td>( F )</td>
<td>by Propagate$^+$</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset )</td>
<td>( F )</td>
<td>by Decide</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z )</td>
<td>( F )</td>
<td>by Propagate</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z \land e \in x )</td>
<td>( F, (x = z \lor e \in x \land e \in z) ), ( (x = z \lor e \notin x \land e \notin z) )</td>
<td>by T-Learn</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z \land e \in x \land e \notin z )</td>
<td>( F, (x = z \lor e \in x \land e \in z), (x = z \lor e \notin x \land e \notin z) )</td>
<td>by Propagate</td>
</tr>
</tbody>
</table>

This enables an application of T-Conflict with clause \( x \neq y \).
\[ F : \ x = y \cup z \land y \neq \emptyset \lor x \neq z \]

<table>
<thead>
<tr>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y \cup z )</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset )</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z )</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land e \in x )</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land e \in x \land e \notin z )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
</tr>
<tr>
<td>( F )</td>
</tr>
<tr>
<td>( F )</td>
</tr>
<tr>
<td>( F )</td>
</tr>
<tr>
<td>( F )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>by Propagate (^+)</td>
</tr>
<tr>
<td>by Decide</td>
</tr>
<tr>
<td>by Propagate</td>
</tr>
<tr>
<td>by ( T)-Learn</td>
</tr>
<tr>
<td>by Decide</td>
</tr>
<tr>
<td>by Propagate</td>
</tr>
</tbody>
</table>
EXAMPLE — THEORY OF FINITE SETS

\[ F : x = y \cup z \land y \neq \emptyset \lor x \neq z \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y \cup z )</td>
<td>( F )</td>
<td>by Propagate(^+)</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset )</td>
<td>( F )</td>
<td>by Decide</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z )</td>
<td>( F, (x = z \lor e \in x \lor e \in z), (x = z \lor e \notin x \lor e \notin z) )</td>
<td>by Propagate</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z \land e \in x )</td>
<td>( F, (x = z \lor e \in x \lor e \in z), (x = z \lor e \notin x \lor e \notin z) )</td>
<td>by Decide</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z \land e \in x e \notin z )</td>
<td>( F, (x = z \lor e \in x \lor e \in z), (x = z \lor e \notin x \lor e \notin z) )</td>
<td>by Propagate</td>
</tr>
</tbody>
</table>

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

\[ e \in x \; \ldots \; \Rightarrow e \in y \cup z \; \ldots \; \Rightarrow e \in y \; \ldots \; \Rightarrow e \in \emptyset \Rightarrow \bot \]
**Example — Theory of Finite Sets**

\[ F : \ x = y \cup z \land y \neq \emptyset \lor x \neq z \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y \cup z )</td>
<td>( F )</td>
<td>by Propagate +</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset )</td>
<td>( F )</td>
<td>by Decide</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z )</td>
<td>( F )</td>
<td>by Propagate</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z \land e \in x )</td>
<td>( F, (x = z \lor e \in x \lor e \in z), (x = z \lor e \notin x \lor e \notin z) )</td>
<td>by T-Learn</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z \land e \in x \land e \notin z )</td>
<td>( F, (x = z \lor e \in x \lor e \in z), (x = z \lor e \notin x \lor e \notin z) )</td>
<td>by Decide</td>
</tr>
<tr>
<td>( x = y \cup z \land y = \emptyset \land x \neq z \land e \in x \land e \notin z )</td>
<td>( F, (x = z \lor e \in x \lor e \in z), (x = z \lor e \notin x \lor e \notin z) )</td>
<td>by Propagate</td>
</tr>
</tbody>
</table>

\[ T \text{-solver can make the following deductions at this point:} \]

\[ e \in x \implies e \in y \cup z \implies e \in y \implies e \in \emptyset \implies \bot \]

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

\[ x \neq y \cup z \lor y \neq \emptyset \lor x = z \lor e \notin x \lor e \in z \]
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:

\[
a \approx b + 2 \wedge A = \text{store}(B, a + 1, 4) \wedge A[b + 3] = 2 \lor f(a - 1) \neq f(b + 1)
\]
Many applications give rise to mixed-theory formulas like:

\[
\begin{align*}
  a & \approx b + 2 \land A = \text{store}(B, a + 1, 4) \land \\
  A[b + 3] & = 2 \lor f(a - 1) \neq f(b + 1)
\end{align*}
\]

Solving that formula requires reasoning over

- the theory of linear arithmetic ($\mathcal{T}_{\text{LA}}$)
- the theory of arrays ($\mathcal{T}_A$)
- the theory of uninterpreted functions ($\mathcal{T}_{\text{UF}}$)
Many applications give rise to mixed-theory formulas like:

\[ a \approx b + 2 \land A = \text{store}(B, a + 1, 4) \land A[b + 3] = 2 \lor f(a - 1) \neq f(b + 1) \]

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 $T_{LA} \cup T_A \cup T_{UF}$?
Many applications give rise to **mixed-theory** formulas like:

\[
\begin{align*}
    a & \approx b + 2 \land A = \text{store}(B, a + 1, 4) \land \\
    A[b + 3] &= 2 \lor f(a - 1) \neq f(b + 1)
\end{align*}
\]

Solving that formula requires reasoning over

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

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

Under certain conditions, we can do it with the Nelson-Oppen combination method [NO79, Opp80]
Consider the following set of literals over $T_{\text{LRA}} \cup T_{\text{UF}}$ ($T_{\text{LRA}}$, linear real arithmetic):

\[
\begin{align*}
  f(f(x) - f(y)) &= a \\
  f(0) &> a + 2 \\
  x &= y
\end{align*}
\]
Consider the following set of literals over $T_{LRA} \cup T_{UF}$ ($T_{LRA}$, linear real arithmetic):

\[
\begin{align*}
  f(f(x) - f(y)) &= a \\
  f(0) &> a + 2 \\
  x &= y
\end{align*}
\]

**First step: purify** literals so that each belongs to a single theory
Consider the following set of literals over $T_{LRA} \cup T_{UF}$ ($T_{LRA}$, linear real arithmetic):

\[
\begin{align*}
  f(f(x) - f(y)) & = a \\
  f(0) & > a + 2 \\
  x & = y
\end{align*}
\]

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

\[
\begin{align*}
  f(f(x) - f(y)) = a & \implies f(e_1) = a & \implies f(e_1) = a \\
  e_1 = f(x) - f(y) & \\
  e_2 = f(x) & \\
  e_3 = f(y)
\end{align*}
\]
Consider the following set of literals over $T_{\text{LRA}} \cup T_{\text{UF}}$ ($T_{\text{LRA}}$, linear real arithmetic):

\[
\begin{align*}
  f(f(x) - f(y)) &= a \\
  f(0) &> a + 2 \\
  x &= y
\end{align*}
\]

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

\[
\begin{align*}
  f(0) > a + 2 &\implies f(e_4) > a + 2 &\implies f(e_4) &= e_5 \\
  e_4 = 0 &\implies e_4 = 0 & e_5 > a + 2
\end{align*}
\]
Second step: exchange entailed interface equalities, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(e_1) = a$</td>
<td>$e_2 - e_3 = e_1$</td>
</tr>
<tr>
<td>$f(x) = e_2$</td>
<td>$e_4 = 0$</td>
</tr>
<tr>
<td>$f(y) = e_3$</td>
<td>$e_5 &gt; a + 2$</td>
</tr>
<tr>
<td>$f(e_4) = e_5$</td>
<td></td>
</tr>
<tr>
<td>$x = y$</td>
<td></td>
</tr>
</tbody>
</table>
MOTIVATING EXAMPLE (CONVEX CASE)

