An Introduction to Satisfiability Modulo Theories

Clark Barrett and Sanjit Seshia

Roadmap

Theory Solvers

- Examples of Theory Solvers
- Combining Theory Solvers
- Extending Theory Solvers for SMT

Theory Solvers

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

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

We next consider some examples of theory solvers.

Congruence Closure and QF_UF

Recall that QF_UF is the theory with only *equality* and *uninterpreted function* symbols.

If Γ is a set of *equalities* and Δ is a set of *disequalities*, then the satisfiability of $\Gamma \cup \Delta$ in QF_UF can be determined as follows [NO80, DST80]:

- Let τ be the set of terms appearing in $\Gamma \cup \Delta$.
- Let $\sim$ be the equiavlence relation on τ induced by Γ (i.e. $t_1 \sim t_2$ iff $t_1 = t_2 \in \Gamma$ or $t_2 = t_1 \in \Gamma$).
- Let $\sim^*$ be the *congruence closure* of $\sim$, obtained by closing $\sim$ with respect to the congruence property:

$$\overline{s} = \overline{t} \rightarrow f(\overline{s}) = f(\overline{t}).$$

- $\Gamma \cup \Delta$ is satisfiable iff for each $s \neq t \in \Delta$, $s \not\sim^* t$.

A Solver for QF_UF

union and *find* are abstract operations for manipulating equivalence classes.

union(x, y) makes y the new equivalence class representative for x.

find(x) returns the unique representative for the equivalence class containing x.

The *signature* of a term is defined as:
sig$(f(x_1, \ldots, x_n)) = f($*find*$(x_1), \ldots,$ *find*$(x_n))$.

A Solver for QF_UF

```
CC(Γ, Δ)
  while Γ ≠ ∅
    Remove some equality a = b from Γ;
    Merge(find(a), find(b));
  if find(a) = find(b) for some a ≠ b ∈ Δ then
    return False;
  return True;
Merge(a, b)
  if a = b then return;
  Let A be the set of terms containing
      a as an argument
  union(a, b);
  foreach x ∈ A
    if sig(x) = sig(y) for some y then
      Merge(find(x), find(y));
```

Example

$$f(f(a)) = a \ \wedge \ f(f(f(a))) = a \ \wedge \ g(a,b) \neq g(f(a),b)$$

t	$\textsf{find}(t)$	$\textsf{sig}(t)$
a	a	a
$f(a)$	$f(a)$	$f(a)$
$f(f(a))$	$f(f(a))$	$f(f(a))$
$f(f(f(a)))$	$f(f(f(a)))$	$f(f(f(a)))$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(f(a),b)$

Example

$$f(f(a)) = a \;\wedge\; f(f(f(a))) = a \;\wedge\; g(a,b) \neq g(f(a),b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	$f(a)$	$f(a)$
$f(f(a))$	$f(f(a))$	$f(f(a))$
$f(f(f(a)))$	$f(f(f(a)))$	$f(f(f(a)))$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(f(a),b)$

Example

$$f(f(a)) = a \;\wedge\; f(f(f(a))) = a \;\wedge\; g(a,b) \neq g(f(a),b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	$f(a)$	$f(a)$
$f(f(a))$	a	$f(f(a))$
$f(f(f(a)))$	$f(f(f(a)))$	$f(f(f(a)))$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(f(a),b)$

Example

$$f(f(a)) = a \;\wedge\; f(f(f(a))) = a \;\wedge\; g(a,b) \neq g(f(a),b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	$f(a)$	$f(a)$
$f(f(a))$	a	$f(f(a))$
$f(f(f(a)))$	$f(f(f(a)))$	$f(f(f(a)))$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(f(a),b)$

Example

$$f(f(a)) = a \;\wedge\; f(f(f(a))) = a \;\wedge\; g(a,b) \neq g(f(a),b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	$f(a)$	$f(a)$
$f(f(a))$	a	$f(f(a))$
$f(f(f(a)))$	$f(f(f(a)))$	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(f(a),b)$

Example

$$f(f(a)) = a \ \wedge \ f(f(f(a))) = a \ \wedge \ g(a, b) \neq g(f(a), b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	$f(a)$	$f(a)$
$f(f(a))$	a	$f(f(a))$
$f(f(f(a)))$	$f(f(f(a)))$	$f(a)$
b	b	b
$g(a, b)$	$g(a, b)$	$g(a, b)$
$g(f(a), b)$	$g(f(a), b)$	$g(f(a), b)$

Example

$$f(f(a)) = a \;\wedge\; f(f(f(a))) = a \;\wedge\; g(a,b) \neq g(f(a),b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	$f(a)$	$f(a)$
$f(f(a))$	a	$f(f(a))$
$f(f(f(a)))$	$f(a)$	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(f(a),b)$

Example

$$f(f(a)) = a \;\wedge\; f(f(f(a))) = a \;\wedge\; g(a,b) \neq g(f(a),b)$$

t	$\textsf{find}(t)$	$\textsf{sig}(t)$
a	a	a
$f(a)$	$f(a)$	$f(a)$
$f(f(a))$	a	$f(f(a))$
$f(f(f(a)))$	$f(a)$	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(f(a),b)$

Example

$$f(f(a)) = a \ \wedge \ f(f(f(a))) = a \ \wedge \ g(a,b) \neq g(f(a),b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	$f(a)$	$f(a)$
$f(f(a))$	a	$f(f(a))$
$f(f(f(a)))$	$f(a)$	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(f(a),b)$

Example

$$f(f(a)) = a \;\wedge\; f(f(f(a))) = a \;\wedge\; g(a,b) \neq g(f(a),b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	a	$f(a)$
$f(f(a))$	a	$f(f(a))$
$f(f(f(a)))$	a	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(f(a),b)$

Example

$$f(f(a)) = a \;\wedge\; f(f(f(a))) = a \;\wedge\; g(a,b) \neq g(f(a),b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	a	$f(a)$
$f(f(a))$	a	$f(f(a))$
$f(f(f(a)))$	a	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(f(a),b)$

Example

$$f(f(a)) = a \ \wedge \ f(f(f(a))) = a \ \wedge \ g(a,b) \neq g(f(a),b)$$

t	*find*(t)	*sig*(t)
a	a	a
$f(a)$	a	$f(a)$
$f(f(a))$	a	$f(a)$
$f(f(f(a)))$	a	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(a,b)$

Example

$$f(f(a)) = a \ \wedge \ f(f(f(a))) = a \ \wedge \ g(a,b) \neq g(f(a),b)$$

t	*find*(t)	*sig*(t)
a	a	a
$f(a)$	a	$f(a)$
$f(f(a))$	a	$f(a)$
$f(f(f(a)))$	a	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(f(a),b)$	$g(a,b)$

Example

$$f(f(a)) = a \;\wedge\; f(f(f(a))) = a \;\wedge\; g(a,b) \neq g(f(a),b)$$

t	*find*(t)	*sig*(t)
a	a	a
$f(a)$	a	$f(a)$
$f(f(a))$	a	$f(a)$
$f(f(f(a)))$	a	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(a,b)$	$g(a,b)$

Example

$$f(f(a)) = a \ \wedge \ f(f(f(a))) = a \ \wedge \ g(a,b) \neq g(f(a),b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	a	$f(a)$
$f(f(a))$	a	$f(a)$
$f(f(f(a)))$	a	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(a,b)$	$g(a,b)$

Example

$$f(f(a)) = a \ \wedge \ f(f(f(a))) = a \ \wedge \ g(a,b) \neq g(f(a),b)$$

t	*find*(t)	*sig*(t)
a	a	a
$f(a)$	a	$f(a)$
$f(f(a))$	a	$f(a)$
$f(f(f(a)))$	a	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(a,b)$	$g(a,b)$

Example

$$f(f(a)) = a \;\wedge\; f(f(f(a))) = a \;\wedge\; g(a,b) \neq g(f(a),b)$$

t	find(t)	sig(t)
a	a	a
$f(a)$	a	$f(a)$
$f(f(a))$	a	$f(a)$
$f(f(f(a)))$	a	$f(a)$
b	b	b
$g(a,b)$	$g(a,b)$	$g(a,b)$
$g(f(a),b)$	$g(a,b)$	$g(a,b)$

find$(g(a,b))$ = find$(g(f(a),b)) \rightarrow$ *Unsatisfiable*

Difference Logic

In *difference logic* [NO05], we are interested in the satisfiability of a conjunction of arithmetic atoms.

