Logical Aspects of AI

Lecture 3 - Combining decision procedures

Sylvain Conchon

LRI (UMR 8623), Université Paris-Sud
Équipe Toccata, INRIA Saclay – Île-de-France
Lecture 3

1. Theory Combination

2. Quantifiers

3. Extra material
THEORY COMBINATION
In CDCL(T), the theory T is usually combination of theories

For instance,

\[ x + 2 = y \implies f(\text{read}(\text{write}(a, x, 3), y - 2)) = f(y - x + 1) \]
Union of theories

Given two signatures \( \Sigma_1 \) and \( \Sigma_2 \), and two consistent theories \( T_1 \) and \( T_2 \) over \( \Sigma_1 \) and \( \Sigma_2 \), respectively.
Union of theories

Given two signatures \( \Sigma_1 \) and \( \Sigma_2 \), and two consistent theories \( T_1 \) and \( T_2 \) over \( \Sigma_1 \) and \( \Sigma_2 \), respectively

- Is the union \( T_1 \cup T_2 \) consistent?
Given two signatures $\Sigma_1$ and $\Sigma_2$, and two consistent theories $T_1$ and $T_2$ over $\Sigma_1$ and $\Sigma_2$, respectively

- Is the union $T_1 \cup T_2$ consistent?

Undecidable in the general case
Union of theories

Given two signatures $\Sigma_1$ and $\Sigma_2$, and two consistent theories $T_1$ and $T_2$ over $\Sigma_1$ and $\Sigma_2$, respectively

- Is the union $T_1 \cup T_2$ consistent?

  Undecidable in the general case

- Can we build a decision procedure for $T_1 \cup T_2$ from decision procedures of $T_1$ and $T_2$?
Given two signatures $\Sigma_1$ and $\Sigma_2$, and two consistent theories $\mathcal{T}_1$ and $\mathcal{T}_2$ over $\Sigma_1$ and $\Sigma_2$, respectively

- Is the union $\mathcal{T}_1 \cup \mathcal{T}_2$ consistent?

  Undecidable in the general case

- Can we build a decision procedure for $\mathcal{T}_1 \cup \mathcal{T}_2$ from decision procedures of $\mathcal{T}_1$ and $\mathcal{T}_2$?

  Methods exist only for restricted classes of theories
Given two \textbf{consistent} theories $T_1$ and $T_2$ over $\Sigma_1$ and $\Sigma_2$, respectively

\textbf{Theorem:}

$T_1 \cup T_2$ is \textbf{not consistent} if there exists a formula $\varphi$ over $\Sigma_1 \cap \Sigma_2$ such that $T_1 \models \varphi$ and $T_2 \models \neg \varphi$
When $\Sigma_1$ and $\Sigma_2$ are disjoint signatures

**Theorem [Tinelli]:**

$T_1 \cup T_2$ is consistent if $T_1$ and $T_2$ have a infinite model
Lowenheim-Skolem Upward Theorem

Given a signature $\Sigma$ and a theory $\mathcal{T}$ over $\Sigma$.

**Theorem:**

If $\mathcal{T}$ has an infinite model of cardinality $\kappa$, then $\mathcal{T}$ has a model of cardinality $\kappa'$, for any $\kappa' \geq \kappa$

- used to align cardinalities of models
- useful to prove completeness of combination methods
Proof. Let $A_1$ and $A_2$ models of $T_1$ and $T_2$, respectively.
Proof. Let $A_1$ and $A_2$ models of $T_1$ and $T_2$, respectively

By the Lowenheim-Skolem Upward theorem, if $T_1$ and $T_2$ have an infinite model then they also have models of any infinite cardinality. We can thus assume that $A_1$ and $A_2$ have the same cardinality.
Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ models of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively.

By the Lowenheim-Skolem Upward theorem, if $\mathcal{T}_1$ and $\mathcal{T}_2$ have an infinite model then they also have models of any infinite cardinality. We can thus assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same cardinality.

By the Joint Consistency theorem, if $\mathcal{T}_1 \cup \mathcal{T}_2$ is not consistent then there exists a formula $\psi$ such that $\mathcal{A}_1 \models \psi$ et $\mathcal{A}_2 \models \neg \psi$ (1).
Union of Disjoint Theories

Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ models of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively.

