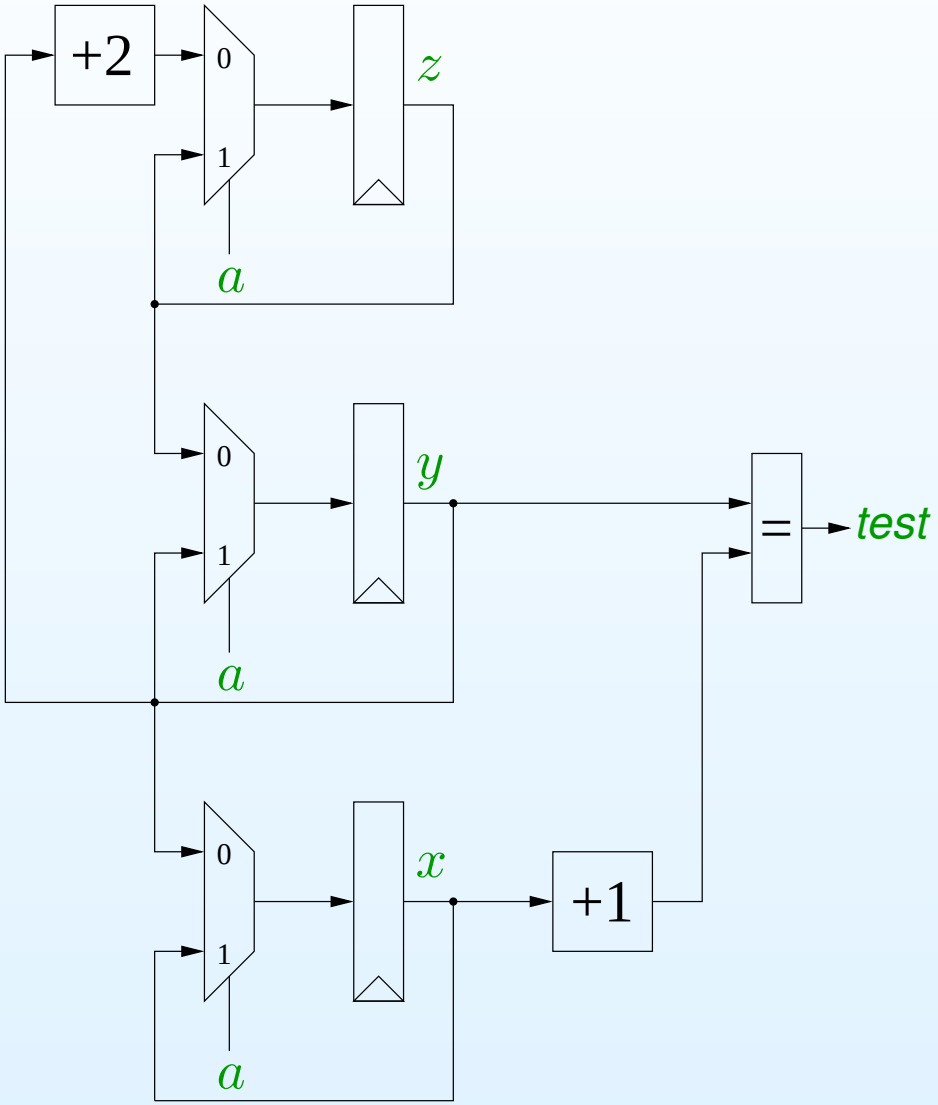
CS357: From SAT to SMT: The DPLL(T) Architecture

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:

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

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.