Each atom is of the form $x - y \bowtie c$, where x and y are variables, c is a numeric constant, and $\bowtie \in \{=, <, \leq, >, \geq\}$.

The variables can range over either the *integers* (QF_IDL) or the *reals* (QF_RDL).

Difference Logic

The first step is to rewrite everything in terms of $\leq$:

Difference Logic

The first step is to rewrite everything in terms of $\leq$:

- $x - y = c \quad \Longrightarrow \quad x - y \leq c \wedge x - y \geq c$

Difference Logic

The first step is to rewrite everything in terms of $\leq$:

- $x - y = c \quad \Longrightarrow \quad x - y \leq c \wedge x - y \geq c$
- $x - y \geq c \quad \Longrightarrow \quad y - x \leq -c$

Difference Logic

The first step is to rewrite everything in terms of $\leq$:

- $x - y = c \quad \Longrightarrow \quad x - y \leq c \wedge x - y \geq c$
- $x - y \geq c \quad \Longrightarrow \quad y - x \leq -c$
- $x - y > c \quad \Longrightarrow \quad y - x < -c$

Difference Logic

The first step is to rewrite everything in terms of $\leq$:

- $x - y = c \quad \Longrightarrow \quad x - y \leq c \wedge x - y \geq c$
- $x - y \geq c \quad \Longrightarrow \quad y - x \leq -c$
- $x - y > c \quad \Longrightarrow \quad y - x < -c$
- $x - y < c \quad \Longrightarrow \quad x - y \leq c - 1$ (integers)

Difference Logic

The first step is to rewrite everything in terms of $\leq$:

- $x - y = c \quad \Longrightarrow \quad x - y \leq c \,\wedge\, x - y \geq c$
- $x - y \geq c \quad \Longrightarrow \quad y - x \leq -c$
- $x - y > c \quad \Longrightarrow \quad y - x < -c$
- $x - y < c \quad \Longrightarrow \quad x - y \leq c - 1$ (integers)
- $x - y < c \quad \Longrightarrow \quad x - y \leq c - \delta$ (reals)

Difference Logic

Now we have a conjunction of literals, all of the form $x - y \leq c$.

From these literals, we form a weighted directed graph with a vertex for each variable.

For each literal $x - y \leq c$, there is an edge $x \xrightarrow{c} y$.

The set of literals is satisfiable iff there is no cycle for which the sum of the weights on the edges is negative.

There are a number of efficient algorithms for detecting negative cycles in graphs [CG96].

Example: QF_IDL

$$x - y = 5 \ \wedge \ z - y \geq 2 \ \wedge \ z - x > 2 \ \wedge \ w - x = 2 \ \wedge \ z - w < 0$$

Example: QF_IDL

$$x - y = 5 \;\land\; z - y \geq 2 \;\land\; z - x > 2 \;\land\; w - x = 2 \;\land\; z - w < 0$$

$$\begin{array}{lcl} x - y = 5 & & \\ z - y \geq 2 & & \\ z - x > 2 & \Rightarrow & \\ w - x = 2 & & w - x \leq 2 \land x - w \leq -2 \\ z - w < 0 & & \end{array}$$

Example: QF_IDL

$$x - y = 5 \;\wedge\; z - y \geq 2 \;\wedge\; z - x > 2 \;\wedge\; w - x = 2 \;\wedge\; z - w < 0$$

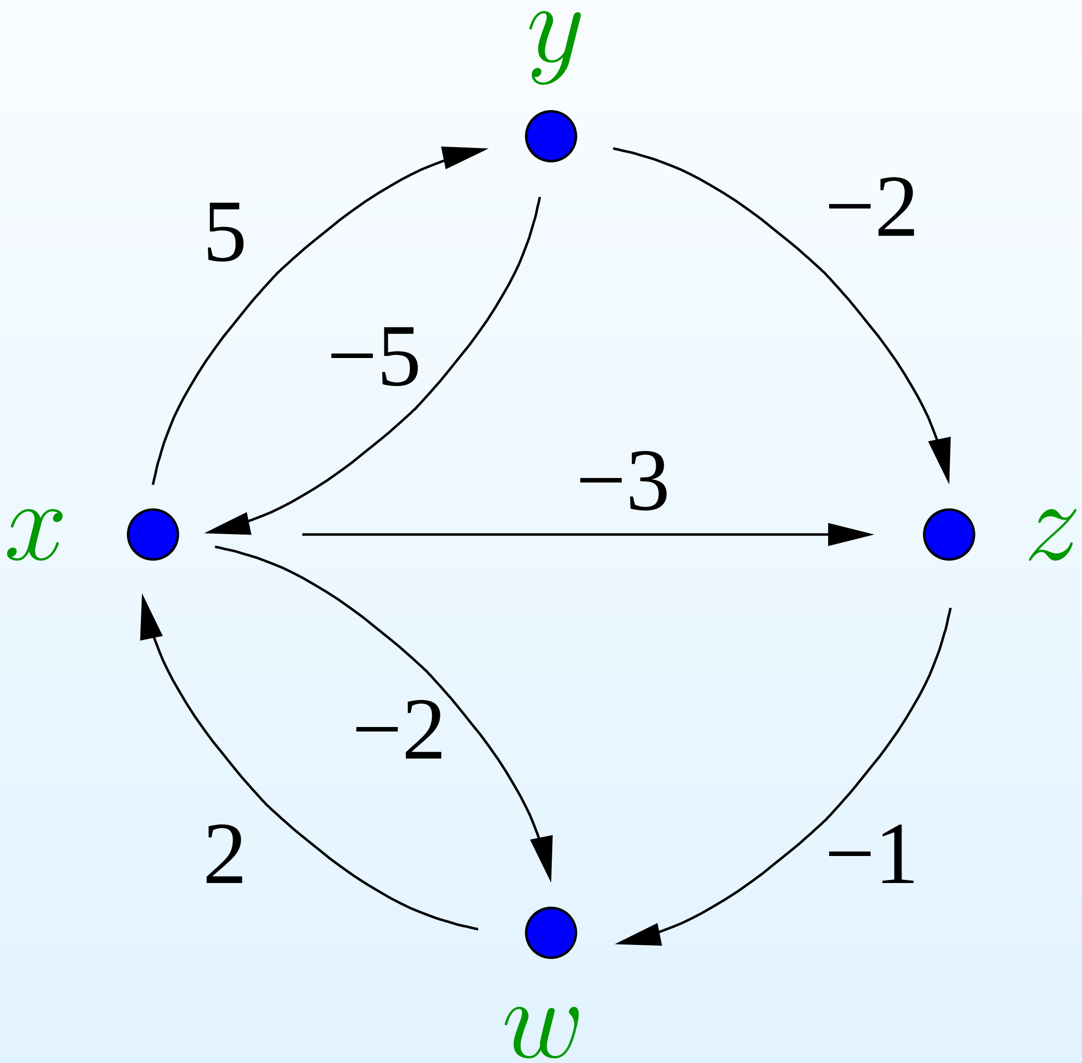
$$\begin{array}{lcl}
x - y = 5 & & \\
z - y \geq 2 & & \\
z - x > 2 & \Rightarrow & \\
w - x = 2 & & w - x \leq 2 \wedge x - w \leq -2 \\
z - w < 0 & &
\end{array}$$