By the **Lowenheim-Skolem Upward** theorem, if $\mathcal{T}_1$ and $\mathcal{T}_2$ have an infinite model then they also have models of any infinite cardinality. We can thus assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same cardinality.

By the **Joint Consistency** theorem, if $\mathcal{T}_1 \cup \mathcal{T}_2$ is not consistent then there exists a formula $\psi$ such that $\mathcal{A}_1 \models \psi$ et $\mathcal{A}_2 \models \neg \psi$ \((1)\).

Now, as $\Sigma_1$ and $\Sigma_2$ are disjoint, $\mathcal{T}_1 \cap \mathcal{T}_2$-formulas can only be equational formulas, that is $\psi$ only contains literals of the form $x = y$ or $x \neq y$. 
Union of Disjoint Theories

Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ models of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively.

By the Lowenheim-Skolem Upward theorem, if $\mathcal{T}_1$ and $\mathcal{T}_2$ have an infinite model then they also have models of any infinite cardinality. We can thus assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same cardinality.

By the Joint Consistency theorem, if $\mathcal{T}_1 \cup \mathcal{T}_2$ is not consistent then there exists a formula $\psi$ such that $\mathcal{A}_1 \models \psi$ et $\mathcal{A}_2 \models \neg \psi \ (1)$.

Now, as $\Sigma_1$ and $\Sigma_2$ are disjoint, $\mathcal{T}_1 \cap \mathcal{T}_2$-formulas can only be equational formulas, that is $\psi$ only contains literals of the form $x = y$ or $x \neq y$.

It is a well-known result in model theory that the reducts of any two models to the empty signature are isomorphic when they have the same cardinality (any one-to-one correspondence works)
Union of Disjoint Theories

Proof. Let $A_1$ and $A_2$ models of $T_1$ and $T_2$, respectively.

By the Lowenheim-Skolem Upward theorem, if $T_1$ and $T_2$ have an infinite model then they also have models of any infinite cardinality. We can thus assume that $A_1$ and $A_2$ have the same cardinality.

By the Joint Consistency theorem, if $T_1 \cup T_2$ is not consistent then there exists a formula $\psi$ such that $A_1 \models \psi$ et $A_2 \models \neg \psi$ (1).

Now, as $\Sigma_1$ and $\Sigma_2$ are disjoint, $T_1 \cap T_2$-formulas can only be equational formulas, that is $\psi$ only contains literals of the form $x = y$ or $x \neq y$.

It is a well-known result in model theory that the reducts of any two models to the empty signature are isomorphic when they have the same cardinality (any one-to-one correspondence works).

Consequently, either $A_1$ and $A_2$ are model of $\psi$ or neither of them does, which contradicts (1).
Assume $\mathcal{T}_1$ is the theory of (integer) arithmetic and $\mathcal{T}_2$ the theory of arrays, defined by the following axioms

\[
\begin{align*}
    v[i \leftarrow e][i] &= e \\
    i \neq j &\Rightarrow v[i \leftarrow e][j] = v[i]
\end{align*}
\]

Is the following formula $\psi$ $(\mathcal{T}_1 \cup \mathcal{T}_2)$-satisfiable?

\[
v[i \leftarrow v[j]][i] \neq v[i] \land i + j \leq 2j \land j + 4i \leq 5i
\]
First step: decompose $\psi$ in two pure formulas $\psi_1$ and $\psi_2$ of $T_1$ and $T_2$

$$
\psi_1 = v[i \leftarrow v[j]][i] \neq v[i] \\
\psi_2 = i + j \leq 2j \land j + 4i \leq 5i
$$
Naive Combination of Decision Procedures

\[ \psi_1 = v[i \leftarrow v[j]][i] \neq v[i] \]
\[ \psi_2 = i + j \leq 2j \land j + 4i \leq 5i \]

Second step: use the decision procedures of \( T_1 \) and \( T_2 \) to determine the satisfiability of \( \psi_1 \) and \( \psi_2 \), respectively.
Naive Combination of Decision Procedures

\[
\psi_1 = v[i \leftarrow v[j]](i) \neq v[i] \\
\psi_2 = i + j \leq 2j \land j + 4i \leq 5i
\]

Second step: use the decision procedures of \( T_1 \) and \( T_2 \) to determine the satisfiability of \( \psi_1 \) and \( \psi_2 \), respectively