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

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(e_1) = a$</td>
<td>$e_2 - e_3 = e_1$</td>
</tr>
<tr>
<td>$f(x) = e_2$</td>
<td>$e_4 = 0$</td>
</tr>
<tr>
<td>$f(y) = e_3$</td>
<td>$e_5 &gt; a + 2$</td>
</tr>
<tr>
<td>$f(e_4) = e_5$</td>
<td></td>
</tr>
<tr>
<td>$x = y$</td>
<td></td>
</tr>
</tbody>
</table>

$L_1 \models_{UF} e_2 = e_3$
**Second step:** exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(e_1) = a$</td>
<td>$e_2 - e_3 = e_1$</td>
</tr>
<tr>
<td>$f(x) = e_2$</td>
<td>$e_4 = 0$</td>
</tr>
<tr>
<td>$f(y) = e_3$</td>
<td>$e_5 &gt; a + 2$</td>
</tr>
<tr>
<td>$f(e_4) = e_5$</td>
<td>$e_2 = e_3$</td>
</tr>
<tr>
<td>$x = y$</td>
<td></td>
</tr>
</tbody>
</table>
**Second step:** exchange entailed *interface equalities*, equalities over shared constants \(e_1, e_2, e_3, e_4, e_5, a\)

\[
\begin{array}{c|c}
L_1 & L_2 \\
\hline
f(e_1) = a & e_2 - e_3 = e_1 \\
f(x) = e_2 & e_4 = 0 \\
f(y) = e_3 & e_5 > a + 2 \\
f(e_4) = e_5 & e_2 = e_3 \\
x = y & \\
\end{array}
\]

\[L_2 \models_{\text{LRA}} e_1 = e_4\]
Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(e_1) = a$</td>
<td>$e_2 - e_3 = e_1$</td>
</tr>
<tr>
<td>$f(x) = e_2$</td>
<td>$e_4 = 0$</td>
</tr>
<tr>
<td>$f(y) = e_3$</td>
<td>$e_5 &gt; a + 2$</td>
</tr>
<tr>
<td>$f(e_4) = e_5$</td>
<td>$e_2 = e_3$</td>
</tr>
<tr>
<td>$x = y$</td>
<td>$e_1 = e_4$</td>
</tr>
</tbody>
</table>
**Second step:** exchange entailed interface equalities, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(e_1) = a$</td>
<td>$e_2 - e_3 = e_1$</td>
</tr>
<tr>
<td>$f(x) = e_2$</td>
<td>$e_4 = 0$</td>
</tr>
<tr>
<td>$f(y) = e_3$</td>
<td>$e_5 &gt; a + 2$</td>
</tr>
<tr>
<td>$f(e_4) = e_5$</td>
<td>$e_2 = e_3$</td>
</tr>
<tr>
<td>$x = y$</td>
<td></td>
</tr>
<tr>
<td>$e_1 = e_4$</td>
<td></td>
</tr>
</tbody>
</table>

$L_1 \models_{UF} a = e_5$
**Second step:** exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(e_1) = a$</td>
<td>$e_2 - e_3 = e_1$</td>
</tr>
<tr>
<td>$f(x) = e_2$</td>
<td>$e_4 = 0$</td>
</tr>
<tr>
<td>$f(y) = e_3$</td>
<td>$e_5 &gt; a + 2$</td>
</tr>
<tr>
<td>$f(e_4) = e_5$</td>
<td>$e_2 = e_3$</td>
</tr>
<tr>
<td>$x = y$</td>
<td>$a = e_5$</td>
</tr>
<tr>
<td>$e_1 = e_4$</td>
<td></td>
</tr>
</tbody>
</table>
**Second step:** exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(e_1) = a$</td>
<td>$e_2 - e_3 = e_1$</td>
</tr>
<tr>
<td>$f(x) = e_2$</td>
<td>$e_4 = 0$</td>
</tr>
<tr>
<td>$f(y) = e_3$</td>
<td>$e_5 &gt; a + 2$</td>
</tr>
<tr>
<td>$f(e_4) = e_5$</td>
<td>$e_2 = e_3$</td>
</tr>
<tr>
<td>$x = y$</td>
<td>$a = e_5$</td>
</tr>
<tr>
<td>$e_1 = e_4$</td>
<td></td>
</tr>
</tbody>
</table>

**Third step:** check for satisfiability locally

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

$L_2 \models_{LRA} \bot$
**Second step:** exchange entailed *interface equalities*, equalities over shared constants \( e_1, e_2, e_3, e_4, e_5, a \)

\[
\begin{array}{c|c}
L_1 & L_2 \\
\hline
f(e_1) = a & e_2 - e_3 = e_1 \\
f(x) = e_2 & e_4 = 0 \\
f(y) = e_3 & e_5 > a + 2 \\
f(e_4) = e_5 & e_2 = e_3 \\
x = y & a = e_5 \\
e_1 = e_4 & \\
\end{array}
\]

**Third step:** check for satisfiability locally

\( L_1 \not\models_{UF} \bot \)
\( L_2 \models_{LRA} \bot \)

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

$$1 \leq x \leq 2$$

$$f(1) = a$$

$$f(2) = f(1) + 3$$

$$a = b + 2$$
Consider the following \textit{unsatisfiable} set of literals over $T_{\text{LIA}} \cup T_{\text{UF}}$ ($T_{\text{LIA}}$, linear integer arithmetic):

\begin{align*}
1 & \leq x \leq 2 \\
f(1) & = a \\
f(2) & = f(1) + 3 \\
a & = b + 2
\end{align*}

\textbf{First step:}\ \textit{purify} literals so that each belongs to a single theory
Consider the following unsatisfiable set of literals over $T_{\text{LIA}} \cup T_{\text{UF}}$ ($T_{\text{LIA}}$, linear integer arithmetic):

\begin{align*}
1 & \leq x \leq 2 \\
f(1) & = a \\
f(2) & = f(1) + 3 \\
a & = b + 2
\end{align*}

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

\[ f(1) = a \implies f(e_1) = a \]

\[ e_1 = 1 \]
Consider the following unsatisfiable set of literals over $T_{\text{LIA}} \cup T_{\text{UF}}$ ($T_{\text{LIA}}$, linear integer arithmetic):

\[
\begin{align*}
1 \leq x & \leq 2 \\
f(1) &= a \\
f(2) &= f(1) + 3 \\
a &= b + 2
\end{align*}
\]