Example: QF_IDL

$$x - y = 5 \ \wedge \ z - y \geq 2 \ \wedge \ z - x > 2 \ \wedge \ w - x = 2 \ \wedge \ z - w < 0$$

$$\begin{array}{lcl}
x - y = 5 & & x - y \leq 5 \wedge y - x \leq -5 \\
z - y \geq 2 & & y - z \leq -2 \\
z - x > 2 & \Rightarrow & x - z \leq -3 \\
w - x = 2 & & w - x \leq 2 \wedge x - w \leq -2 \\
z - w < 0 & & z - w \leq -1
\end{array}$$

Example: QF_IDL



Roadmap

Theory Solvers

- Examples of Theory Solvers
- Combining Theory Solvers
- Extending Theory Solvers for SMT

Combining Theory Solvers

Theory solvers become much more useful if they can be used together.

$$mux\_sel = 0 \rightarrow mux\_out = select(regfile, addr)$$
$$mux\_sel = 1 \rightarrow mux\_out = ALU(alu0, alu1)$$

For such formulas, we are interested in satisfiability with respect to a *combination* of theories.

Fortunately, there exist methods for combining theory solvers. The standard technique for this is the Nelson-Oppen method [NO79, TH96].

The Nelson-Oppen Method

The Nelson-Oppen method is applicable when:

1. The theories have *no shared symbols* (other than equality).
2. The theories are *stably-infinite*.

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

3. The formulas to be tested for satisfiability are *quantifier-free*

Many theories fit these criteria, and extensions exist in some cases when they do not.

The Nelson-Oppen Method

Suppose that T_1 and T_2 are theories and that Sat_1 is a theory solver for T_1-satisfiability and Sat_2 for T_1-satisfiability.

We wish to determine if ϕ is $T_1 \cup T_2$-satisfiable.

1. Convert ϕ to its *separate form* $\phi_1 \wedge \phi_2$.
2. Let S be the set of variables shared between ϕ_1 and ϕ_2.
3. For each *arrangement* Δ of S:
   (a) Run Sat_1 on $\phi_1 \cup \Delta$.
   (b) Run Sat_2 on $\phi_2 \cup \Delta$.

The Nelson-Oppen Method

If there exists an arrangement such that both Sat_1 and Sat_2 succeed, then ϕ is $T_1 \cup T_2$-satisfiable.

If no such arrangement exists, then ϕ is $T_1 \cup T_2$-unsatisfiable.

Example

Consider the following QF_UFLIA formula:

$\phi = 1 \leq x \;\wedge\; x \leq 2 \;\wedge\; f(x) \neq f(1) \;\wedge\; f(x) \neq f(2)$.

Example

Consider the following QF_UFLIA formula:

$$\phi = 1 \leq x \ \wedge \ x \leq 2 \ \wedge \ f(x) \neq f(1) \ \wedge \ f(x) \neq f(2).$$

We first convert ϕ to a separate form:

$$\phi_{UF} = f(x) \neq f(y) \ \wedge \ f(x) \neq f(z)$$
$$\phi_{LIA} = 1 \leq x \ \wedge \ x \leq 2 \ \wedge \ y = 1 \ \wedge \ z = 2$$

The shared variables are $\{x, y, z\}$. There are 5 possible arrangements based on equivalence classes of x, y, and z.

Example

$$\phi_{UF} = f(x) \neq f(y) \ \wedge \ f(x) \neq f(z)$$
$$\phi_{LIA} = 1 \leq x \ \wedge \ x \leq 2 \ \wedge \ y = 1 \ \wedge \ z = 2$$

1. $\{x = y, x = z, y = z\}$
2. $\{x = y, x \neq z, y \neq z\}$
3. $\{x \neq y, x = z, y \neq z\}$
4. $\{x \neq y, x \neq z, y = z\}$
5. $\{x \neq y, x \neq z, y \neq z\}$

Example

$$\phi_{UF} = f(x) \neq f(y) \;\wedge\; f(x) \neq f(z)$$
$$\phi_{LIA} = 1 \leq x \;\wedge\; x \leq 2 \;\wedge\; y = 1 \;\wedge\; z = 2$$

1. $\{x = y, x = z, y = z\}$: inconsistent with ϕ_{UF}.
2. $\{x = y, x \neq z, y \neq z\}$
3. $\{x \neq y, x = z, y \neq z\}$
4. $\{x \neq y, x \neq z, y = z\}$
5. $\{x \neq y, x \neq z, y \neq z\}$

Example

$$\phi_{UF} = f(x) \neq f(y) \ \wedge \ f(x) \neq f(z)$$
$$\phi_{LIA} = 1 \leq x \ \wedge \ x \leq 2 \ \wedge \ y = 1 \ \wedge \ z = 2$$

1. $\{x = y, x = z, y = z\}$: inconsistent with ϕ_{UF}.
2. $\{x = y, x \neq z, y \neq z\}$: inconsistent with ϕ_{UF}.
3. $\{x \neq y, x = z, y \neq z\}$
4. $\{x \neq y, x \neq z, y = z\}$
5. $\{x \neq y, x \neq z, y \neq z\}$

Example

$$\phi_{UF} = f(x) \neq f(y) \;\wedge\; f(x) \neq f(z)$$
$$\phi_{LIA} = 1 \leq x \;\wedge\; x \leq 2 \;\wedge\; y = 1 \;\wedge\; z = 2$$

1. $\{x = y, x = z, y = z\}$: inconsistent with ϕ_{UF}.
2. $\{x = y, x \neq z, y \neq z\}$: inconsistent with ϕ_{UF}.
3. $\{x \neq y, x = z, y \neq z\}$: inconsistent with ϕ_{UF}.
4. $\{x \neq y, x \neq z, y = z\}$
5. $\{x \neq y, x \neq z, y \neq z\}$

Example

$$\phi_{UF} = f(x) \neq f(y) \;\wedge\; f(x) \neq f(z)$$
$$\phi_{LIA} = 1 \leq x \;\wedge\; x \leq 2 \;\wedge\; y = 1 \;\wedge\; z = 2$$

1. $\{x = y, x = z, y = z\}$: inconsistent with ϕ_{UF}.
2. $\{x = y, x \neq z, y \neq z\}$: inconsistent with ϕ_{UF}.
3. $\{x \neq y, x = z, y \neq z\}$: inconsistent with ϕ_{UF}.
4. $\{x \neq y, x \neq z, y = z\}$: inconsistent with ϕ_{LIA}.
5. $\{x \neq y, x \neq z, y \neq z\}$

Example

$$\phi_{UF} = f(x) \neq f(y) \ \wedge \ f(x) \neq f(z)$$
$$\phi_{LIA} = 1 \leq x \ \wedge \ x \leq 2 \ \wedge \ y = 1 \ \wedge \ z = 2$$

1. $\{x = y, x = z, y = z\}$: inconsistent with ϕ_{UF}.
2. $\{x = y, x \neq z, y \neq z\}$: inconsistent with ϕ_{UF}.
3. $\{x \neq y, x = z, y \neq z\}$: inconsistent with ϕ_{UF}.
4. $\{x \neq y, x \neq z, y = z\}$: inconsistent with ϕ_{LIA}.
5. $\{x \neq y, x \neq z, y \neq z\}$: inconsistent with ϕ_{LIA}.