- \( \psi_1 \) is satisfiable
Naive Combination of Decision Procedures

\[ \psi_1 = v[i \leftarrow v[j]][i] \neq v[i] \]
\[ \psi_2 = i + j \leq 2j \land j + 4i \leq 5i \]

Second step: use the decision procedures of \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \) to determine the satisfiability of \( \psi_1 \) and \( \psi_2 \), respectively

- \( \psi_1 \) is satisfiable
- \( \psi_2 \) is satisfiable
Naive Combination of Decision Procedures

$$\psi_1 = v[i \leftarrow v[j]][i] \neq v[i]$$
$$\psi_2 = i + j \leq 2j \land j + 4i \leq 5i$$

Second step: use the decision procedures of $T_1$ and $T_2$ to determine the satisfiability of $\psi_1$ and $\psi_2$, respectively

- $\psi_1$ is satisfiable
- $\psi_2$ is satisfiable

But is $\psi$ satisfiable?
Naive Combination of Decision Procedures

ψ = v[i ← v[j]][i] ≠ v[i] ∧ i + j ≤ 2j ∧ j + 4i ≤ 5i

ψ is unsatisfiable

Proof.
Naive Combination of Decision Procedures

\[ \psi = v[i \leftarrow v[j]][i] \neq v[i] \land i + j \leq 2j \land j + 4i \leq 5i \]

\( \psi \) is unsatisfiable

Proof.

\[ i + j \leq 2j \land j + 4i \leq 5i \text{ implies } i = j \]

\[ v[i \leftarrow v[j]][i] \neq v[i] \land i = j \text{ implies } v[i] \neq v[i] \]
Naive Combination of Decision Procedures

\[ \psi = v[i \leftarrow v[j]][i] \neq v[i] \land i + j \leq 2j \land j + 4i \leq 5i \]

\( \psi \) is unsatisfiable

Proof.

\( i + j \leq 2j \land j + 4i \leq 5i \) implies \( i = j \)

\( v[i \leftarrow v[j]][i] \neq v[i] \land i = j \) implies \( v[i] \neq v[i] \)

The problem is that \( \psi_1 \) and \( \psi_2 \) are not independent, they are sharing variables and the equality predicate

Solution: compute the implied formula \( i = j \)
Craig Interpolation Theorem

Given two *pure* formulas $\varphi_1$ and $\varphi_2$ over $\Sigma_1$ and $\Sigma_2$, respectively

**Theorem:**

If $\varphi_1 \land \varphi_2$ is $T_1 \cup T_2$-unsatisfiable then there exists a sentence $\psi$ over $\Sigma_1 \cap \Sigma_2$ such that

1) $\models_{T_1} \varphi_1 \Rightarrow \psi$

2) $\varphi_2 \land \psi$ is $T_2$-unsatisfiable
Craig Interpolation Theorem

Given two pure formulas $\varphi_1$ and $\varphi_2$ over $\Sigma_1$ and $\Sigma_2$, respectively

Theorem:

If $\varphi_1 \land \varphi_2$ is $T_1 \cup T_2$-unsatisfiable then there exists a sentence $\psi$ over $\Sigma_1 \cap \Sigma_2$ such that

1) $\models_{T_1} \varphi_1 \Rightarrow \psi$

2) $\varphi_2 \land \psi$ is $T_2$-unsatisfiable

$\psi$ is an interpolant

Computing interpolants is the basis of combination methods like Nelson-Oppen
Nelson-Oppen (NO) Combination Methods

Let $\Sigma_1$ and $\Sigma_2$ two disjoint signatures

Input. $\psi$ a conjunction of literals over $\Sigma_1 \cup \Sigma_2$

Step 1. Purify $\psi$ into a equisatisfiable formula $\psi_1 \land \psi_2$ such that $\psi_i \in \Sigma_i$

Step 2. Guess a partition of the variables of $\psi_1$ and $\psi_2$. Express it as a conjunction of literals $\varphi$.

Example. The partition $\{x_1\}, \{x_2, x_3\}, \{x_4\}$ is represented as $x_1 \neq x_2, x_1 \neq x_4, x_2 \neq x_4, x_2 = x_3$

Step 3. Decide whether $\psi_i \land \varphi$ is satisfiable by using individual decision procedures

Output. yes if all the decision procedures return yes, no otherwise
A simple and elegant correctness proof of NO has been given by Tinelli and Harandi in 1996.

