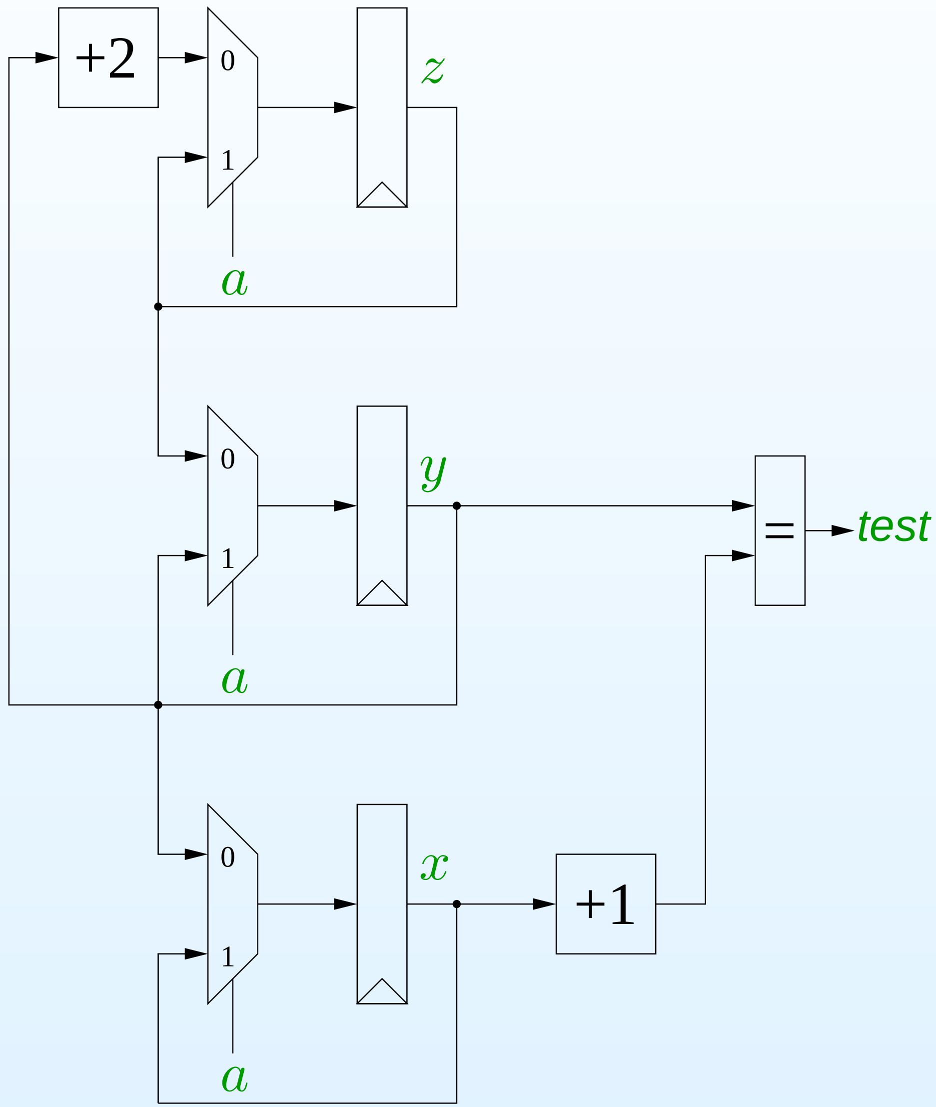
From SAT to SMT: Successes and Challenges

Clark Barrett

barrett@cs.nyu.edu

New York University

Example



Circuit Example

In this example, the value of *test* is always supposed to be *true*.

Circuit Example

In this example, the value of *test* is always supposed to be *true*.

Under what conditions does this hold?

Circuit Example

In this example, the value of *test* is always supposed to be *true* .

Under what conditions does this hold?

How do we prove it?

Circuit Example

In this example, the value of *test* is always supposed to be *true* .

Under what conditions does this hold?

How do we prove it?

One way to prove this is by induction over the number of clock cycles the circuit has executed.

The inductive step is to prove that if *test* is *true* in the current state, then *test* should be *true* in the next state.

Circuit Example

In this example, the value of *test* is always supposed to be *true* .

Under what conditions does this hold?

How do we prove it?

One way to prove this is by induction over the number of clock cycles the circuit has executed.

The inductive step is to prove that if *test* is *true* in the current state, then *test* should be *true* in the next state.

We will look at a couple of possible ways to prove this.