Example

$$\phi_{UF} = f(x) \neq f(y) \ \wedge \ f(x) \neq f(z)$$
$$\phi_{LIA} = 1 \leq x \ \wedge \ x \leq 2 \ \wedge \ y = 1 \ \wedge \ z = 2$$

1. $\{x = y, x = z, y = z\}$: inconsistent with ϕ_{UF}.
2. $\{x = y, x \neq z, y \neq z\}$: inconsistent with ϕ_{UF}.
3. $\{x \neq y, x = z, y \neq z\}$: inconsistent with ϕ_{UF}.
4. $\{x \neq y, x \neq z, y = z\}$: inconsistent with ϕ_{LIA}.
5. $\{x \neq y, x \neq z, y \neq z\}$: inconsistent with ϕ_{LIA}.

Therefore, ϕ is *unsatisfiable*.

Roadmap

Theory Solvers

- Examples of Theory Solvers
- Combining Theory Solvers
- Extending Theory Solvers for SMT

Desirable Characteristics of Theory Solvers

Theory solvers must be able to determine whether a conjunction of literals is satisfiable.

However, in order to integrate a theory solver into a modern SMT solver, it is helpful if the theory solvers can do more.

Desirable Characteristics of Theory Solvers

Some desirable characterstics of theory solvers include:

- *Incrementality* - easy to add new literals or backtrack to a previous state
- *Layered/Lazy* - able to detect simple inconsistencies quickly, able to detect difficult inconsistencies eventually
- *Equality Propagating* - If theory solvers can detect when two terms are equivalent, this greatly simplifies the search for a satisfying arrangement

Desirable Characteristics of Theory Solvers

Some desirable characterstics of theory solvers include:

- *Model Generating* - When reporting satisfiable, the theory solver also provides a concrete value for each variable or function symbol
- *Proof Generating* - When reporting unsatisfiable, the theory solver also provides a checkable proof
- *Interpolant Generating* - If $\phi \wedge \neg\psi$ is unsatisfiable, find a formula α containing only symbols appearing in both ϕ and ψ such that:
  - $\phi \wedge \neg\alpha$ is unsatisfiable
  - $\alpha \wedge \neg\psi$ is unsatisfiable

Lazy SMT

Theory solvers check the satisfiability of conjunctions of literals.

What about more general Boolean structure?

What is needed is a combination of *Boolean reasoning* and *theory reasoning*.

The *eager* approach to SMT does this by encoding theory reasoning as a Boolean satisfiability problem.

Here, I will focus on the *lazy* approach in which both a Boolean engine and a theory solver work together to solve the problem [dMRS02, BDS02a].

Roadmap

From SAT to SMT

- Abstract DPLL
- Abstract DPLL Modulo Theories
- Key Optimizations
- Quantifier Instantiation

Abstract DPLL

We start with an abstract description of DPLL, the algorithm used by most SAT solvers [NOT06].

Abstract DPLL

We start with an abstract description of DPLL, the algorithm used by most SAT solvers [NOT06].

- Abstract DPLL uses *states* and *transitions* to model the progress of the algorithm.

Abstract DPLL

We start with an abstract description of DPLL, the algorithm used by most SAT solvers [NOT06].

- Abstract DPLL uses *states* and *transitions* to model the progress of the algorithm.
- Most states are of the form $M \parallel F$, where
  - M is a *sequence of* annotated *literals* denoting a partial truth assignment, and
  - F is the CNF formula being checked, represented as a *set of clauses*.

Abstract DPLL

We start with an abstract description of DPLL, the algorithm used by most SAT solvers [NOT06].

- Abstract DPLL uses *states* and *transitions* to model the progress of the algorithm.
- Most states are of the form $M \parallel F$, where
  - M is a *sequence of* annotated *literals* denoting a partial truth assignment, and
  - F is the CNF formula being checked, represented as a *set of clauses*.
- The *initial state* is $\emptyset \parallel F$, where F is to be checked for satisfiability.

Abstract DPLL

We start with an abstract description of DPLL, the algorithm used by most SAT solvers [NOT06].

- Abstract DPLL uses *states* and *transitions* to model the progress of the algorithm.
- Most states are of the form $M \parallel F$, where
    - M is a *sequence of* annotated *literals* denoting a partial truth assignment, and
    - F is the CNF formula being checked, represented as a *set of clauses*.
- The *initial state* is $\emptyset \parallel F$, where F is to be checked for satisfiability.
- Transitions between states are defined by a set of *conditional transition rules*.

Abstract DPLL

The *final state* is either:

- a special fail state: $fail$, if F is unsatisfiable, or
- $M \parallel G$, where G is a CNF formula equisatisfiable with the original formula F, with $M \models G$

We write $M \models C$ to mean that C is satisfied whenever M is satisfied. Or in other words, C is a *propositional consequence* of the conjunction of the literals in M.

Abstract DPLL Rules

UnitProp :

$$M \parallel F, C \vee l \quad \Longrightarrow \quad M\ l \parallel F, C \vee l \quad \textbf{if} \begin{cases} M \models \neg C \\ l \text{ is undefined in } M \end{cases}$$

PureLiteral :

$$M \parallel F \quad \Longrightarrow \quad M\ l \parallel F \quad \textbf{if} \begin{cases} l \text{ occurs in some clause of } F \\ -l \text{ occurs in no clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Decide :

$$M \parallel F \quad \Longrightarrow \quad M\ l^{\mathsf{d}} \parallel F \quad \textbf{if} \begin{cases} l \text{ or } \neg l \text{ occurs in a clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Backtrack :

$$M\ l^{\mathsf{d}}\ N \parallel F, C \quad \Longrightarrow \quad M\ \neg l \parallel F, C \quad \textbf{if} \begin{cases} M\ l^{\mathsf{d}}\ N \models \neg C \\ \mathrm{N \text{ contains no decision literals}} \end{cases}$$

Fail :

$$M \parallel F, C \quad \Longrightarrow \quad \mathit{fail} \quad \textbf{if} \begin{cases} M \models \neg C \\ M \text{ contains no decision literals} \end{cases}$$

Example

$$\emptyset \;\|\; 1 \vee \overline{2},\; \overline{1} \vee \overline{2},\; 2 \vee 3,\; \overline{3} \vee 2,\; 1 \vee 4 \quad \Longrightarrow \quad (\text{PureLiteral})$$

$$4\; 1^{d}\; \overline{2}\; 3 \;\|$$

Example

$$
\begin{array}{rcll}
\emptyset & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{3}\vee 2,\ \ 1\vee 4 & \Longrightarrow \quad (\text{PureLiteral}) \\
4 & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{3}\vee 2,\ \ 1\vee 4 & \\
4\ 1^{d}\ \overline{2}\ 3 & \| & &
\end{array}
$$