Correctness becomes an issue for deterministic and efficient implementations:

- purification with term *sharing*
- *deducing* the equalities to be shared
- theory state *normalization*
- deduction by *lookup*
- *Relevant equation* selection
- etc.
We present a deterministic version of NO at a description level high-enough to enjoy a simple correctness proof, and low-enough to describe crucial implementation details.

- The algorithm is described as a set of inference rules
- Specific rules for optimizations
- Strategies as regular expressions
- Shostak pattern for efficient deduction
The internal state of the algorithm is represented by configurations of the form

$$\langle V \mid \Delta \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle$$
The internal state of the algorithm is represented by configurations of the form

\[
\langle V \mid \Delta \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle
\]

- \( \Gamma \) is a set of literals of the form \( a = b \) or \( a \neq b \), where \( a \) and \( b \) are terms in the union of theories \( \mathcal{T}_1 \cup \cdots \cup \mathcal{T}_n \)
The internal state of the algorithm is represented by configurations of the form

\[ \langle V \mid \Delta \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle \]

- \( \Gamma \) is a set of literals of the form \( a = b \) or \( a \neq b \), where \( a \) and \( b \) are terms in the union of theories \( \mathcal{T}_1 \cup \cdots \cup \mathcal{T}_n \)
- \( \Delta \) is a set of literals of the form \( x = y \) or \( x \neq y \), where \( x \) and \( y \) are variables
The internal state of the algorithm is represented by configurations of the form

\[ \langle V \mid \Delta \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle \]

- \( \Gamma \) is a set of literals of the form \( a = b \) or \( a \neq b \), where \( a \) and \( b \) are terms in the union of theories \( \mathcal{T}_1 \cup \cdots \cup \mathcal{T}_n \)
- \( \Delta \) is a set of literals of the form \( x = y \) or \( x \neq y \), where \( x \) and \( y \) are variables
- Each \( \Phi_i \) is a set equations of the form \( x = a \) where \( x \) is a variable and \( a \) is a term in \( \mathcal{T}_i \)
The internal state of the algorithm is represented by configurations of the form

\[ \langle V \mid \Delta \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle \]

- \( \Gamma \) is a set of literals of the form \( a = b \) or \( a \neq b \), where \( a \) and \( b \) are terms in the union of theories \( T_1 \cup \cdots \cup T_n \).
- \( \Delta \) is a set of literals of the form \( x = y \) or \( x \neq y \), where \( x \) and \( y \) are variables.
- Each \( \Phi_i \) is a set of equations of the form \( x = a \) where \( x \) is a variable and \( a \) is a term in \( T_i \).
- \( V \) is a set of variables that appear in \( \Gamma \) and \( \Delta \).
The internal state of the algorithm is represented by configurations of the form

\[ \langle V \mid \Delta \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle \]

- \( \Gamma \) is a set of literals of the form \( a = b \) or \( a \neq b \), where \( a \) and \( b \) are terms in the union of theories \( T_1 \cup \cdots \cup T_n \)
- \( \Delta \) is a set of literals of the form \( x = y \) or \( x \neq y \), where \( x \) and \( y \) are variables
- Each \( \Phi_i \) is a set equations of the form \( x = a \) where \( x \) is a variable and \( a \) is a term in \( T_i \)
- \( V \) is a set of variables that appear in \( \Gamma \) and \( \Delta \)

We also use the symbol \( \perp \) as a configuration
**Abstract**

\[
\langle V \mid \Delta \mid \Gamma \cup \{a = b\} \mid \ldots, \Phi_i, \ldots \rangle
\]

\[
\langle V \cup \{z\} \mid \Delta \mid \Gamma \cup \{a[\pi \mapsto z] = b\} \mid \ldots, \Phi_i \cup \{z = a_\pi\}, \ldots \rangle
\]

where \(a_\pi \in T_i\) and \(z\) is a fresh variable

**Share**

\[
\langle V \mid \Delta \mid \Gamma \cup \{a = b\} \mid \Phi_1, \ldots, \Phi_n \rangle
\]

\[
\langle V \mid \Delta \mid \Gamma \cup \{a[\pi \mapsto z] = b\} \mid \Phi_1, \ldots, \Phi_n \rangle
\]

where \(a_\pi \in T_i\) and \(z\) is a fresh variable and \(\Phi_i, \Delta \models_{T_i} z = a_\pi\)
Equality Propagation