Circuit Example

The logic of the example can be modeled intuitively as follows:

```
(y = x + 1 AND z = x + 2 AND
 x′ = IF a THEN x ELSE y AND
 y′ = IF a THEN y ELSE z AND
 z′ = IF a THEN z ELSE y + 2) IMPLIES
 y′ = x′ + 1 AND z′ = x′ + 2
```

We can prove this formula by showing that the negation is unsatisfiable.

We can write this formula in propositional logic by using one propositional variable for each bit in the current and next states.

Circuit Example

Assuming a bit-width of 2 for simplicity and skipping the details, we get the following formula:

$$
\begin{aligned}
&(z1 \leftrightarrow \neg x1) \wedge (z0 \leftrightarrow x0) \wedge \\
&(y1 \leftrightarrow (x1 \oplus x0)) \wedge (y0 \leftrightarrow \neg x0) \wedge \\
&(a \longrightarrow ((xp1 \leftrightarrow x1) \wedge (xp0 \leftrightarrow x0))) \wedge \\
&(\neg a \longrightarrow ((xp1 \leftrightarrow y1) \wedge (xp0 \leftrightarrow y0))) \wedge \\
&(a \longrightarrow ((yp1 \leftrightarrow y1) \wedge (yp0 \leftrightarrow y0))) \wedge \\
&(\neg a \longrightarrow ((yp1 \leftrightarrow z1) \wedge (yp0 \leftrightarrow z0))) \wedge \\
&(a \longrightarrow ((zp1 \leftrightarrow z1) \wedge (zp0 \leftrightarrow z0))) \wedge \\
&(\neg a \longrightarrow ((zp1 \leftrightarrow \neg y1) \wedge (zp0 \leftrightarrow y0))) \wedge \\
&(\neg(zp1 \leftrightarrow \neg xp1) \vee \neg(zp0 \leftrightarrow xp0) \vee \\
&\neg(yp1 \leftrightarrow (xp1 \oplus xp0)) \wedge (yp0 \leftrightarrow \neg xp0)
\end{aligned}
$$

Circuit Example

Recall that the invariant of the circuit is captured by the following formula:

```
(y = x + 1 AND z = x + 2 AND
 x′ = IF a THEN x ELSE y AND
 y′ = IF a THEN y ELSE z AND
 z′ = IF a THEN z ELSE y + 2) IMPLIES
 y′ = x′ + 1 AND z′ = x′ + 2
```

When using a SAT solver, this formula must be encoded into propositional logic

Using an SMT solver, the formula can be solved as it is

Motivation

Automatic analysis of computer hardware and software requires engines capable of reasoning efficiently about large and complex systems.

Boolean engines such as Binary Decision Diagrams and SAT solvers are typical engines of choice for today's industrial verification applications.

However, systems are usually designed and modeled at a higher level than the Boolean level and the translation to Boolean logic can be expensive.

A primary goal of research in Satisfiability Modulo Theories (SMT) is to create verification engines that can reason natively at a higher level of abstraction, while still retaining the speed and automation of today's Boolean engines.

Roadmap

- SMT and Theories
- Combining Theories
- From SAT to SMT
- Building on SMT
- Successes and Challenges

Satisfiability Modulo Theories

It is important to make a distinction between SMT and first order satisfiability. For example, is the following sentence satisfiable?

$$\mathit{read}\,(\mathit{write}\,(a, i, v), i) \neq v$$

Satisfiability Modulo Theories

It is important to make a distinction between SMT and first order satisfiability. For example, is the following sentence satisfiable?

$$\textit{read}\,(\textit{write}\,(a, i, v), i) \neq v$$

If the set of allowable models is unrestricted, then the answer is yes.

Satisfiability Modulo Theories

It is important to make a distinction between SMT and first order satisfiability. For example, is the following sentence satisfiable?

$$\mathit{read}\,(\mathit{write}\,(a, i, v), i) \neq v$$

If the set of allowable models is unrestricted, then the answer is yes.

However, if we only consider models that obey the axioms for *read* and *write* then the answer is no.

Satisfiability Modulo Theories

For a theory T, the T-satisfiability problem consists of deciding whether there exists a model $\mathcal{A}$ and variable assignment α such that $(\mathcal{A}, \alpha) \models T \cup \varphi$ for an given formula φ.

Another way to think of this is as a restriction on the models we are willing to consider when trying to satisfy φ.

Some recent work in SMT considers a theory to be a collection of models rather than a set of sentences.