Example

$$
\begin{array}{rcllllll}
\emptyset & \parallel & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow \quad (\textsf{PureLiteral}) \\
4 & \parallel & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow \quad (\textsf{Decide}) \\
4\ 1^{\mathsf{d}} & \parallel & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \\
4\ 1^{\mathsf{d}}\ \overline{2}\ 3 & \parallel & & & & & &
\end{array}
$$

Example

$$
\begin{array}{rcllllllcl}
\emptyset & \parallel & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & & \Longrightarrow & (\textsf{PureLiteral}) \\
4 & \parallel & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & & \Longrightarrow & (\textsf{Decide}) \\
4\ 1^{\mathsf{d}} & \parallel & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & & \Longrightarrow & (\textsf{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2} & \parallel & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & & & \\
4\ 1^{\mathsf{d}}\ \overline{2}\ 3 & \parallel & & & & & & & &
\end{array}
$$

Example

$$
\begin{array}{rcll}
\emptyset & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{3}\vee2,\ \ 1\vee4 & \Longrightarrow \ (\textsf{PureLiteral}) \\
4 & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{3}\vee2,\ \ 1\vee4 & \Longrightarrow \ (\textsf{Decide}) \\
4\ 1^{\mathsf{d}} & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{3}\vee2,\ \ 1\vee4 & \Longrightarrow \ (\textsf{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2} & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{3}\vee2,\ \ 1\vee4 & \Longrightarrow \ (\textsf{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2}\ 3 & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{3}\vee2,\ \ 1\vee4 &
\end{array}
$$

Example

$$
\begin{array}{rcllllllll}
\emptyset & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{PureLiteral}) \\
4 & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{Decide}) \\
4\ 1^{\mathsf{d}} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2}\ 3 & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{Backtrack}) \\
4\ \overline{1} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & &
\end{array}
$$

Example

$$
\begin{array}{rclcl}
\emptyset & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{3}\vee 2,\ \ 1\vee 4 & \Longrightarrow & (\text{PureLiteral}) \\
4 & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{3}\vee 2,\ \ 1\vee 4 & \Longrightarrow & (\text{Decide}) \\
4\ 1^{\mathsf{d}} & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{3}\vee 2,\ \ 1\vee 4 & \Longrightarrow & (\text{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2} & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{3}\vee 2,\ \ 1\vee 4 & \Longrightarrow & (\text{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2}\ 3 & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{3}\vee 2,\ \ 1\vee 4 & \Longrightarrow & (\text{Backtrack}) \\
4\ \overline{1} & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{3}\vee 2,\ \ 1\vee 4 & \Longrightarrow & (\text{UnitProp}) \\
4\ \overline{1}\ \overline{2}\ \overline{3} & \| & 1\vee\overline{2},\ \ \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{3}\vee 2,\ \ 1\vee 4 & &
\end{array}
$$

Example

$$\begin{array}{rcllllllll}
\emptyset & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\textsf{PureLiteral}) \\
4 & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\textsf{Decide}) \\
4\ 1^{\mathsf{d}} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\textsf{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\textsf{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2}\ 3 & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\textsf{Backtrack}) \\
4\ \overline{1} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\textsf{UnitProp}) \\
4\ \overline{1}\ \overline{2}\ \overline{3} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\textsf{Fail}) \\
 & fail & & & & & & &
\end{array}$$

Example

$$
\begin{array}{rcllllllr}
\emptyset & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{PureLiteral}) \\
4 & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{Decide}) \\
4\ 1^{\mathsf{d}} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2}\ 3 & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{Backtrack}) \\
4\ \overline{1} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{UnitProp}) \\
4\ \overline{1}\ \overline{2}\ \overline{3} & \| & 1\vee\overline{2}, & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{3}\vee 2, & 1\vee 4 & \Longrightarrow & (\text{Fail}) \\
 & \mathit{fail} & & & & & & &
\end{array}
$$

Result: *Unsatisfiable*

Additional Abstract DPLL Rules

Backjump :

$$M\ l^{\mathsf{d}}\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C \quad \textbf{if} \begin{cases} M\ l^{\mathsf{d}}\ N \models \neg C, \text{ and there is} \\ \text{some clause } C' \vee l' \text{ such that:} \\ \quad F, C \models C' \vee l' \text{ and } M \models \neg C', \\ \quad l' \text{ is undefined in } M, \text{ and} \\ \quad l' \text{ or } \neg l' \text{ occurs in } F \text{ or in } M\ l^{\mathsf{d}}\ N \end{cases}$$

Learn :

$$M \parallel F \quad \Longrightarrow \quad M \parallel F, C \quad \textbf{if} \begin{cases} \text{all atoms of } C \text{ occur in } F \\ F \models C \end{cases}$$

Forget :

$$M \parallel F, C \quad \Longrightarrow \quad M \parallel F \quad \textbf{if} \ \{ \ F \models C$$

Restart :

$$M \parallel F \quad \Longrightarrow \quad \emptyset \parallel F$$

Roadmap

From SAT to SMT

- Abstract DPLL
- Abstract DPLL Modulo Theories
- Key Optimizations
- Quantifier Instantiation

Abstract DPLL Modulo Theories

The *Abstract DPLL Modulo Theories* framework *extends* the Abstract DPLL framework to include theory reasoning [NOT06].

Assume we have a theory T and a solver Sat_T that can check satisfiability of conjunctions of literals in T.

Suppose we want to check the T-satisfiability of an *arbitrary* (quantifier-free) formula ϕ.

We start by converting ϕ to CNF.

We can then use the *Abstract DPLL* rules, allowing *any first-order* literal where before we had propositional literals.

Abstract DPLL Modulo Theories

The *Abstract DPLL Modulo Theories* framework *extends* the Abstract DPLL framework to include theory reasoning [NOT06].

Assume we have a theory T and a solver Sat_T that can check satisfiability of conjunctions of literals in T.

Suppose we want to check the T-satisfiability of an *arbitrary* (quantifier-free) formula ϕ.

We start by converting ϕ to CNF.

What other changes do we need to make to Abstract DPLL so it will work for SMT?

Abstract DPLL Modulo Theories

The first change is to the definition of a *final state*. A final state is now:

- the special fail state: $fail$, or
- $M \parallel F$, where $M \models F$, *and* $\mathsf{Sat}_T(M)$ reports satisfiable.

Abstract DPLL Modulo Theories

The first change is to the definition of a *final state*. A final state is now:

- the special fail state: $fail$, or
- $M \parallel F$, where $M \models F$, *and* $\textsf{Sat}_T(M)$ reports satisfiable.

What happens if we reach a state in which: $M \parallel F$, $M \models F$, and $\textsf{Sat}_T(M)$ reports *unsatisfiable*? (call this a *pseudo-final* state)

Abstract DPLL Modulo Theories

The first change is to the definition of a *final state*. A final state is now:

- the special fail state: $fail$, or
- $M \parallel F$, where $M \models F$, *and* $\mathsf{Sat}_T(M)$ reports satisfiable.