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

\[
\begin{align*}
f(2) &= f(1) + 3 &\implies & e_2 = 2 \\
f(e_2) &= e_3 \\
f(e_1) &= e_4 \\
e_3 &= e_4 + 3
\end{align*}
\]
**Second step:** exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \leq x$</td>
<td>$f(e_1) = a$</td>
</tr>
<tr>
<td>$x \leq 2$</td>
<td>$f(x) = b$</td>
</tr>
<tr>
<td>$e_1 = 1$</td>
<td>$f(e_2) = e_3$</td>
</tr>
<tr>
<td>$a = b + 2$</td>
<td>$f(e_1) = e_4$</td>
</tr>
<tr>
<td>$e_2 = 2$</td>
<td></td>
</tr>
<tr>
<td>$e_3 = e_4 + 3$</td>
<td></td>
</tr>
<tr>
<td>$a = e_4$</td>
<td></td>
</tr>
</tbody>
</table>
Second step: exchange entailed interface equalities over shared constants $x, e_1, a, b, e_2, e_3, e_4$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \leq x$</td>
<td>$f(e_1) = a$</td>
</tr>
<tr>
<td>$x \leq 2$</td>
<td>$f(x) = b$</td>
</tr>
<tr>
<td>$e_1 = 1$</td>
<td>$f(e_2) = e_3$</td>
</tr>
<tr>
<td>$a = b + 2$</td>
<td>$f(e_1) = e_4$</td>
</tr>
<tr>
<td>$e_2 = 2$</td>
<td></td>
</tr>
<tr>
<td>$e_3 = e_4 + 3$</td>
<td></td>
</tr>
<tr>
<td>$a = e_4$</td>
<td></td>
</tr>
</tbody>
</table>

No more entailed equalities, but $L_1 \models_{\text{LIA}} x = e_1 \lor x = e_2$
Second step: exchange entailed interface equalities over shared constants $x, e_1, a, b, e_2, e_3, e_4$

<table>
<thead>
<tr>
<th></th>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 $\leq x$</td>
<td>$f(e_1) = a$</td>
<td></td>
</tr>
<tr>
<td>$x \leq 2$</td>
<td>$f(x) = b$</td>
<td></td>
</tr>
<tr>
<td>$e_1 = 1$</td>
<td>$f(e_2) = e_3$</td>
<td></td>
</tr>
<tr>
<td>$a = b + 2$</td>
<td>$f(e_1) = e_4$</td>
<td></td>
</tr>
<tr>
<td>$e_2 = 2$</td>
<td>$f(e_1) = e_4$</td>
<td></td>
</tr>
<tr>
<td>$e_3 = e_4 + 3$</td>
<td>$a = e_4$</td>
<td></td>
</tr>
</tbody>
</table>

Consider each case of $x = e_1 \lor x = e_2$ separately
Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

<table>
<thead>
<tr>
<th></th>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \leq x$</td>
<td>$f(e_1) = a$</td>
<td></td>
</tr>
<tr>
<td>$x \leq 2$</td>
<td>$f(x) = b$</td>
<td></td>
</tr>
<tr>
<td>$e_1 = 1$</td>
<td></td>
<td>$f(e_2) = e_3$</td>
</tr>
<tr>
<td>$a = b + 2$</td>
<td></td>
<td>$f(e_1) = e_4$</td>
</tr>
<tr>
<td>$e_2 = 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e_3 = e_4 + 3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a = e_4$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Case 1) $x = e_1$
**Second step:** exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \leq x$</td>
<td>$f(e_1) = a$</td>
</tr>
<tr>
<td>$x \leq 2$</td>
<td>$f(x) = b$</td>
</tr>
<tr>
<td>$e_1 = 1$</td>
<td>$f(e_2) = e_3$</td>
</tr>
<tr>
<td>$a = b + 2$</td>
<td>$f(e_1) = e_4$</td>
</tr>
<tr>
<td>$e_2 = 2$</td>
<td>$x = e_1$</td>
</tr>
<tr>
<td>$e_3 = e_4 + 3$</td>
<td></td>
</tr>
<tr>
<td>$a = e_4$</td>
<td></td>
</tr>
<tr>
<td>$x = e_1$</td>
<td></td>
</tr>
</tbody>
</table>
**Second step:** exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \leq x$</td>
<td>$f(e_1) = a$</td>
</tr>
<tr>
<td>$x \leq 2$</td>
<td>$f(x) = b$</td>
</tr>
<tr>
<td>$e_1 = 1$</td>
<td>$f(e_2) = e_3$</td>
</tr>
<tr>
<td>$a = b + 2$</td>
<td>$f(e_1) = e_4$</td>
</tr>
<tr>
<td>$e_2 = 2$</td>
<td>$x = e_1$</td>
</tr>
<tr>
<td>$e_3 = e_4 + 3$</td>
<td></td>
</tr>
<tr>
<td>$a = e_4$</td>
<td></td>
</tr>
<tr>
<td>$x = e_1$</td>
<td></td>
</tr>
</tbody>
</table>

$L_2 \models_{UF} a = b$, which entails $\bot$ when sent to $L_1$
**Second step**: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

<table>
<thead>
<tr>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \leq x$</td>
<td>$f(e_1) = a$</td>
</tr>
<tr>
<td>$x \leq 2$</td>
<td>$f(x) = b$</td>
</tr>
<tr>
<td>$e_1 = 1$</td>
<td>$f(e_2) = e_3$</td>
</tr>
<tr>
<td>$a = b + 2$</td>
<td>$f(e_1) = e_4$</td>
</tr>
<tr>
<td>$e_2 = 2$</td>
<td></td>
</tr>
<tr>
<td>$e_3 = e_4 + 3$</td>
<td></td>
</tr>
<tr>
<td>$a = e_4$</td>
<td></td>
</tr>
</tbody>
</table>
**Second step:** exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \leq x$</td>
<td>$f(e_1) = a$</td>
</tr>
<tr>
<td>$x \leq 2$</td>
<td>$f(x) = b$</td>
</tr>
<tr>
<td>$e_1 = 1$</td>
<td>$f(e_2) = e_3$</td>
</tr>
<tr>
<td>$a = b + 2$</td>
<td>$f(e_1) = e_4$</td>
</tr>
<tr>
<td>$e_2 = 2$</td>
<td></td>
</tr>
<tr>
<td>$e_3 = e_4 + 3$</td>
<td></td>
</tr>
<tr>
<td>$a = e_4$</td>
<td></td>
</tr>
</tbody>
</table>

Case 2) $x = e_2$
Second step: exchange entailed interface equalities over shared constants $x, e_1, a, b, e_2, e_3, e_4$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \leq x$</td>
<td>$f(e_1) = a$</td>
</tr>
<tr>
<td>$x \leq 2$</td>
<td>$f(x) = b$</td>
</tr>
<tr>
<td>$e_1 = 1$</td>
<td>$f(e_2) = e_3$</td>
</tr>
<tr>
<td>$a = b + 2$</td>
<td>$f(e_1) = e_4$</td>
</tr>
<tr>
<td>$e_2 = 2$</td>
<td>$x = e_2$</td>
</tr>
<tr>
<td>$e_3 = e_4 + 3$</td>
<td></td>
</tr>
<tr>
<td>$a = e_4$</td>
<td></td>
</tr>
<tr>
<td>$x = e_2$</td>
<td></td>
</tr>
</tbody>
</table>
**Second step:** exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \leq x$</td>
<td>$f(e_1) = a$</td>
</tr>
<tr>
<td>$x \leq 2$</td>
<td>$f(x) = b$</td>
</tr>
<tr>
<td>$e_1 = 1$</td>
<td>$f(e_2) = e_3$</td>
</tr>
<tr>
<td>$a = b + 2$</td>
<td>$f(e_1) = e_4$</td>
</tr>
<tr>
<td>$e_2 = 2$</td>
<td>$x = e_2$</td>
</tr>
<tr>
<td>$e_3 = e_4 + 3$</td>
<td></td>
</tr>
<tr>
<td>$a = e_4$</td>
<td></td>
</tr>
<tr>
<td>$x = e_2$</td>
<td></td>
</tr>
</tbody>
</table>

$L_2 \models_{UF} e_3 = b$, which entails $\perp$ when sent to $L_1$
· For $i = 1, 2$, let $T_i$ be a first-order theory of signature $\Sigma_i$ (set of function and predicate symbols in $T_i$ other than $=$)

· Let $T = T_1 \cup T_2$

· Let $C$ 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 $\Sigma_i$ (set of function and predicate symbols in $T_i$ other than $=$).