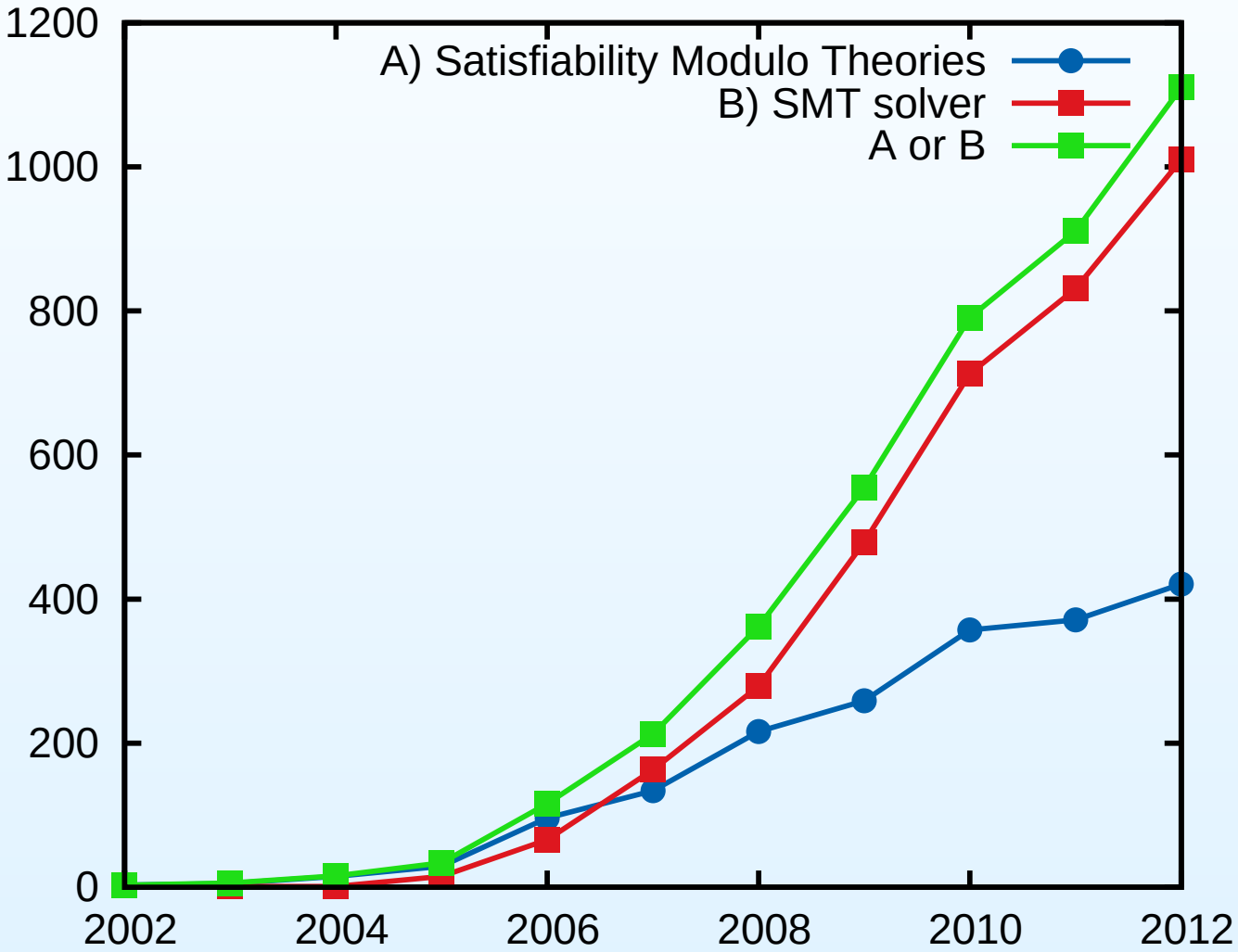
Impact of SMT

What people are saying:

- Most promising contribution to fields of software and hardware verification and test in the last five years (from the text of the HVC 2010 award)
- The biggest advance in formal methods in last 25 years (John Rushby, FMIS 2011)
- Most successful academic community related to logics and verification ... built in the last decade (editors of FMSD special issue on SMT, 2012)

Growth of SMT

Articles per year by search phrase (from Google Scholar)



Roadmap

- First-Order Logic in 5 minutes
- The SMT Problem
- Abstract DPLL
- Abstract DPLL(T)
- Extensions

SMT Solvers: Language

Whereas the language of SAT solvers is Boolean logic, the language of SMT solvers is first-order logic.

The language includes the Boolean operations of Boolean logic, but instead of propositional variables, more complicated expressions are allowed.

A first-order language must specify its signature: the set of constant, function, and predicate symbols that are allowed.

Each predicate and function symbol has an associated arity: a natural number indicating how many arguments it takes.

- Equality is a special predicate symbol of arity 2
- Constant symbols can also be thought of as functions whose arity is 0

First-Order Languages: Examples

Propositional Logic

- Equality: no
- Predicate symbols: x_1, x_2, ...
- Constant symbols: none
- Function symbols: none

Elementary Number Theory

- Equality: yes
- Predicate symbols: $<$
- Constant symbols: 0
- Function symbols: S (successor), $+$, $\times$, exp

First-Order Logic: Syntax

Terms

- Variables and constants are terms
- For each function symbol f of arity n, and terms t_1,...,t_n, $f(t_1, \ldots, t_n)$ is a term.

An atomic formula is an expression of the form: $P(t_1, \ldots, t_n)$ where P is a predicate symbol of arity n and t_1,...,t_n are terms.

An atomic formula or its negation is called a literal.