What happens if we reach a state in which: $M \parallel F$, $M \models F$, and $\mathsf{Sat}_T(M)$ reports *unsatisfiable*? (call this a *pseudo-final* state)

We need to backtrack. The DPLL rules will take care of this automatically if we add a clause C such that $M \models \neg C$.

Abstract DPLL Modulo Theories

The first change is to the definition of a *final state*. A final state is now:

- the special fail state: $fail$, or
- $M \parallel F$, where $M \models F$, *and* $\mathsf{Sat}_T(M)$ reports satisfiable.

What happens if we reach a state in which: $M \parallel F$, $M \models F$, and $\mathsf{Sat}_T(M)$ reports *unsatisfiable*? (call this a *pseudo-final* state)

We need to backtrack. The DPLL rules will take care of this automatically if we add a clause C such that $M \models \neg C$.

What clause should we add?

Abstract DPLL Modulo Theories

The first change is to the definition of a *final state*. A final state is now:

- the special fail state: $fail$, or
- $M \parallel F$, where $M \models F$, *and* $\mathsf{Sat}_T(M)$ reports satisfiable.

What happens if we reach a state in which: $M \parallel F$, $M \models F$, and $\mathsf{Sat}_T(M)$ reports *unsatisfiable*? (call this a *pseudo-final* state)

We need to backtrack. The DPLL rules will take care of this automatically if we add a clause C such that $M \models \neg C$.

What clause should we add?   How about $\neg M$?

Abstract DPLL Modulo Theories

The justification for adding $\neg M$ is that $\models_T \neg M$.

Note that $\Gamma \models_T \phi$ denotes that ϕ holds whenever both Γ and T are satisfied.

We can generalize this to allow any clause C to be added as long as $F \models_T C$. The following modified Learn rule allows this (we also modify the Forget rule in an analagous way):

Theory Learn :

$$M \parallel F \quad \Longrightarrow \quad M \parallel F, C \quad \textbf{if} \begin{cases} \text{all atoms of } C \text{ occur in } F \\ F \models_T C \end{cases}$$

Theory Forget :

$$M \parallel F, C \quad \Longrightarrow \quad M \parallel F \quad \textbf{if} \left\{ F \models_T C \right.$$

Abstract DPLL Modulo Theories

The resulting set of rules is almost enough to correctly implement an SMT solver. We need one more change.

Abstract DPLL Modulo Theories

The resulting set of rules is almost enough to correctly implement an SMT solver. We need one more change.

A somewhat surprising observation is that the *pure literal* rule has to be abandoned. *Why?*

Abstract DPLL Modulo Theories

The resulting set of rules is almost enough to correctly implement an SMT solver. We need one more change.

A somewhat surprising observation is that the *pure literal* rule has to be abandoned. *Why?*

Propositional literals are independent of each other, but first order literals may not be.

Abstract DPLL Modulo Theories

The resulting set of rules is almost enough to correctly implement an SMT solver. We need one more change.

A somewhat surprising observation is that the *pure literal* rule has to be abandoned. *Why?*

Propositional literals are independent of each other, but first order literals may not be.

The remaining rules form a sound and complete procedure for SMT.

Example of Lazy SMT

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

$$\emptyset \;\|\; \quad 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$$

Example of Lazy SMT

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

$$\emptyset \parallel \quad 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} \qquad \Longrightarrow \qquad (\text{UnitProp})$$

$$1 \parallel \quad 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$$

Example of Lazy SMT

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

$$\begin{array}{rcl l l}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow & (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow & (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & &
\end{array}$$

Example of Lazy SMT

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} &
\end{array}$$

Example of Lazy SMT

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3} & \Longrightarrow \quad (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3} & \Longrightarrow \quad (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3} & \Longrightarrow \quad (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3} & \Longrightarrow \quad (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2\vee 4 &
\end{array}$$

Example of Lazy SMT

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

$\emptyset$	$\|$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$	$\Longrightarrow$	(UnitProp)
1	$\|$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$	$\Longrightarrow$	(Decide)
$1\ \overline{2}^{\mathsf{d}}$	$\|$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$	$\Longrightarrow$	(Decide)
$1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}}$	$\|$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$	$\Longrightarrow$	(Theory Learn)
$1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}}$	$\|$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4$	$\Longrightarrow$	(Backjump)
$1\ \overline{2}^{\mathsf{d}}\ 4$	$\|$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4$		

Example of Lazy SMT

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow \ (\text{Backjump}) \\
1\ \overline{2}^{\mathsf{d}}\ 4 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow \ (\text{UnitProp}) \\
1\ \overline{2}^{\mathsf{d}}\ 4\ \overline{3} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 &
\end{array}$$

Example of Lazy SMT

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow \ (\text{Backjump}) \\
1\ \overline{2}^{\mathsf{d}}\ 4 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow \ (\text{UnitProp}) \\
1\ \overline{2}^{\mathsf{d}}\ 4\ \overline{3} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow \ (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ 4\ \overline{3} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee 2 \vee \overline{4} \vee 3 &
\end{array}$$

Example of Lazy SMT

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

$$\begin{array}{rcl}
\emptyset \;\|\; 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow & (\text{UnitProp}) \\
1 \;\|\; 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow & (\text{Decide}) \\
1\ \overline{2}^{\text{d}} \;\|\; 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow & (\text{Decide}) \\
1\ \overline{2}^{\text{d}}\ \overline{4}^{\text{d}} \;\|\; 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow & (\text{Theory Learn}) \\
1\ \overline{2}^{\text{d}}\ \overline{4}^{\text{d}} \;\|\; 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow & (\text{Backjump}) \\
1\ \overline{2}^{\text{d}}\ 4 \;\|\; 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow & (\text{UnitProp}) \\
1\ \overline{2}^{\text{d}}\ 4\ \overline{3} \;\|\; 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow & (\text{Theory Learn}) \\
1\ \overline{2}^{\text{d}}\ 4\ \overline{3} \;\|\; 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee 2 \vee \overline{4} \vee 3 & \Longrightarrow & (\text{Backjump}) \\
1\ 2 \;\|\; 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee 2 \vee \overline{4} \vee 3 & &
\end{array}$$

Example of Lazy SMT

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

$$
\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow \ (\text{Backjump}) \\
1\ \overline{2}^{\mathsf{d}}\ 4 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow \ (\text{UnitProp}) \\
1\ \overline{2}^{\mathsf{d}}\ 4\ \overline{3} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4 & \Longrightarrow \ (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ 4\ \overline{3} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee 2 \vee \overline{4} \vee 3 & \Longrightarrow \ (\text{Backjump}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee 2 \vee \overline{4} \vee 3 & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee 2 \vee \overline{4} \vee 3 &
\end{array}
$$