Let $T = T_1 \cup T_2$.

Let $C$ 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) $(\Sigma_i \cup C)$-literals.
· For $i = 1, 2$, let $T_i$ be a first-order theory of signature $\Sigma_i$ (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]:

Input:

\[ L_1, L_2 \]

with \( L_i \) finite set of ground literals

Output:

sat or unsat

1. Guess an arrangement \( A \), i.e., a set of equalities and disequalities over \( C \) such that \( c = d \) or \( c \neq d \) for all \( c, d \in C \).

2. If \( L_i[A] \) is \( T_i \)-unsatisfiable for \( i = 1 \) or \( i = 2 \), return unsat.

3. Otherwise, return sat.
Bare-bones, non-deterministic, non-incremental version

[Opp80, Rin96, TH96]:

Input: \( L_1 \cup L_2 \) with \( L_i \) finite set of ground \((\Sigma_i \cup C)\)-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 \((\Sigma_i \cup C)\)-literals

Output: sat or unsat

1. Guess an arrangement \( A \), i.e., a set of equalities and disequalities over \( C \) such that

\[
    c = d \in A \text{ or } c \neq d \in A \quad \text{for all } c, d \in C
\]
The Nelson-Oppen Method

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

[Opp80, Rin96, TH96]:

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

1. Guess an arrangement \( A \), i.e., a set of equalities and disequalities over \( C \) such that
\[
c = d \in A \text{ or } c \neq d \in A \text{ for all } c, d \in C
\]
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
   \[ c = d \in A \text{ or } c \neq d \in A \text{ for all } c, d \in \mathcal{C} \]

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)
CORRECTNESS OF THE NO METHOD

**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

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

**Note:** With convex theories, arrangements do not need to be guessed, they can 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 $T$. 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]
**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 \textit{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 \textit{cooperatively} into a single SMT solver for $T = T_1 \cup \cdots \cup T_n$?

\textbf{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
MODELING DPLL\( (t_1, \ldots, t_n) \) ABSTRACTLY

- Let \( n = 2 \), for simplicity
- Let \( T_i \) be of signature \( \Sigma_i \) 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 \( \mathcal{M}_i \triangleq \{ (\Sigma_i \cup \mathcal{C})\text{-}literals of } \mathcal{M} \text{ and their complement} \}
- Let \( \mathbf{I}(\mathcal{M}) \triangleq \{ c = d \mid c, d \text{ occur in } \mathcal{C}, \mathcal{M}_1 \text{ and } \mathcal{M}_2 \} \cup \{ c \neq d \mid c, d \text{ occur in } \mathcal{C}, \mathcal{M}_1 \text{ and } \mathcal{M}_2 \} \)
  (interface literals)
Propagate, Conflict, Explain, Backjump, Fail (unchanged)
Propagate, Conflict, Explain, Backjump, Fail (unchanged)

Decide

\[
\frac{l \in \text{Lit}(F) \cup \text{I}(M) \quad l, \overline{l} \notin M}{M := M \cdot l}
\]

Only change: decide on interface equalities as well
Propagate, Conflict, Explain, Backjump, Fail (unchanged)

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

Only change: decide on interface equalities as well

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

Only change: propagate interface equalities as well, but reason locally in each \( T_i \)
**T-Conflict**

\[ C = \text{no} \quad l_1, \ldots, l_n \in M \quad l_1, \ldots, l_n \models_{T_i} \bot \quad i \in \{1, 2\} \]

\[ C := \overline{l}_1 \lor \cdots \lor \overline{l}_n \]

**T-Explain**

\[ C = l \lor D \quad \overline{l}_1, \ldots, \overline{l}_n \models_{T_i} \overline{l} \quad i \in \{1, 2\} \quad \overline{l}_1, \ldots, \overline{l}_n \prec_M \overline{l} \]

\[ C := l_1 \lor \cdots \lor l_n \lor D \]

Only change: reason locally in each \( T_i \)
**Abstract DPLL Modulo Multiple Theories**

**T-Conflict**

\[ C \equiv \text{no} \quad l_1, \ldots, l_n \in M \quad l_1, \ldots, l_n \models_{T_i} \bot \quad i \in \{1, 2\} \]

\[ C := \overline{l}_1 \lor \cdots \lor \overline{l}_n \]

**T-Explain**

\[ C = l \lor D \quad \overline{l}_1, \ldots, \overline{l}_n \models_{T_i} \overline{l} \quad i \in \{1, 2\} \quad \overline{l}_1, \ldots, \overline{l}_n \prec_M \overline{l} \]

\[ C := l_1 \lor \cdots \lor l_n \lor D \]

**Only change:** reason locally in each \( T_i \)

**I-Learn**

\[ \models_{T_i} l_1 \lor \cdots \lor l_n \quad l_1, \ldots, l_n \in M|_i \cup I(M) \quad i \in \{1, 2\} \]

\[ F := F \cup \{l_1 \lor \cdots \lor l_n\} \]

**New rule:** for entailed disjunctions of interface literals
\[ F := \begin{cases} 0 & f(e_1) = a \land e_2 - e_3 = e_1 \\ 1 & f(x) = e_2 \land e_4 = 0 \\ 2 & f(y) = e_3 \land e_5 > a + 2 \\ 3 & f(e_4) = e_5 \land x = y \\ 4 & \end{cases} \]
EXAMPLE — CONVEX THEORIES

\[ F := \begin{cases} 
0 & f(e_1) = a \land e_2 - e_3 = e_1 \\
1 & f(x) = e_2 \land e_4 = 0 \\
2 & f(y) = e_3 \land e_5 > a + 2 \\
3 & f(e_4) = e_5 \\
4 & x = y \land 
\end{cases} \]

\[ \begin{align*}
& e_2 = e_3 \\
& e_1 = e_4 \\
& a = e_5 
\end{align*} \]

M | F | C | rule
---|---|---|---
F | no
EXAMPLE — CONVEX THEORIES

\[ F := \begin{array}{c}
\begin{aligned}
&0: f(e_1) = a \land e_2 - e_3 = e_1 \\
&1: f(x) = e_2 \land e_4 = 0 \\
&2: f(y) = e_3 \land e_5 > a + 2 \\
&3: f(e_4) = e_5 \\
&4: x = y \\
&5: e_2 = e_3 \\
&6: e_1 = e_4 \\
&7: a = e_5 \\
&8: 0 \leq 1 \leq 2 \leq 3 \leq 4 \leq 5 \leq 6 \leq 7 \\
\end{aligned}
\end{array} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
\[ F := \left\{ \begin{array}{l}
\begin{align*}
f(e_1) &= a \\
e_2 - e_3 &= e_1
\end{align*}
\right\} \land
\left\{ \begin{array}{l}
\begin{align*}
f(x) &= e_2 \\
e_4 &= 0
\end{align*}
\right\} \land
\left\{ \begin{array}{l}
\begin{align*}
f(y) &= e_3 \\
e_5 &= a + 2
\end{align*}
\right\} \land
\left\{ \begin{array}{l}
\begin{align*}
f(e_4) &= e_5 \\
x &= y
\end{align*}
\right\} \land
\left\{ \begin{array}{l}
\begin{align*}
e_2 &= e_3 \\
e_1 &= e_4 \\
a &= e_5
\end{align*}
\right\}
\right\}
\]
EXAMPLE — CONVEX THEORIES