Theories

In principle, SMT can be applied to any theory T.

In practice, when people talk about SMT, they are usually referring to a small set of specific theories.

We will consider a few examples of theories which are of particular interest in verification applications (MZ03).

All of these assume first order logic with equality.

The Theory $T_{\mathcal{E}}$ of Equality

The theory $T_{\mathcal{E}}$ of equality is the empty theory.

The theory does not restrict the possible values of symbols in any way. For this reason, it is sometimes called the theory of equality with uninterpreted functions (EUF).

The satisfiability problem for $T_{\mathcal{E}}$ is just the satisfiability problem for first order logic, which is undecidable.

The satisfiability problem for conjunctions of literals in $T_{\mathcal{E}}$ is decidable in polynomial time using congruence closure.

The Theory $T_{\mathcal{Z}}$ of Integers

Let $\Sigma_{\mathcal{Z}}$ be the signature $(0, 1, +, -, \leq)$.

Let $\mathcal{A}_{\mathcal{Z}}$ be the standard model of the integers with domain $\mathcal{Z}$.

Then $T_{\mathcal{Z}}$ is defined to be the set of all $\Sigma_{\mathcal{Z}}$-sentences true in the model $A_{\mathcal{Z}}$.

As showed by Presburger in 1929, the general satisfiability problem for $T_{\mathcal{Z}}$ is decidable, but its complexity is triply-exponential.

The quantifier-free satisfiability problem for $T_{\mathcal{Z}}$ is “only” NP-complete.

The Theory $T_{\mathcal{Z}}$ of Integers

Let $\Sigma_{\mathcal{Z}}^{\times}$ be the same as $\Sigma_{\mathcal{Z}}$ with the addition of the symbol $\times$ for multiplication, and define $\mathcal{A}_{\mathcal{Z}}^{\times}$ and $T_{\mathcal{Z}}^{\times}$ in the obvious way.

The satisfiability problem for $T_{\mathcal{Z}}^{\times}$ is undecidable (a consequence of Gödel's incompleteness theorem).

In fact, even the quantifier-free satisfiability problem for $T_{\mathcal{Z}}^{\times}$ is undecidable.

The Theory $T_{\mathcal{R}}$ of Reals

Let $\Sigma_{\mathcal{R}}$ be the signature $(0, 1, +, -, \leq)$.

Let $\mathcal{A}_{\mathcal{R}}$ be the standard model of the reals with domain $\mathcal{R}$.

Then $T_{\mathcal{R}}$ is defined to be the set of all $\Sigma_{\mathcal{R}}$-sentences true in the model $A_{\mathcal{R}}$.

The satisfiability problem for $T_{\mathcal{R}}$ is decidable, but the complexity is doubly-exponential.

The quantifier-free satisfiability problem for conjunctions of literals (atomic formulas or their negations) in $T_{\mathcal{R}}$ is solvable in polynomial time, though exponential methods (like Simplex or Fourier-Motzkin) tend to perform best in practice.

The Theory $T_{\mathcal{R}}$ of Reals

Let $\Sigma_{\mathcal{R}}^{\times}$ be the same as $\Sigma_{\mathcal{R}}$ with the addition of the symbol $\times$ for multiplication, and define $\mathcal{A}_{\mathcal{R}}^{\times}$ and $T_{\mathcal{R}}^{\times}$ in the obvious way.

In contrast to the theory of integers, the satisfiability problem for $T_{\mathcal{R}}^{\times}$ is decidable though the complexity is inherently doubly-exponential.

The Theory $T_{\mathcal{A}}$ of Arrays

Let $\Sigma_{\mathcal{A}}$ be the signature $(\mathit{read}, \mathit{write})$.

Let $\Lambda_{\mathcal{A}}$ be the following axioms:

$$\forall a\, \forall i\, \forall v\, (\mathit{read}(\mathit{write}(a, i, v), i) = v)$$
$$\forall a\, \forall i\, \forall j\, \forall v\, (i \neq j \rightarrow \mathit{read}(\mathit{write}(a, i, v), j) = \mathit{read}(a, j))$$
$$\forall a\, \forall b\, ((\forall i\, (\mathit{read}(a, i) = \mathit{read}(b, i))) \rightarrow a = b)$$

Then $T_{\mathcal{A}} = \mathit{Cn}\ \Lambda_{\mathcal{A}}$.

The satisfiability problem for $T_{\mathcal{A}}$ is undecidable, but the quantifier-free satisfiability problem for $T_{\mathcal{A}}$ is decidable (the problem is NP-complete).