Example of Lazy SMT

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

$\emptyset$	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3}$	$\Longrightarrow$	(UnitProp)
1	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3}$	$\Longrightarrow$	(Decide)
$1\ \overline{2}^{\mathsf{d}}$	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3}$	$\Longrightarrow$	(Decide)
$1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}}$	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3}$	$\Longrightarrow$	(Theory Learn)
$1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}}$	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2\vee 4$	$\Longrightarrow$	(Backjump)
$1\ \overline{2}^{\mathsf{d}}\ 4$	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2\vee 4$	$\Longrightarrow$	(UnitProp)
$1\ \overline{2}^{\mathsf{d}}\ 4\ \overline{3}$	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2\vee 4$	$\Longrightarrow$	(Theory Learn)
$1\ \overline{2}^{\mathsf{d}}\ 4\ \overline{3}$	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2\vee 4,\ \overline{1}\vee 2\vee\overline{4}\vee 3$	$\Longrightarrow$	(Backjump)
$1\ 2$	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2\vee 4,\ \overline{1}\vee 2\vee\overline{4}\vee 3$	$\Longrightarrow$	(UnitProp)
$1\ 2\ 3\ \overline{4}$	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2\vee 4,\ \overline{1}\vee 2\vee\overline{4}\vee 3$	$\Longrightarrow$	(Theory Learn)
$1\ 2\ 3\ \overline{4}$	$\parallel$	$1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2\vee 4,\ \overline{1}\vee 2\vee\overline{4}\vee 3,\ \overline{1}\vee\overline{2}\vee\overline{3}\vee 4$		

Example of Lazy SMT

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

$\emptyset$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$	$\Longrightarrow$	(UnitProp)
1	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$	$\Longrightarrow$	(Decide)
$1\ \overline{2}^{\mathsf{d}}$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$	$\Longrightarrow$	(Decide)
$1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}}$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$	$\Longrightarrow$	(Theory Learn)
$1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}}$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4$	$\Longrightarrow$	(Backjump)
$1\ \overline{2}^{\mathsf{d}}\ 4$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4$	$\Longrightarrow$	(UnitProp)
$1\ \overline{2}^{\mathsf{d}}\ 4\ \overline{3}$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4$	$\Longrightarrow$	(Theory Learn)
$1\ \overline{2}^{\mathsf{d}}\ 4\ \overline{3}$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee 2 \vee \overline{4} \vee 3$	$\Longrightarrow$	(Backjump)
$1\ 2$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee 2 \vee \overline{4} \vee 3$	$\Longrightarrow$	(UnitProp)
$1\ 2\ 3\ \overline{4}$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee 2 \vee \overline{4} \vee 3$	$\Longrightarrow$	(Theory Learn)
$1\ 2\ 3\ \overline{4}$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 \vee 4,\ \overline{1} \vee 2 \vee \overline{4} \vee 3,\ \overline{1} \vee \overline{2} \vee \overline{3} \vee 4$	$\Longrightarrow$	(Fail)
fail				

Roadmap

From SAT to SMT

- Abstract DPLL
- Abstract DPLL Modulo Theories
- Key Optimizations
- Quantifier Instantiation

Key Optimizations

We will mention three ways to improve the algorithm.

- Minimizing learned clauses
- Early conflict detection
- Theory propagation

Minimizing Learned Clauses

The main problem with the approach as described so far is that learning $\neg M$ in every pseudo-final state is very inefficient.

To see why, recall that a pseudo-final state is:

- $M \parallel F$, where
- $M \models F$, and
- $\mathit{Sat}_T(M) = \mathit{False}$

Note that M is a sequence of literals and could be quite large.

However, it is often the case that a small subset of M is sufficient to cause an inconsistency in T.

Minimizing Learned Clauses

To solve the problem, whenever $\mathit{Sat}_T(M)$ is called, an effort must be made to find the *smallest* possible subset of M which is inconsistent.

There are several methods:

- Brute-force minimization (typically too slow)
- Traverse a proof tree for each inconsistency (similar to traversing an implication graph in SAT solvers)
- Ad hoc per-theory techniques

Example with Minimized Learned Clauses

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

$$\emptyset \;\parallel\; \quad 1,\; \overline{2} \vee 3,\; \overline{4} \vee \overline{3}$$

Example with Minimized Learned Clauses

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

$$\begin{array}{rcll} \emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{UnitProp}) \\ 1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \end{array}$$

Example with Minimized Learned Clauses

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

$$\begin{array}{rclcl}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow & (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow & (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & &
\end{array}$$

Example with Minimized Learned Clauses

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} &
\end{array}$$

Example with Minimized Learned Clauses

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

$$\begin{array}{rcllcl}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & & &
\end{array}$$

Example with Minimized Learned Clauses

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\textsf{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\textsf{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\textsf{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\textsf{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & \Longrightarrow \quad (\textsf{Backjump}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 &
\end{array}$$

Example with Minimized Learned Clauses

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

$$\begin{array}{rcllcl}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & & \Longrightarrow & (\text{Backjump}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & & \Longrightarrow & (\text{UnitProp}) \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & & &
\end{array}$$

Example with Minimized Learned Clauses

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3} & \Longrightarrow \ (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3} & \Longrightarrow \ (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2 & \Longrightarrow \ (\text{Backjump}) \\
1\ 2 & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2 & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2 & \Longrightarrow \ (\text{Theory Learn}) \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2}\vee 3,\ \overline{4}\vee\overline{3},\ \overline{1}\vee 2,\ \overline{1}\vee\overline{3}\vee 4 &
\end{array}$$

Example with Minimized Learned Clauses

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & \Longrightarrow \ (\text{Backjump}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & \Longrightarrow \ (\text{Theory Learn}) \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 & \Longrightarrow \ (\text{Fail}) \\
 & fail & &
\end{array}$$

Early Conflict Detection

So far, we have indicated that we will check M for T-satisfiability only when a pseudo-final state is reached.

In contrast, we could check M for T-satisfiability every time M changes, possibly resulting in earlier detection of conflicts.

Experimental results show that this approach is significantly better.

It requires *Sat*$_T$ to be *online*: able quickly to determine the consistency of incrementally more literals or to backtrack to a previous state.

It also requires that the SAT solver be instrumented to call *Sat*$_T$ every time a variable is assigned a value.

Example with Early Conflict Detection

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

$$\emptyset \ \| \quad 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$$

Example with Early Conflict Detection

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

$$\begin{array}{rcll} \emptyset & \parallel & 1,\ \overline{2} \lor 3,\ \overline{4} \lor \overline{3} & \Longrightarrow \quad (\text{UnitProp}) \\ 1 & \parallel & 1,\ \overline{2} \lor 3,\ \overline{4} \lor \overline{3} & \end{array}$$

Example with Early Conflict Detection

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

$$\begin{array}{rcll} \emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\textsf{UnitProp}) \\ 1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\textsf{Decide}) \\ 1\ \overline{2}^{\textsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \end{array}$$

Example with Early Conflict Detection

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 &
\end{array}$$

Example with Early Conflict Detection

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

$$\begin{array}{rcllcl}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & & \Longrightarrow & (\text{Backjump}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & & &
\end{array}$$

Example with Early Conflict Detection

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \text{(UnitProp)} \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \text{(Decide)} \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \text{(Theory Learn)} \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & \Longrightarrow \text{(Backjump)} \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & \Longrightarrow \text{(UnitProp)} \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 &
\end{array}$$

Example with Early Conflict Detection

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & \Longrightarrow \ (\text{Backjump}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & \Longrightarrow \ (\text{Theory Learn}) \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 &
\end{array}$$

Example with Early Conflict Detection

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

$$
\begin{array}{rcllcl}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{Decide}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \Longrightarrow & (\text{Theory Learn}) \\
1\ \overline{2}^{\mathsf{d}} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & & \Longrightarrow & (\text{Backjump}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & & \Longrightarrow & (\text{UnitProp}) \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2 & & \Longrightarrow & (\text{Theory Learn}) \\
1\ 2\ 3\ \overline{4} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 & & \Longrightarrow & (\text{Fail}) \\
 & & fail & & &
\end{array}
$$

Theory Propagation

A final improvement is to add the following rule:

Theory Propagate :

$$M \parallel F \quad \Longrightarrow \quad M\, l \parallel F \quad \textbf{if} \begin{cases} M \models_T l \\ l \text{ or } \neg l \text{ occurs in } F \\ l \text{ is undefined in } M \end{cases}$$

This rule allows *Sat*$_T$ to inform the SAT solver if it is able to deduce that an unassigned literal is entailed by the current set of literals (M).