\[ F := \begin{cases} 0 & f(e_1) = a \land e_2 - e_3 = e_1 \\ 1 & f(x) = e_2 \land e_4 = 0 \\ 2 & f(y) = e_3 \land e_5 > a + 2 \\ 3 & f(e_4) = e_5 \land x = y \land e_2 = e_3 \\ 4 & a = e_5 \end{cases} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7</td>
<td>F</td>
<td>no</td>
<td>by Propagate$^+$</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>F</td>
<td>no</td>
<td>by $T$-Propagate $(1, 2, 4 \models_{UF} 8)$</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9</td>
<td>F</td>
<td>no</td>
<td>by $T$-Propagate $(5, 6, 8 \models_{LRA} 9)$</td>
</tr>
</tbody>
</table>
$F := \begin{align*}
\begin{array}{c}
\text{0} \\
\text{1} \\
\text{2} \\
\text{3} \\
\text{4} \\
\text{5} \\
\text{6} \\
\text{7} \\
\text{8} \\
\text{9} \\
\text{10}
\end{array}
\begin{array}{c}
f(e_1) = a & \land & e_2 - e_3 = e_1 \\
f(x) = e_2 & \land & e_4 = 0 \\
f(y) = e_3 & \land & e_5 > a + 2 \\
f(e_4) = e_5 & \land & x = y \\
e_2 = e_3 & \land & e_1 = e_4 \\
a = e_5
\end{array}
\end{align*}$

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7</td>
<td>F</td>
<td>no</td>
<td>by Propagate$^+$</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>F</td>
<td>no</td>
<td>by $T$-Propagate $(1, 2, 4 \models_{\text{UF}} 8)$</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9</td>
<td>F</td>
<td>no</td>
<td>by $T$-Propagate $(5, 6, 8 \models_{\text{LRA}} 9)$</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10</td>
<td>F</td>
<td>no</td>
<td>by $T$-Propagate $(0, 3, 9 \models_{\text{UF}} 10)$</td>
</tr>
</tbody>
</table>
### Example — Convex Theories

\[ F := \begin{align*}
& f(e_1) = a \quad \land \quad f(x) = e_2 \quad \land \quad f(y) = e_3 \quad \land \quad f(e_4) = e_5 \quad \land \quad x = y \quad \land \\
& e_2 - e_3 = e_1 \quad \land \quad e_4 = 0 \quad \land \quad e_5 > a + 2
\end{align*} \]

\[ e_2 = e_3 \quad \land \quad e_1 = e_4 \quad \land \quad a = e_5 \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7</td>
<td>F</td>
<td>no</td>
<td>by Propagate$^+$</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>F</td>
<td>no</td>
<td>by (T)-Propagate ( (1, 2, 4 \models_{UF} 8) )</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9</td>
<td>F</td>
<td>no</td>
<td>by (T)-Propagate ( (5, 6, 8 \models_{LRA} 9) )</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10</td>
<td>F</td>
<td>no</td>
<td>by (T)-Propagate ( (0, 3, 9 \models_{UF} 10) )</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10</td>
<td>F</td>
<td>( \overline{7} \lor \overline{10} )</td>
<td>by (T)-Conflict ( (7, 10 \models_{LRA} \bot) )</td>
</tr>
</tbody>
</table>
### Example — Convex Theories

\[
F := \begin{align*}
&f(e_1) = a \land e_2 - e_3 = e_1 \land e_2 - e_3 = e_1 \\
&f(x) = e_2 \land e_4 = 0 \land e_2 = e_3 \\
&f(y) = e_3 \land e_5 > a + 2 \land e_1 = e_4 \\
&f(e_4) = e_5 \land x = y \land e_1 = e_4 \land a = e_5
\end{align*}
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\overline{0})</td>
<td>(\overline{0})</td>
<td>by Propagate(^+)</td>
</tr>
<tr>
<td>01234567</td>
<td>(\overline{F})</td>
<td>no</td>
<td>by (T)-Propagate ((1, 2, 4 \models_{UF} 8))</td>
</tr>
<tr>
<td>012345678</td>
<td>(\overline{F})</td>
<td>no</td>
<td>by (T)-Propagate ((5, 6, 8 \models_{LRA} 9))</td>
</tr>
<tr>
<td>0123456789</td>
<td>(\overline{F})</td>
<td>no</td>
<td>by (T)-Propagate ((0, 3, 9 \models_{UF} 10))</td>
</tr>
<tr>
<td>012345678910</td>
<td>(\overline{F})</td>
<td>(\overline{F})</td>
<td>by (T)-Conflict ((7, 10 \models_{LRA} \bot))</td>
</tr>
<tr>
<td>012345678910</td>
<td>(\overline{F})</td>
<td>(\overline{F})</td>
<td>by Fail</td>
</tr>
<tr>
<td>fail</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### M F C rule

- **M**, **F**, and **C** are placeholders for various elements.
- The rules for propagation and conflict resolution are indicated in the `rule` column.
- The example showcases how convex theories are handled in a formal logic framework.
\[ F := \begin{cases} \left\{ \begin{array}{l} 0 \\ f(e_1) = a \wedge f(x) = b \wedge f(e_2) = e_3 \wedge f(e_1) = e_4 \wedge \\ 1 \leq x \wedge x \leq 2 \wedge e_1 = 1 \wedge a = b + 2 \wedge e_2 = 2 \wedge e_3 = e_4 + 3 \end{array} \right. \\ \left\{ \begin{array}{l} 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \right. \\ \left\{ \begin{array}{l} 10 \\ 11 \\ 12 \\ 13 \end{array} \right. \\ \left\{ \begin{array}{l} a = e_4 \\ x = e_1 \\ x = e_2 \\ a = b \end{array} \right. \end{cases} \]
EXAMPLE — NON-CONVEX THEORIES

\[
F := \begin{cases}
  0 : f(e_1) = a \land f(x) = b \land 1 \leq x \land x \leq 2 \land e_1 = 1 \\
  1 : f(e_2) = e_3 \land a = b + 2 \\
  2 : f(e_1) = e_4 \land e_2 = 2 \\
  3 : e_3 = e_4 + 3
\end{cases}
\]

\[
\begin{align*}
  a &= e_4 \\
  x &= e_1 \\
  x &= e_2 \\
  a &= b
\end{align*}
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>no</td>
</tr>
</tbody>
</table>
**EXAMPLE — NON-CONVEX THEORIES**

\[
F := \begin{cases} 
0 & f(e_1) = a \land f(x) = b \land f(e_2) = e_3 \land f(e_1) = e_4 \land 1 \leq x \land x \leq 2 \land e_1 = 1 \land a = b + 2 \land e_2 = 2 \land e_3 = e_4 + 3 \\
1 & a = e_4 \\
2 & x = e_1 \\
3 & x = e_2 \\
4 & a = b \\
5 & a = e_4 \\
6 & x = e_1 \\
7 & x = e_2 \\
8 & a = b \\
9 & a = e_4 \\
10 & x = e_1 \\
11 & x = e_2 \\
12 & a = b \\
13 & a = e_4 \\
\end{cases}
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>no</td>
<td>Propagate^+</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>no</td>
<td>Propagate^+</td>
</tr>
</tbody>
</table>
EXAMPLE — NON-CONVEX THEORIES