Theories of Inductive Data Types

An inductive data type (IDT) defines one or more constructors, and possibly also selectors and testers.

Example: *list* **of** *int*

- Constructors: $\mathit{cons} : (\mathit{int}, \mathit{list}) \longrightarrow \mathit{list}$, $\mathit{null} : \mathit{list}$
- Selectors: $\mathit{car} : \mathit{list} \longrightarrow \mathit{int}$, $\mathit{cdr} : \mathit{list} \longrightarrow \mathit{list}$
- Testers: $\mathit{is\_cons}$, $\mathit{is\_null}$

The first order theory of a inductive data type associates a function symbol with each constructor and selector and a predicate symbol with each tester.

Example: $\forall\, x : \mathit{list}.\ (x = \mathit{null} \lor \exists\, y : \mathit{int}, z : \mathit{list}.\ x = \mathit{cons}(y, z))$

Theories of Inductive Data Types

An inductive data type (IDT) defines one or more constructors, and possibly also selectors and testers.

Example: *list* **of** *int*

- Constructors: $cons : (int, list) \longrightarrow list$, $null : list$
- Selectors: $car : list \longrightarrow int$, $cdr : list \longrightarrow list$
- Testers: is_cons, is_null

For IDTs with a single constructor, a conjunction of literals is decidable in polynomial time (Opp80).

For more general IDTs, the problem is NP complete, but reasonbly efficient algorithms exist in practice (ZSM04a; ZSM04b; BST07).

Other Interesting Theories

Some other interesting theories include:

- Theories of bit-vectors
  (CMR97; Möl97; BDL98; BP98; EKM98; GBD05)
- Fragments of set theory (CZ00)
- Theories of pointers and reachability
  (RBH07; YRS^{+}06; LQ08)

Roadmap

- SMT and Theories
- Combining Theories
- From SAT to SMT
- Building on SMT
- Successes and Challenges

Combining Theories

We are usually interested in a combination of theories. The standard technique for this is the Nelson-Oppen method (NO79; TH96).

Suppose that $T_1, \ldots, T_n$ are stably-infinite theories with disjoint signatures $\Sigma_1, \ldots, \Sigma_n$ and Sat_i decides T_i-satisfiability of $\Sigma_i(C)$ literals.

We wish to determine the satisfiability of a ground conjunction Γ of $\Sigma(C)$-literals.

1. Purify Γ to obtain an equisatisfiable set $\bigwedge \varphi_i$, where each φ_i is i-pure.
2. Let S be the set of shared variables (i.e. appearing in more than one φ_i).
3. For each arrangement Δ of S,

   Check $\mathit{Sat}_i(\varphi_i \wedge \Delta)$ for each i.

Combining Theories

If S is a set of terms and $\sim$ is an equivalence relation on S, then the arrangement of S induced by $\sim$ is $\{x = y \mid x \sim y\} \cup \{x \neq y \mid x \not\sim y\}$.

Example

Consider the following example in a combination of $T_{\mathcal{E}}$, $T_{\mathcal{Z}}$, and $T_{\mathcal{A}}$:

$$\neg p(y) \wedge s = \textit{write}\,(t, i, 0) \wedge x - y - z = 0 \wedge$$
$$z + \textit{read}\,(s, i) = f(x - y) \wedge p(x - f(f(z))).$$

After purification, we have the following:

$\varphi_{\mathcal{E}}$	$\varphi_{\mathcal{Z}}$	$\varphi_{\mathcal{A}}$
$\neg p(y)$	$l - z = j$	$s = \textit{write}\,(t, i, j)$
$m = f(l)$	$j = 0$	$k = \textit{read}\,(s, i)$
$p(v)$	$l = x - y$	
$n = f(f(z))$	$m = z + k$	
	$v = x - n$	

Example

$\varphi_{\mathcal{E}}$	$\varphi_{\mathcal{Z}}$	$\varphi_{\mathcal{A}}$
$\neg p(y)$	$l - z = j$	$s = \textit{write}\,(t, i, j)$
$m = f(l)$	$j = 0$	$k = \textit{read}\,(s, i)$
$p(v)$	$l = x - y$	
$n = f(f(z))$	$m = z + k$	
	$v = x - n$	

There are 12 variables in this example:

- Shared: l, z, j, y, m, k, v, n
- Unshared: x, s, t, i

There are 21147 arrangements of $\{l, z, j, y, m, k, v, n\}$.
Practical implementations have efficient strategies for searching the space of arrangements.