Formulas are built from literals using the Boolean operators and quantification. If α is a formula, then for every variable x,

- $\forall\, x\ \alpha$ is a formula
- $\exists\, x\ \alpha$ is a formula

First-Order Logic: Semantics

Given a signature Σ and a set V of variables, an interpretation (or model) M of Σ consists of the following:

1. A nonempty set called the domain of M, written $\mathit{dom}(M)$. Elements of $\mathit{dom}(M)$ are also called elements of M.
2. A map from each variable v in V to an element v^M of M.
3. A map from each constant c in Σ to an element c^M of M.
4. A map from each n-ary function symbol f in Σ to f^M, an n-ary function from $[\mathit{dom}(M)]^n$ to $\mathit{dom}(M)$.
5. A map from each n-ary predicate symbol p in Σ to $p^M \subseteq [\mathit{dom}(M)]^n$, an n-ary relation on the set $\mathit{dom}(M)$.

Example

Consider the signature with a single predicate symbol $\in$ and a single constant symbol $\emptyset$.

A possible model M for this signature has $\mathit{dom}(M) = \mathcal{N}$, the set of natural numbers, $\in^M = <$, and $\emptyset^M = 0$.

Now consider the sentence $\exists\, x\, \forall\, y\, \neg y \in x$.

Example

Consider the signature with a single predicate symbol $\in$ and a single constant symbol $\emptyset$.

A possible model M for this signature has $\mathit{dom}(M) = \mathcal{N}$, the set of natural numbers, $\in^M = <$, and $\emptyset^M = 0$.

Now consider the sentence $\exists\, x\, \forall\, y\, \neg y \in x$.

What does this sentence mean in this model?

Example

Consider the signature with a single predicate symbol $\in$ and a single constant symbol $\emptyset$.

A possible model M for this signature has $\mathit{dom}(M) = \mathcal{N}$, the set of natural numbers, $\in^M = <$, and $\emptyset^M = 0$.

Now consider the sentence $\exists\, x\, \forall\, y\, \neg y \in x$.

What does this sentence mean in this model?

The translation of the sentence in the model M is that there is a natural number x such that no other natural number is smaller than x.

Example

Consider the signature with a single predicate symbol $\in$ and a single constant symbol $\emptyset$.

A possible model M for this signature has $\mathit{dom}(M) = \mathcal{N}$, the set of natural numbers, $\in^M = <$, and $\emptyset^M = 0$.

Now consider the sentence $\exists\, x\, \forall\, y\, \neg y \in x$.

What does this sentence mean in this model?

The translation of the sentence in the model M is that there is a natural number x such that no other natural number is smaller than x.

Is this sentence true in the model?

Example

Consider the signature with a single predicate symbol $\in$ and a single constant symbol $\emptyset$.

A possible model M for this signature has $\mathit{dom}(M) = \mathcal{N}$, the set of natural numbers, $\in^M = <$, and $\emptyset^M = 0$.

Now consider the sentence $\exists\, x\, \forall\, y\, \neg y \in x$.

What does this sentence mean in this model?

The translation of the sentence in the model M is that there is a natural number x such that no other natural number is smaller than x.

Is this sentence true in the model?

Since 0 has this property, the sentence is true in this model.

Examples

Which models satisfy the following sentences?

1. $\forall x \forall y \, x = y$
2. $\forall x \forall y \, Qxy$
3. $\forall x \exists y \, Qxy$

Examples

Which models satisfy the following sentences?

1. $\forall x \forall y \, x = y$ Models with exactly one element.
2. $\forall x \forall y \, Qxy$
3. $\forall x \exists y \, Qxy$

Examples

Which models satisfy the following sentences?

1. $\forall x \forall y \, x = y$ Models with exactly one element.
2. $\forall x \forall y \, Qxy$ Models (A, R) where $R = A \times A$.
3. $\forall x \exists y \, Qxy$

Examples

Which models satisfy the following sentences?

1. $\forall x \forall y \, x = y$ Models with exactly one element.
2. $\forall x \forall y \, Qxy$ Models (A, R) where $R = A \times A$.
3. $\forall x \exists y \, Qxy$ Models (A, R) where $\mathit{dom}(R) = A$.

First-Order Logic: Semantics

Given a formula ϕ, we say that M satisfies ϕ and write $M \models \phi$ if ϕ is true in the model M.

A formula ϕ is valid, written $\models \phi$ iff $M \models \phi$ for every M.

A theory is a set of closed formulas (no free variables).

Given a theory T, a formula ϕ is

1. T-valid if $M \models \phi$ for all models M of T;
2. T-satisfiable if there exists some model M of T such that $M \models \phi$.
3. T-unsatisfiable if $M \not\models \phi$ for all models M of T.