\[ F := \begin{cases} 0 & f(e_1) = a \land f(x) = b \land 1 \leq x \land x \leq 2 \land e_1 = 1 \land a = b + 2 \land e_2 = 2 \land e_3 = e_4 + 3 \\ 1 & f(e_2) = e_3 \\ 2 & f(e_1) = e_4 \\ 3 & 4 \end{cases} \]

\[ a = e_4 \quad x = e_1 \quad x = e_2 \quad a = b \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( F )</td>
<td>no</td>
<td>Propagate (^+)</td>
</tr>
<tr>
<td>0 \ldots 9</td>
<td>( F )</td>
<td>no</td>
<td>\text{by } T-Propagate ((0, 3 \models_{\text{UF}} 10))</td>
</tr>
<tr>
<td>0 \ldots 9 10</td>
<td>( F )</td>
<td>no</td>
<td>T-Propagate ((0, 3 \models_{\text{UF}} 10))</td>
</tr>
</tbody>
</table>
EXAMPLE — NON-CONVEX THEORIES

\[ F := \begin{align*}
  f(e_1) &= a & f(x) &= b & f(e_2) &= e_3 & f(e_1) &= e_4 \\
  1 &\leq x & x &\leq 2 & e_1 &= 1 & a &= b + 2 & e_2 &= 2 & e_3 &= e_4 + 3
\end{align*} \]

\[ \begin{aligned}
  a &= e_4 & x &= e_1 & x &= e_2 & a &= b \\
  10 & & 11 & & 12 & & 13
\end{aligned} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( F )</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>0 \cdots 9</td>
<td>( F )</td>
<td>no</td>
<td>by Propagate +</td>
</tr>
<tr>
<td>0 \cdots 9 10</td>
<td>( F )</td>
<td>no</td>
<td>by T-Propagate ( (0, 3 \models_{UF} 10) )</td>
</tr>
<tr>
<td>0 \cdots 9 10</td>
<td>( F, \overline{4} \lor \overline{5} \lor 11 \lor 12 )</td>
<td>no</td>
<td>by I-Learn ( (\models_{LIA} \overline{4} \lor \overline{5} \lor 11 \lor 12) )</td>
</tr>
</tbody>
</table>
**EXAMPLE — NON-CONVEX THEORIES**

\[ F := \{ f(e_1) = a \land f(x) = b \land f(e_2) = e_3 \land f(e_1) = e_4 \land \begin{align*} &1 \leq x \land x \leq 2 \land e_1 = 1 \land a = b + 2 \land e_2 = 2 \land e_3 = e_4 + 3 \end{align*} \} \]

\[
\begin{align*}
& a = e_4 \\
& x = e_1 \\
& x = e_2 \\
& a = b
\end{align*}
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 \ldots 9</td>
<td>F</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>0 \ldots 9 10</td>
<td>F</td>
<td>no</td>
<td>by Propagate$^+$</td>
</tr>
<tr>
<td>0 \ldots 9 10</td>
<td>F, 4 \lor 5 \lor 11 \lor 12</td>
<td>no</td>
<td>by $T$-Propagate ($0, 3 \models_{UF} 10$)</td>
</tr>
<tr>
<td>0 \ldots 9 10 \bullet 11</td>
<td>F, 4 \lor 5 \lor 11 \lor 12</td>
<td>no</td>
<td>by I-Learn ($\models_{LIA} 4 \lor 5 \lor 11 \lor 12$)</td>
</tr>
<tr>
<td>0 \ldots 9 10 \bullet 11</td>
<td>F, 4 \lor 5 \lor 11 \lor 12</td>
<td>no</td>
<td>by Decide</td>
</tr>
</tbody>
</table>
EXAMPLE — NON-CONVEX THEORIES

\[ F := f(e_1) = a \land f(x) = b \land f(e_2) = e_3 \land f(e_1) = e_4 \land 1 \leq x \land x \leq 2 \land e_1 = 1 \land a = b + 2 \land e_2 = 2 \land e_3 = e_4 + 3 \]

\[
\begin{align*}
& a = e_4 \quad x = e_1 \quad x = e_2 \quad a = b \\
& 10 \quad 11 \quad 12 \quad 13
\end{align*}
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 9</td>
<td>F</td>
<td>no</td>
<td>by \text{Propagate}^+</td>
</tr>
<tr>
<td>0 9 10</td>
<td>F</td>
<td>no</td>
<td>by \text{T-Propagate} ((0, 3 \models_{\text{UF}} 10))</td>
</tr>
<tr>
<td>0 9 10</td>
<td>F, \overline{4} \lor \overline{5} \lor 11 \lor 12</td>
<td>no</td>
<td>by \text{I-Learn} ((\models_{\text{LIA}} \overline{4} \lor \overline{5} \lor 11 \lor 12))</td>
</tr>
<tr>
<td>0 9 10 11</td>
<td>F, \overline{4} \lor \overline{5} \lor 11 \lor 12</td>
<td>no</td>
<td>by \text{Decide}</td>
</tr>
<tr>
<td>0 9 10 11 13</td>
<td>F, \overline{4} \lor \overline{5} \lor 11 \lor 12</td>
<td>no</td>
<td>by \text{T-Propagate} ((0, 1, 11 \models_{\text{UF}} 13))</td>
</tr>
</tbody>
</table>
Example — Non-Convex Theories

\[ F := \begin{cases} \begin{align*} f(e_1) &= a & & (0) \\ f(x) &= b & & (1) \\ f(e_2) &= e_3 & & (2) \\ f(e_1) &= e_4 & & (3) \\ 1 \leq x & & (4) \\ x \leq 2 & & (5) \\ e_1 &= 1 & & (6) \\ a &= b + 2 & & (7) \\ e_2 &= 2 & & (8) \\ e_3 &= e_4 + 3 & & (9) \end{align*} \end{cases} \]

\begin{align*} a &= e_4 & & (10) \\ x &= e_1 & & (11) \\ x &= e_2 & & (12) \\ a &= b & & (13) \end{align*}

\begin{tabular}{llll}
M & F & C & rule \\
\hline
10 & & & \\
11 & & & \\
12 & & & \\
13 & & & \\
\end{tabular}

\textbf{M} \textbf{F} \textbf{C} \textbf{rule} \\
0 \ldots 9 & \textbf{F} & \text{no} & \text{by Propagate}^+ \\
0 \ldots 9 10 & \textbf{F} & \text{no} & \text{by T-Propagate} (0, 3 \models_{\text{UF}} 10) \\
0 \ldots 9 10 & \textbf{F}, \overline{4} \lor \overline{5} \lor 11 \lor 12 & \text{no} & \text{by I-Learn} (\models_{\text{LIA}} \overline{4} \lor \overline{5} \lor 11 \lor 12) \\
0 \ldots 9 10 \bullet 11 & \textbf{F}, \overline{4} \lor \overline{5} \lor 11 \lor 12 & \text{no} & \text{by Decide} \\
0 \ldots 9 10 \bullet 11 13 & \textbf{F}, \overline{4} \lor \overline{5} \lor 11 \lor 12 & \text{no} & \text{by T-Propagate} (0, 1, 11 \models_{\text{UF}} 13) \\
0 \ldots 9 10 \bullet 11 13 & \textbf{F}, \overline{4} \lor \overline{5} \lor 11 \lor 12 \lor 7 \lor 13 & \text{by T-Conflict} (7, 13 \models_{\text{UF}} \bot) \\
\end{tabular}
**EXAMPLE — NON-CONVEX THEORIES**