Roadmap

- SMT and Theories
- Combining Theories
- From SAT to SMT
- Building on SMT
- Successes and Challenges

Combining SAT and SMT

Theory solvers check the satisfiability of conjunctions of literals.

What about more general Boolean combinations of literals?

What is needed is a combination of SAT reasoning and theory reasoning.

The so-called eager approach to SMT tries to find ways of encoding everything into SAT. There are a variety of techniques, and for some theories, this works quite well.

In this talk, I will focus on the lazy combination of SAT and theory reasoning. The lazy approach is the basis for most modern SMT solvers (BDS02).

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, and M satisfies G

We write $M \models C$ to mean that for every truth assignment v, $v(M) = \mathit{true}$ implies $v(C) = \mathit{true}$.

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 :

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

Fail :

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

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 :

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

Restart :

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

Example

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

Example

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

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

$$4\ 1^{d}\ \overline{2}\ 3 \parallel$$

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Result: Unsatisfiable

Abstract DPLL Modulo Theories

The Abstract DPLL Modulo Theories framework extends the Abstract DPLL framework, providing an abstract and formal setting for reasoning about the combination of SAT and theory reasoning (NOT06).

Assume we have a theory T with signature Σ and a solver *Sat*$_T$ that can check T-satisfiability of conjunctions of Σ-literals.

Suppose we want to check the satisfiability of an arbitray (quantifier-free) Σ-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, providing an abstract and formal setting for reasoning about the combination of SAT and theory reasoning (NOT06).

Assume we have a theory T with signature Σ and a solver *Sat*$_T$ that can check T-satisfiability of conjunctions of Σ-literals.

Suppose we want to check the satisfiability of an arbitray (quantifier-free) Σ-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 $\mathsf{Sat}_T(M)$ 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)

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 SAT solver will take care of this automatically if we can 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 SAT solver will take care of this automatically if we can 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 SAT solver will take care of this automatically if we can 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 $T \models \neg M$.

We can generalize this to any clause C such that $T \models C$. 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 :

$$M \parallel F, C \quad \Longrightarrow \quad M \parallel F \quad \textbf{if} \begin{cases} F \models_T C \end{cases}$$

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.

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to 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				

Improving Abstract DPLL Modulo Theories

We will mention three ways to improve the algorithm.

- Minimizing learned clauses
- Eager conflict detection
- Theory propagation

Minimizing Learned Clauses

The main difficulty with the approach as it stands is that learned clauses can be highly redundant.

Suppose that F contains $n + 2$ propositional variables.

When a pseudo-final state is reached, M will determine a value for all $n + 2$ variables.

But what if only 2 of these assignments are already T-unsatisfiable?

If we always learn $\neg M$ in a pseudo-final state, in the worst case, 2^n clauses will be need to be learned when a single clause containing the two offending literals would have sufficed.

Minimizing Learned Clauses

To avoid this kind of redundancy, we can be smarter about the clauses that are learned with Theory Learn.

In particular, when $\mathit{Sat}_T(M)$ is called, we should make an effort to find the smallest possible subset of M which is inconsistent.

We can then learn a clause derived from only these literals.

One way to implement this is to start removing literals one at a time from M and repeatedly call Sat_T until a minimal inconsistent set is found.

However, this is typically too slow to be practical.

Minimizing Learned Clauses

A better, but more difficult way to implement this is to instrument Sat_T to keep track of which facts are used to derive an inconsistency.

We can use a data structure similar to the implication graph discussed earlier.

Alternatively, if Sat_T happens to produce proofs, the proof of unsatisfiability of M can be traversed to obtain this information.

This is the approach used in the CVC tools.

From SAT to SMT — Minimized Clauses

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

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

From SAT to SMT — Minimized Clauses

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

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

From SAT to SMT — Minimized Clauses

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

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

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

$$1\ \overline{2}^{\mathsf{d}} \parallel 1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$$

From SAT to SMT — Minimized Clauses

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

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

From SAT to SMT — Minimized Clauses

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

$\emptyset$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3}$	$\Longrightarrow$	(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)
$1\ \overline{2}^{\mathsf{d}}\ \overline{4}^{\mathsf{d}}$	$\parallel$	$1,\ \overline{2} \vee 3,\ \overline{4} \vee \overline{3},\ \overline{1} \vee 2$		

From SAT to SMT — Minimized Clauses

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

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

From SAT to SMT — Minimized Clauses

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

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

From SAT to SMT — Minimized Clauses

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

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

From SAT to SMT — Minimized Clauses

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

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

Eager Conflict Detection

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

In contrast, a more eager policy would be to check M for T-satisfiability every time M changes.