Experimental results show that this can also be very helpful in practice.

Techniques for implementing theory propagation vary by solver and by theory.

Example with Theory Propagation

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

$$\emptyset \;\; \| \quad 1,\; \overline{2} \vee 3,\; \overline{4} \vee \overline{3}$$

Example with Theory Propagation

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

$$\begin{array}{lcll} \emptyset \;\|\; 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow & (\text{UnitProp}) \\ 1 \;\|\; \color{green}{1},\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & & \end{array}$$

Example with Theory Propagation

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

$$\begin{array}{rcll} \emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{UnitProp}) \\ 1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{Theory Propagate}) \\ 1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \end{array}$$

Example with Theory Propagation

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{Theory Propagate}) \\
1\ \overline{2} & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{UnitProp}) \\
1\ \overline{2}\ 3 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} &
\end{array}$$

Example with Theory Propagation

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Theory Propagate}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2\ 3 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \ (\text{Theory Propagate}) \\
1\ 2\ 3\ 4 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} &
\end{array}$$

Example with Theory Propagation

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

$$\begin{array}{rcll}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{Theory Propagate}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{UnitProp}) \\
1\ 2\ 3 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{Theory Propagate}) \\
1\ 2\ 3\ 4 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3} & \Longrightarrow \quad (\text{Fail}) \\
& fail & &
\end{array}$$

Roadmap

From SAT to SMT

- Abstract DPLL
- Abstract DPLL Modulo Theories
- Key Optimizations
- Quantifier Instantiation

Quantifiers

The Abstract DPLL Modulo Theories framework can also be extended to include rules for quantifier instantiation [GBT07].

- First, we extend the notion of literal to that of an *abstract* literal which may include quantified formulas in place of atomic formulas.
- Add two additional rules:

Inst_∃ :

$$M \parallel F \quad \Longrightarrow \quad M \parallel F, (\neg\exists\, x.\, P \vee P[x/sk]) \quad \textbf{if} \begin{cases} \exists\, x\; P \text{ is an abstract literal in } M \\ sk \text{ is a fresh constant.} \end{cases}$$

Inst_∀ :

$$M \parallel F \quad \Longrightarrow \quad M \parallel F, (\neg\forall\, x.\, P \vee P[x/t]) \quad \textbf{if} \begin{cases} \forall\, x\; P \text{ is an abstract literal in } M \\ t \text{ is a ground term.} \end{cases}$$

An Example

Suppose a and b are constant symbols and f is an uninterpreted function symbol. We show how to prove the validity of the following formula:

$$(0 \leq b \wedge (\forall x.\, 0 \leq x \rightarrow f(x) = a)) \rightarrow f(b) = a$$

We first negate the formula and put it into abstract CNF. The result is three unit clauses:

$$(0 \leq b) \wedge (\forall x.\, (\neg 0 \leq x \vee f(x) = a)) \wedge (\neg f(b) = a)$$

An Example

Let l_1, l_2, l_3 denote the three abstract literals in the above clauses. Then the following is a derivation in the extended framework:

$$\begin{array}{rcll}
\emptyset & \| & (l_1)(l_2)(l_3) & \Longrightarrow (\text{UnitProp}) \\
l_1, l_2, l_3 & \| & (l_1)(l_2)(l_3) & \Longrightarrow (\text{Inst_}\forall) \\
l_1, l_2, l_3 & \| & (l_1)(l_2)(l_3)(\neg(0 \leq b) \vee f(b) = a) & \Longrightarrow (\text{Fail}) \\
fail & & &
\end{array}$$

The last transition is possible because M falsifies the last clause in F and contains no decisions (case-splits). As a result, we may conclude that the original set of clauses is unsatisfiable, which implies that the original formula is valid.

Quantifiers

The simple technique of quantifier instantiation is remarkably effective on verification benchmarks.

The main difficulty is coming up with the right terms to instantiate.

Matching techniques pioneered by Simplify [DNS03] have recently been adopted and improved by several modern SMT solvers [FJS04, BdM07, GBT07].

References

[BdM07] Nikolaj Bjørner and Leonardo de Moura. Efficient E-matching for SMT solvers. In Frank Pfenning, editor, *Proceedings of the* 21^{st} *International Conference on Automated Deduction (CADE '07)*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 183–198. Springer-Verlag, July 2007

[BDS02a] Clark W. Barrett, David L. Dill, and Aaron Stump. Checking satisfiability of first-order formulas by incremental translation to SAT. In Ed Brinksma and Kim Guldstrand Larsen, editors, *Proceedings of the* 14^{th} *International Conference on Computer Aided Verification (CAV '02)*, volume 2404 of *Lecture Notes in Computer Science*, pages 236–249. Springer-Verlag, July 2002. Copenhagen, Denmark

[CG96] B. V. Cherkassy and A. V. Goldberg. Negative-cycle detection algorithms. In *European Symposium on Algorithms*, pages 349–363, 1996

[dMRS02] L. de Moura, H. Rueß, and M. Sorea. Lazy Theorem Proving for Bounded Model Checking over Infinite Domains. In *Proc. of the 18th International Conference on Automated Deduction*, volume 2392 of *LNCS*, pages 438–455. Springer, July 2002

[DNS03] David Detlefs, Greg Nelson, and James B. Saxe. Simplify: A theorem prover for program checking. Technical Report HPL-2003-148, HP Laboratories Palo Alto, 2003

References

[DST80] P. J. Downey, R. Sethi, and R. E. Tarjan. Variations on the common subexpression problem. *Journal of the Association for Computing Machinery*, 27(4):758–771, October 1980

[FJS04] Cormac Flanagan, Rajeev Joshi, and James B. Saxe. An explicating theorem prover for quantified formulas. Technical Report HPL-2004-199, HP Laboratories Palo Alto, 2004

[GBT07] Yeting Ge, Clark Barrett, and Cesare Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In Frank Pfenning, editor, *Proceedings of the 21^{st} International Conference on Automated Deduction (CADE '07)*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 167–182. Springer-Verlag, July 2007. Bremen, Germany

[NO79] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. on Programming Languages and Systems*, 1(2):245–257, October 1979

[NO80] Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980

References

[NO05] Robert Nieuwenhuis and Albert Oliveras. DPLL(T) with exhaustive theory propagation and its application to difference logic. In Kousha Etessami and Sriram K. Rajamani, editors, *Proceedings of the* 17^{th} *International Conference on Computer Aided Verification (CAV '05)*, volume 3576 of *Lecture Notes in Computer Science*, pages 321–334. Springer, July 2005

[NOT06] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, November 2006

[TH96] C. Tinelli and M. Harandi. A new correctness proof of the nelson-oppen combination procedure. In F. Baader and K. Schulz, editors, *1st International Workshop on Frontiers of Combining Systems (FroCoS'96)*, volume 3 of *Applied Logic Series*. Kluwer Academic Publishers, 1996