\[ F := \left\{ \begin{array}{l}
0 \quad f(e_1) = a \land f(x) = b \land 1 \leq x \land x \leq 2 \land e_1 = 1 \land a = b + 2 \land e_2 = 2 \land e_3 = e_4 + 3 \\
1 \quad f(e_2) = e_3 \\
2 \quad f(e_1) = e_4 \\
3 \quad e = e_4 + 3
\end{array} \right. \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0 \ldots 9)</td>
<td>(F)</td>
<td>no</td>
<td>by <strong>Propagate</strong>$^+$</td>
</tr>
<tr>
<td>(0 \ldots 9)</td>
<td>(F)</td>
<td>no</td>
<td>by <strong>T-Propagate</strong> ((0, 3 \models_{\text{UF}} 10))</td>
</tr>
<tr>
<td>(0 \ldots 9)</td>
<td>(F, \overline{4} \lor 5 \lor 11 \lor 12)</td>
<td>no</td>
<td>by <strong>I-Learn</strong> ((\models_{\text{LIA}} \overline{4} \lor 5 \lor 11 \lor 12))</td>
</tr>
<tr>
<td>(0 \ldots 9)</td>
<td>(F, \overline{4} \lor 5 \lor 11 \lor 12)</td>
<td>no</td>
<td>by <strong>Decide</strong></td>
</tr>
<tr>
<td>(0 \ldots 9)</td>
<td>(\overline{11})</td>
<td>no</td>
<td>by <strong>T-Propagate</strong> ((0, 1, 11 \models_{\text{UF}} 13))</td>
</tr>
<tr>
<td>(0 \ldots 9)</td>
<td>(F, \overline{4} \lor 5 \lor 11 \lor 12)</td>
<td>(7 \lor 13)</td>
<td>by <strong>T-Conflict</strong> ((7, 13 \models_{\text{UF}} \bot))</td>
</tr>
<tr>
<td>(0 \ldots 9)</td>
<td>(13)</td>
<td>no</td>
<td>by <strong>Backjump</strong></td>
</tr>
</tbody>
</table>
### Example — Non-Convex Theories

\[
F := \left\{ \begin{array}{ll}
0 & \text{if } f(e_1) = a \\
1 & \text{if } f(x) = b \\
2 & \text{if } f(e_2) = e_3 \\
3 & \text{if } f(e_1) = e_4 \\
4 & \text{if } 1 \leq x \leq 2 \\
5 & \text{if } e_1 = 1 \\
6 & \text{if } a = b + 2 \\
7 & \text{if } e_2 = 2 \\
8 & \text{if } e_3 = e_4 + 3 \\
9 & \text{otherwise}
\end{array} \right.
\]

\[
\begin{align*}
 a &= e_4 & x &= e_1 & x &= e_2 & a &= b \\
10 & & 11 & & 12 & & 13
\end{align*}
\]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(F)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>0 \ldots 9</td>
<td>(F)</td>
<td>no by Propagate(^+)</td>
<td></td>
</tr>
<tr>
<td>0 \ldots 9 10</td>
<td>(F)</td>
<td>no by T-Propagate (0, 3 (\models_{\text{UF}} 10))</td>
<td></td>
</tr>
<tr>
<td>0 \ldots 9 10</td>
<td>(\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no by I-Learn ((\models_{\text{LIA}} \overline{4} \lor \overline{5} \lor 11 \lor 12))</td>
<td></td>
</tr>
<tr>
<td>0 \ldots 9 10 (\bullet) 11</td>
<td>(\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no by Decide</td>
<td></td>
</tr>
<tr>
<td>0 \ldots 9 10 (\bullet) 11 13</td>
<td>(\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no by T-Propagate (0, 1, 11 (\models_{\text{UF}} 13))</td>
<td></td>
</tr>
<tr>
<td>0 \ldots 9 10 (\bullet) 11 13</td>
<td>(7 \lor 13)</td>
<td>by T-Conflict (7, 13 (\models_{\text{UF}} \perp))</td>
<td></td>
</tr>
<tr>
<td>0 \ldots 9 10 \overline{13}</td>
<td>(\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no by Backjump</td>
<td></td>
</tr>
<tr>
<td>0 \ldots 9 10 \overline{13} 11</td>
<td>(\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no by T-Propagate (0, 1, \overline{13} (\models_{\text{UF}} \overline{11}))</td>
<td></td>
</tr>
</tbody>
</table>
\[ F := \begin{cases} f(e_1) = a & \land \ f(x) = b & \land \ f(e_2) = e_3 & \land \ f(e_1) = e_4 & \land \\ 1 \leq x & \land \ x \leq 2 & \land \ e_1 = 1 & \land \ a = b + 2 & \land \ e_2 = 2 & \land \ e_3 = e_4 + 3 \end{cases} \]

\[
\begin{align*}
M & F & C & \text{rule} \\
\hline
0 \ldots 9 & F & \text{no} & \\
0 \ldots 9 10 & F & \text{no} & \text{by Propagate}^+ \\
0 \ldots 9 10 & F, \overline{4} \lor \overline{5} \lor 11 \lor 12 & \text{no} & \text{by T-Propagate (}0, 3 \models_{UF} 10\text{)} \\
0 \ldots 9 10 \bullet 11 & F, \overline{4} \lor \overline{5} \lor 11 \lor 12 & \text{no} & \text{by I-Learn (}4 \models_{LIA} 4 \lor 5 \lor 11 \lor 12\text{)} \\
0 \ldots 9 10 \bullet 11 13 & F, \overline{4} \lor \overline{5} \lor 11 \lor 12 & \text{no} & \text{by Decide} \\
0 \ldots 9 10 \bullet 11 13 & F, 4 \lor 5 \lor 11 \lor 12 & \overline{7} \lor 13 & \text{by T-Conflict (}7, 13 \models_{UF} \bot\text{)} \\
0 \ldots 9 10 \overline{11} & F, 4 \lor 5 \lor 11 \lor 12 & \text{no} & \text{by Backjump} \\
0 \ldots 9 10 \overline{13} & F, 4 \lor 5 \lor 11 \lor 12 & \text{no} & \text{by T-Propagate (}0, 1, 13 \models_{UF} 11\text{)} \\
0 \ldots 9 10 \overline{13} 11 & F, 4 \lor 5 \lor 11 \lor 12 & \text{no} & \text{by Propagate} \\
0 \ldots 9 10 \overline{13} 11 12 & F, 4 \lor 5 \lor 11 \lor 12 & \text{no} & \text{by Propagate} \\
\end{align*}
\]
\[ F := \begin{align*}
\begin{array}{c}
\text{0} & f(e_1) = a \land f(x) = b \land 1 \leq x \land x \leq 2 \land e_1 = 1 \land a = b + 2 \\
\text{1} & f(e_2) = e_3 \land 2 \land e_2 = 2 \land e_3 = e_4 + 3 \\
\text{2} & f(e_1) = e_4 \\
\text{3} & \end{array}
\end{align*} \]