Experimental results show that this approach is significantly better.

It requires Sat_T 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.

From SAT to SMT — Eager Conflict Detection

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

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

From SAT to SMT — Eager Conflict Detection

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

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

From SAT to SMT — Eager Conflict Detection

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

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

From SAT to SMT — Eager Conflict Detection

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

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

From SAT to SMT — Eager Conflict Detection

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

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

From SAT to SMT — Eager Conflict Detection

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

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

From SAT to SMT — Eager Conflict Detection

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

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

From SAT to SMT — Eager Conflict Detection

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

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

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 a theory solver to inform the SAT solver if it happens to know that an unassigned literal is entailed by M.

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

From SAT to SMT — Theory Propagation

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

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

From SAT to SMT — Theory Propagation

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

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

From SAT to SMT — Theory Propagation

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

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

From SAT to SMT — Theory Propagation

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

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

From SAT to SMT — Theory Propagation

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

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

From SAT to SMT — Theory Propagation

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

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

Roadmap

- SMT and Theories
- Combining Theories
- From SAT to SMT
- Building on SMT
- Successes and Challenges

Building on SMT

We briefly mention two extensions.

The first is to allow the theory solver to use the SAT solver for internal case splitting (BNOT06).

We do this by allowing the learning rule to introduce new variables and terms

Extended T-Learn :

$$M \parallel F \quad \Longrightarrow \quad M \parallel F, C \quad \textbf{if} \begin{cases} \text{each atom of } C \text{ occurs in } F \text{ or in } \mathcal{L}(M) \\ F \models_T \exists^*(C) \end{cases}$$

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$\emptyset \parallel F$

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathrm{d}}, x \neq z \parallel F$

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$\emptyset \parallel F \quad \Longrightarrow \quad$ UnitProp

$x = \{y\}, x = y \cup z \parallel F$

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F$

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$\emptyset \parallel F$	$\Longrightarrow$	UnitProp
$x = \{y\}, x = y \cup z \parallel F$	$\Longrightarrow$	Decide
$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}} \parallel F$		
$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F$		

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$\emptyset \parallel F$	$\Longrightarrow$	UnitProp
$x = \{y\}, x = y \cup z \parallel F$	$\Longrightarrow$	Decide
$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}} \parallel F$	$\Longrightarrow$	UnitProp
$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F$		

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$\emptyset \parallel F$	$\Longrightarrow$	UnitProp
$x = \{y\}, x = y \cup z \parallel F$	$\Longrightarrow$	Decide
$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}} \parallel F$	$\Longrightarrow$	UnitProp
$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F$	$\Longrightarrow$	Extended T-Learn

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$$\begin{array}{lll}
\emptyset \parallel F & \Longrightarrow & \textsf{UnitProp} \\
x = \{y\}, x = y \cup z \parallel F & \Longrightarrow & \textsf{Decide} \\
x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}} \parallel F & \Longrightarrow & \textsf{UnitProp} \\
x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F & \Longrightarrow & \textsf{Extended } T\textsf{-Learn} \\
\multicolumn{3}{l}{x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)} \\
 & \Longrightarrow & \textsf{Decide}
\end{array}$$

$$\begin{array}{l} x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \\ w \in x^{\mathsf{d}} \end{array} \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$$

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$\emptyset \parallel F$	$\Longrightarrow$	UnitProp
$x = \{y\}, x = y \cup z \parallel F$	$\Longrightarrow$	Decide
$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}} \parallel F$	$\Longrightarrow$	UnitProp
$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F$	$\Longrightarrow$	Extended T-Learn

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

$\Longrightarrow$ Decide

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z$
$w \in x^{\mathsf{d}}$ $\parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

$\Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z$
$w \in x^{\mathsf{d}}, w \notin z$ $\parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$\emptyset \parallel F$ $\Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z \parallel F$ $\Longrightarrow$ Decide

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}} \parallel F$ $\Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F$ $\Longrightarrow$ Extended T-Learn

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

$\Longrightarrow$ Decide

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z$
$w \in x^{\mathsf{d}}$ $\parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

$\Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z$
$w \in x^{\mathsf{d}}, w \notin z$ $\parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

Theory: $w \in y \cup z$

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$\emptyset \parallel F \quad \Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z \parallel F \quad \Longrightarrow$ Decide