Roadmap

- First-Order Logic in 5 minutes
- The SMT Problem
- Abstract DPLL
- Abstract DPLL(T)
- Extensions

Validity and Satisfiability Modulo Theories

The validity problem for T is the problem of deciding, for each formula ϕ, whether ϕ is T-valid.

The satisfiability problem for T, or the problem of satisfiability modulo theories is the problem of deciding, for each formula ϕ, whether ϕ is T-satisfiable.

Note that validity problems can always be reduced to satisfiability problems:

ϕ is T-valid iff $\neg\phi$ is T-unsatisfiable.

Satisfiability Modulo Theories

It is important to make a distinction between SMT and standard 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 standard 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.

Satisfiability Modulo Theories

It is important to make a distinction between SMT and standard 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

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

Here, we focus on the lazy combination of SAT and theory reasoning. The lazy approach is the basis for most modern SMT solvers.

SMT: The Big Questions

How to solve conjunctions of literals in a theory?

- Use a theory solver - upcoming lecture

How to combine theory solvers for several theories

- The Nelson-Oppen method and its variants - upcoming lecture

How to combine a theory solver and a SAT solver to reason about arbitrary formulas

- Use the DPLL(T) framework - this lecture

Roadmap

- First-Order Logic in 5 minutes
- The SMT Problem
- Abstract DPLL
- Abstract DPLL(T)
- Extensions

Abstract DPLL

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

Abstract DPLL

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

- 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.

- 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.

- 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.

- 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 :

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

PureLiteral :

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

Decide :

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

Fail :

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

Abstract DPLL Rules

Backjump :

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

Learn :

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

Forget :

$$M \parallel F, C \implies M \parallel F \quad \textbf{if} \begin{cases} F \models C \end{cases}$$

Restart :

$$M \parallel F \implies \emptyset \parallel F$$

Example

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

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

Example

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

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

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Result: Unsatisfiable

Roadmap

- First-Order Logic in 5 minutes
- The SMT Problem
- Abstract DPLL
- Abstract DPLL(T)
- Extensions

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.

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.

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 $\mathit{Sat}_T(M)$ 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 $Sat_T(M)$ reports satisfiable.

What happens if we reach a state in which: $M \parallel F$, $M \models F$, and $Sat_T(M)$ 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 $\mathit{Sat}_T(M)$ reports satisfiable.

What happens if we reach a state in which: $M \parallel F$, $M \models F$, and $\mathit{Sat}_T(M)$ 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 $\mathit{Sat}_T(M)$ reports satisfiable.

What happens if we reach a state in which: $M \parallel F$, $M \models F$, and $\mathit{Sat}_T(M)$ 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 $\mathit{Sat}_T(M)$ reports satisfiable.

What happens if we reach a state in which: $M \parallel F$, $M \models F$, and $\mathit{Sat}_T(M)$ 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

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

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

From SAT to SMT

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

$$\begin{array}{rll}
\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 & \Longrightarrow \quad \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 \quad \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 \quad \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 \quad \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 \quad \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 & \Longrightarrow \quad \text{(Fail)} \\
\mathit{fail} &&
\end{array}$$

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

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

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

From SAT to SMT — Minimized Clauses

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

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

From SAT to SMT — Minimized Clauses

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

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

From SAT to SMT — Minimized Clauses

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

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

From SAT to SMT — Minimized Clauses

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

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

From SAT to SMT — Minimized Clauses

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

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

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

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

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

From SAT to SMT — Minimized Clauses

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

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

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

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

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

From SAT to SMT — Eager Conflict Detection

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

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

From SAT to SMT — Eager Conflict Detection

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

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

From SAT to SMT — Eager Conflict Detection

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

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

From SAT to SMT — Eager Conflict Detection

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

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

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

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

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

From SAT to SMT — Eager Conflict Detection

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

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

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

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

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

From SAT to SMT — Theory Propagation

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

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

From SAT to SMT — Theory Propagation

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

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

From SAT to SMT — Theory Propagation

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

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

From SAT to SMT — Theory Propagation

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

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

From SAT to SMT — Theory Propagation

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

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

Roadmap

- First-Order Logic in 5 minutes
- The SMT Problem
- Abstract DPLL
- Abstract DPLL(T)
- Extensions

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.

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

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

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

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

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

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

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

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

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

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

$\emptyset \parallel F$ $\Longrightarrow$ 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: $w \in y \cup z \quad \ldots \quad w \in y$

Example: Theory of Sets

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

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

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

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

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

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

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

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

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

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

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

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

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.

- 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_$\exists$:

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

Inst_$\forall$:

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

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 have been adopted and improved by modern SMT solvers.

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:

- Boolector (Johannes Kepler U, Austria)
- CVC4 (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.