\[ \begin{array}{c}
\text{4} \ a = e_4 \\
\text{5} \ x = e_1 \\
\text{6} \ x = e_2 \\
\text{7} \ a = b \\
\end{array} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 \ldots 9</td>
<td>F</td>
<td>no</td>
<td>Propagate$^+$</td>
</tr>
<tr>
<td>0 \ldots 9 \ 10</td>
<td>F</td>
<td>no</td>
<td>T-Propagate ((0, 3 \models_{UF} 10))</td>
</tr>
<tr>
<td>0 \ldots 9 \ 10</td>
<td>F, (4 \lor 5 \lor 11 \lor 12)</td>
<td>no</td>
<td>I-Learn ((\models_{LIA} 4 \lor 5 \lor 11 \lor 12))</td>
</tr>
<tr>
<td>0 \ldots 9 \ 10 \ 11</td>
<td>(\mathbf{F}, 4 \lor 5 \lor 11 \lor 12)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>0 \ldots 9 \ 10 \ 11 \ 13</td>
<td>(\mathbf{F}, 4 \lor 5 \lor 11 \lor 12)</td>
<td>no</td>
<td>T-Propagate ((0, 1, 11 \models_{UF} 13))</td>
</tr>
<tr>
<td>0 \ldots 9 \ 10 \ 11 \ 13</td>
<td>(\mathbf{F}, 4 \lor 5 \lor 11 \lor 12)</td>
<td>(\mathbf{7} \lor \mathbf{13})</td>
<td>T-Conflict ((7, 13 \models_{UF} \bot))</td>
</tr>
<tr>
<td>0 \ldots 9 \ 10 \ 13</td>
<td>(\mathbf{F}, 4 \lor 5 \lor 11 \lor 12)</td>
<td>no</td>
<td>Backjump</td>
</tr>
<tr>
<td>0 \ldots 9 \ 10 \ 13 \ 11</td>
<td>(\mathbf{F}, 4 \lor 5 \lor 11 \lor 12)</td>
<td>no</td>
<td>T-Propagate ((0, 1, 13 \models_{UF} 11))</td>
</tr>
<tr>
<td>0 \ldots 9 \ 10 \ 13</td>
<td>(\mathbf{F}, 4 \lor 5 \lor 11 \lor 12)</td>
<td>no</td>
<td>Propagate (exercise)</td>
</tr>
<tr>
<td>0 \ldots 9 \ 10 \ 11 \ 12</td>
<td>(\mathbf{F}, 4 \lor 5 \lor 11 \lor 12)</td>
<td>\ldots</td>
<td></td>
</tr>
</tbody>
</table>
EXAMPLE — NON-CONVEX THEORIES

\[ F := \begin{cases} \begin{align*} f(e_1) &= a \land f(x) &= b \land f(e_2) &= e_3 \land f(e_1) &= e_4 \land \\ 1 &\leq x \land x &\leq 2 \land e_1 &= 1 \land a &= b + 2 \land e_2 &= 2 \land e_3 &= e_4 + 3 \end{align*} \end{cases} \]

\[ \begin{align*} a &= e_4 \quad x &= e_1 \quad x &= e_2 \quad a &= b \end{align*} \]

<table>
<thead>
<tr>
<th>M</th>
<th>F</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 \ldots 9</td>
<td>F</td>
<td>no</td>
<td>by Propagate$^+$</td>
</tr>
<tr>
<td>0 \ldots 9 10</td>
<td>F</td>
<td>no</td>
<td>by T-Propagate ((0, 3 \models_{UF} 10))</td>
</tr>
<tr>
<td>0 \ldots 9 10</td>
<td>F, (\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no</td>
<td>by I-Learn ((\models_{LIA} \overline{4} \lor \overline{5} \lor 11 \lor 12))</td>
</tr>
<tr>
<td>0 \ldots 9 10 \bullet 11</td>
<td>F, (\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no</td>
<td>by Decide</td>
</tr>
<tr>
<td>0 \ldots 9 10 \bullet 11 13</td>
<td>F, (\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no</td>
<td>by T-Propagate ((0, 1, 11 \models_{UF} 13))</td>
</tr>
<tr>
<td>0 \ldots 9 10 \bullet 11 13</td>
<td>F, (\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>(\overline{7} \lor \overline{13})</td>
<td>by T-Conflict ((7, 13 \models_{UF} \bot))</td>
</tr>
<tr>
<td>0 \ldots 9 10 \bullet 13</td>
<td>F, (\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no</td>
<td>by Backjump</td>
</tr>
<tr>
<td>0 \ldots 9 10 \bullet 13 11</td>
<td>F, (\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no</td>
<td>by T-Propagate ((0, 1, \overline{13} \models_{UF} \overline{11}))</td>
</tr>
<tr>
<td>0 \ldots 9 10 \overline{13} \overline{11} 12</td>
<td>F, (\overline{4} \lor \overline{5} \lor 11 \lor 12)</td>
<td>no</td>
<td>by Propagate (exercise)</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>by Fail</td>
</tr>
</tbody>
</table>
REFERENCES


References


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.*

Robert Brummayer and Armin Biere. 
*Lemmas on demand for the extensional theory of arrays.*

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.*


Miquel Bofill, Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio.  
The Barcologic SMT solver.  

Clark Barrett, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli.  
Splitting on demand in sat modulo theories.  

Kshitij Bansal, Andrew Reynolds, Clark Barrett, and Cesare Tinelli.  
A new decision procedure for finite sets and cardinality constraints in SMT.  

Clark Barrett, Igor Shikanian, and Cesare Tinelli.  
An abstract decision procedure for satisfiability in the theory of recursive data types.  


Bruno Dutertre and Leonardo De Moura.
A fast linear-arithmetic solver for DPLL(T).

Martin Davis, George Logemann, and Donald Loveland.
A machine program for theorem proving.

L. de Moura and N. Bjørner.
Generalized, efficient array decision procedures.

Leonardo De Moura and Nikolaj Bjørner.
Generalized, efficient array decision procedures.

L. de Moura and H. Rueß.
Lemmas on Demand for Satisfiability Solvers.


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


S. K. Lahiri, R. Nieuwenhuis, and A. Oliveras.
SMT Techniques for Fast Predicate Abstraction.

Tianyi Liang, Andrew Reynolds, Cesare Tinelli, Clark Barrett, and Morgan Deters.
A DPLL(T) theory solver for a theory of strings and regular expressions.

Baoluo Meng, Andrew Reynolds, Cesare Tinelli, and Clark Barrett.
Relational constraint solving in SMT.

Greg Nelson and Derek C. Oppen.
Simplification by cooperating decision procedures.

Greg Nelson and Derek C. Oppen.
Fast decision procedures based on congruence closure.
Robert Nieuwenhuis and Albert Oliveras.
DPLL(T) with Exhaustive Theory Propagation and its Application to Difference Logic.

R. Nieuwenhuis and A. Oliveras.
Fast Congruence Closure and Extensions.

Robert Nieuwenhuis and Albert Oliveras.
Fast congruence closure and extensions.

Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli.
Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T).

Derek C. Oppen.
Complexity, convexity and combinations of theories.
Christos H Papadimitriou.  
On the complexity of integer programming.  

A. Pnueli, Y. Rodeh, O. Shtrichman, and M. Siegel.  
Deciding Equality Formulas by Small Domains Instantiations.  

Andrew Reynolds and Jasmin Christian Blanchette.  
A decision procedure for (co)datatypes in SMT solvers.  

Christophe Ringeissen.  
Cooperation of decision procedures for the satisfiability problem.  
Silvio Ranise, Christophe Ringeissen, and Calogero G. Zarba. Combining data structures with nonstably infinite theories using many-sorted logic.