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}} \parallel F \quad \Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F \quad \Longrightarrow$ Extended T-Learn

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

$\Longrightarrow$ Decide

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z$
$w \in x^{\mathsf{d}}$ $\parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

$\Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z$
$w \in x^{\mathsf{d}}, w \notin z$ $\parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

Theory: $w \in y \cup z \quad \ldots \quad w \in y$

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$\emptyset \parallel F \quad \Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z \parallel F \quad \Longrightarrow$ Decide

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}} \parallel F \quad \Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F \quad \Longrightarrow$ Extended T-Learn

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

$\Longrightarrow$ Decide

$$\begin{array}{l} x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \\ w \in x^{\mathsf{d}} \end{array} \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$$

$\Longrightarrow$ UnitProp

$$\begin{array}{l} x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \\ w \in x^{\mathsf{d}}, w \notin z \end{array} \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$$

Theory: $w \in y \cup z \quad \ldots \quad w \in y \quad \ldots \quad w \in \emptyset$

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$$\emptyset \parallel F \quad \Longrightarrow \quad \text{UnitProp}$$

$$x = \{y\}, x = y \cup z \parallel F \quad \Longrightarrow \quad \text{Decide}$$

$$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}} \parallel F \quad \Longrightarrow \quad \text{UnitProp}$$

$$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F \quad \Longrightarrow \quad \text{Extended } T\text{-Learn}$$

$$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$$

$$\Longrightarrow \quad \text{Decide}$$

$$\begin{array}{l} x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \\ w \in x^{\mathsf{d}} \end{array} \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$$

$$\Longrightarrow \quad \text{UnitProp}$$

$$\begin{array}{l} x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \\ w \in x^{\mathsf{d}}, w \notin z \end{array} \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$$

Theory: $w \in y \cup z \quad \ldots \quad w \in y \quad \ldots \quad w \in \emptyset \quad \ldots \quad \perp$

Example: Theory of Sets

Let $F = (x = \{y\}), (x = y \cup z), (y \neq \emptyset \vee x \neq z)$:

$\emptyset \parallel F \quad \Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z \parallel F \quad \Longrightarrow$ Decide

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}} \parallel F \quad \Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F \quad \Longrightarrow$ Extended T-Learn

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z \parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

$\Longrightarrow$ Decide

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z$
$w \in x^{\mathsf{d}}$ $\parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

$\Longrightarrow$ UnitProp

$x = \{y\}, x = y \cup z, y = \emptyset^{\mathsf{d}}, x \neq z$
$w \in x^{\mathsf{d}}, w \notin z$ $\parallel F, (x = z \vee w \in x \vee w \in z), (x = z \vee w \notin x \vee w \notin z)$

Theory: $w \in y \cup z \quad \ldots \quad w \in y \quad \ldots \quad w \in \emptyset \quad \ldots \quad \bot$

$\Longrightarrow$ Backjump

$\ldots$

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:

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

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 (BdM07; GBT07).

Roadmap

- SMT and Theories
- Combining Theories
- From SAT to SMT
- Building on SMT
- Successes and Challenges

Successes

Building on fast SAT technology, SMT solvers have been improving dramatically.

The winners of this year's SMT competition are orders of magnitude faster than those of just a couple of years ago.

Current leading solvers include:

- Barcelogic (U Barcelona, Spain)
- CVC3 (NYU, U Iowa)
- MathSAT (U Trento, Italy)
- Yices (SRI)
- Z3 (Microsoft)

SMT solvers are becoming the engine of choice for an ever-increasing set of verification applications.

Successes

What are some factors in the success of SMT?

- Progress in SAT
- Standard format
- Yearly competition
- Nice abstractions
- An idea whose time has come:
  - Lots of new applications need verificaiton engines
  - Threshold of usability has been reached

Challenges

Theory

- Beyond Nelson-Oppen
- New Theories

Engineering

- Better integration of SAT in SMT
- Parallel SMT

Challenges

Embracing Incompleteness

- More techniques for quantifiers
- Nonlinear arithmetic

Reliability and Interoperability

- Producing and Checking Proofs
- Standard formats for communicating with other theorem provers
- API's, communication formats, etc.

More Information

www.smtlib.org

www.smtcomp.org

www.cs.nyu.edu/acsys/cvc3

SMT chapter in the Handbook of Satisfiability (BSST09; BHvMW09)

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

(BDL98) Clark W. Barrett, David L. Dill, and Jeremy R. Levitt. A decision procedure for bit-vector arithmetic. In *Proceedings of the* 35^{th} *Design Automation Conference (DAC '98)*, pages 522–527. Association for Computing Machinery, June 1998. San Francisco, California. *Best paper award*

(BDS02) 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

(BNOT06) Clark Barrett, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Splitting on demand in SAT modulo theories. In Miki Hermann and Andrei Voronkov, editors, *Proceedings of the* 13^{th} *International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR '06)*, volume 4246 of *Lecture Notes in Computer Science*, pages 512–526. Springer-Verlag, November 2006. Phnom Penh, Cambodia

References

(BP98) Nikolaj Bjørner and Mark C. Pichora. Deciding fixed and non-fixed size bit-vectors. In *TACAS '98: Proceedings of the 4^{th} International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 376–392. Springer-Verlag, 1998

(BSST09) Clark Barrett, Roberto Sebastiani, Sanjit Seshia, and Cesare Tinelli. *Satisfiability Modulo Theories*, chapter 26, pages 825–885. Volume 185 of Biere et al. (BHvMW09), February 2009

(BST07) Clark Barrett, Igor Shikanian, and Cesare Tinelli. An abstract decision procedure for a theory of inductive data types. *Journal on Satisfiability, Boolean Modeling and Computation*, 3:21–46, 2007

(BHvMW09) Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, February 2009

(CMR97) David Cyrluk, M. Oliver Möller, and Harald Ruess. An efficient decision procedure for the theory of fixed-size bit-vectors. In *Proceedings of the 9^{th} International Conference on Computer Aided Verification (CAV '97)*, pages 60–71. Springer-Verlag, 1997

References

(CZ00) Domenico. Cantone and Calogero G. Zarba. A new fast tableau-based decision procedure for an unquantified fragment of set theory. In Ricardo Caferra and Gernot Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *Lecture Notes in Artificial Intelligence*, pages 127–137. Springer, 2000

(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

(EKM98) Jacob Elgaard, Nils Klarlund, and Anders Möller. Mona 1.x: New techniques for WS1S and WS2S. In *Proceedings of the* 10^{th} *International Conference on Computer Aided Verification (CAV '98)*, volume 1427 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998

(GBD05) Vijay Ganesh, Sergey Berezin, and David L. Dill. A decision procedure for fixed-width bit-vectors, January 2005. Unpublished Manuscript

(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

References

(LQ08) S. K. Lahiri and S. Qadeer. Back to the future: Revisiting precise program verification using smt solvers. In *Proceedings of the 35th Annual ACM SIGPLAN - SIGACT Symposium on Principles of Programming Languages*, 2008

(Möl97) M. Oliver Möller. *Solving Bit-Vector Equations – a Decision Procedure for Hardware Verification*. PhD thesis, University of Ulm, 1997

(MZ03) Zohar Manna and Calogero Zarba. Combining decision procedures. In *Formal Methods at the Crossroads: from Panacea to Foundational Support*, volume 2787 of *Lecture Notes in Computer Science*, pages 381–422. Springer-Verlag, November 2003

(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

(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

(Opp80) Derek C. Oppen. Reasoning about recursively defined data structures. *J. ACM*, 27(3):403–411, 1980

References

(RBH07) Zvonimir Rakamarić, Jesse Bingham, and Alan J. Hu. An inference-rule-based decision procedure for verification of heap-manipulating programs with mutable data and cyclic data structures. In *Verification, Model Checking, and Abstract Interpretation: 8th International Conference*, pages 106–121. Springer, 2007. Lecture Notes in Computer Science Vol. 4349

(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

(YRS$^+$06) Greta Yorsh, Alexander Rabinovich, Mooly Sagiv, Antoine Meyer, and Ahmed Bouajjani. A logic of reachable patterns in linked data-structures. In *Proceedings of Foundations of Software Science and Computation Structures (FOSSACS '06)*, 2006

References

(ZSM04a) T. Zhang, H. B. Sipma, and Z. Manna. Decision procedures for term algebras with integer constraints. In *Proceedings of the 2^{nd} International Joint Conference on Automated Reasoning (IJCAR '04) LNCS 3097*, pages 152–167, 2004

(ZSM04b) Ting Zhang, Henny B. Sipma, and Zohar Manna. Term algebras with length function and bounded quantifier alternation. In *Proceedings of the 17^{th} International Conference on Theorem Proving in Higher Order Logics (TPHOLs '04)*, volume 3223 of *Lecture Notes in Computer Science*, pages 321–336, 2004