**Arrange**

\[
\frac{\langle V \mid \Delta \mid \Gamma \cup \{x \Join y\} \mid \Phi_1, \ldots, \Phi_n \rangle}{\langle V \mid \Delta \cup \{x \Join y\} \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle}
\]

**Deduce**

\[
\frac{\langle V \mid \Delta \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle}{\langle V \mid \Delta \cup x = y \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle}
\]

if \( \Phi_i, \Delta \models_{\mathcal{T}_i} x = y \) and \( \Delta \not\models x = y \)

**Contradict**

\[
\frac{\langle V \mid \Delta \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle}{\bot}
\]

if \( \Phi_i \wedge \Delta \) is not \( \mathcal{T}_i \)-satisfiable
Example

\[ f(x) = x \land f(2x - f(x)) \neq x \]

<table>
<thead>
<tr>
<th>( V )</th>
<th>( \Delta )</th>
<th>( \Gamma )</th>
<th>( \Phi_1 )</th>
<th>( \Phi_2 )</th>
<th>Rule</th>
</tr>
</thead>
</table>
| \( x \) | \( \emptyset \) | \[ f(x) = x \]
|  |  | \( f(2x - f(x)) \neq x \) | \( \emptyset \) | \( \emptyset \) | |
| \( x, y \) | \( \emptyset \) | \( y = x \)
|  |  | \( f(2x - f(x)) \neq x \) | \( y = f(x) \) | \( \emptyset \) | \( \text{Ab}_1 \) |
| \( x, y \) | \( y = x \) | \( y = x \)
|  |  | \( f(2x - f(x)) \neq x \) | \( y = f(x) \) | \( \emptyset \) | \( \text{Ar} \) |
| \( x, y \) | \( y = x \) | \( y = x \)
|  |  | \( f(2x - y) \neq x \) | \( y = f(x) \) | \( \emptyset \) | \( \text{Sh}_1 \) |
| \( x, y, z \) | \( y = x \) | \( f(z) \neq x \)
|  |  | \( y = f(x) \) | \( z = 2x - y \) | \( \text{Ab}_2 \) |
| \( x, y, z, u \) | \( y = x \) | \( u \neq x \)
|  |  | \( y = f(x) \)
|  |  | \( u = f(z) \) | \( z = 2x - y \) | \( \text{Ab}_1 \) |
| \( x, y, z, u \) | \( y = x \)
|  |  | \( u \neq x \) | \( \emptyset \) | \( \emptyset \) | \( \text{De}_2 \) |
| \( x, y, z, u \) | \( y = x \)
|  |  | \( u \neq x \) | \( \emptyset \) | \( \emptyset \) | \( \text{De}_2 \) |
| \( x, y, z, u \) | \( y = x \)
|  |  | \( u \neq x \) | \( \emptyset \) | \( \emptyset \) | \( \text{De}_2 \) |
| \( x, y, z, u \) | \( y = x \)
|  |  | \( z = x \) | \( \emptyset \) | \( \emptyset \) | \( \text{De}_2 \) |
Rule **Deduct** is applicable only if a theory can always infer a **unique** equality from $\Phi_i \land \Delta$. This property is called **convexity**.

**Convex Theories.**

A theory $\mathcal{T}$ is **convex** if and only if for all finite set $\Gamma$ of literals, and for all non-empty disjunction $\bigvee_{i \in I} x_i = y_i$ of variables

$$\Gamma \models_{\mathcal{T}} \bigvee_{i \in I} x_i = y_i \quad \text{iff} \quad \Gamma \models_{\mathcal{T}} x_i = y_i \quad \text{for some} \ i \in I$$
Convexity

Some theories are convex

- Linear rational arithmetic
- Equational theories

Many theories are not convex

- Linear integer arithmetic
  \[ y = 1, z = 2, 1 \leq x \leq 2 \models x = y \lor x = z \]

- Non linear arithmetic
  \[ x^2 = 1, y = 1, z = -1 \models x = y \lor x = z \]

- Theory of Bit-vectors

- Theory of Arrays
  \[ v_1 = a[i \leftarrow v_2][j], v_3 = a[j] \models v_1 = v_2 \lor v_1 = v_3 \]
A configuration $\langle V \mid \Delta \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle$ is satisfiable if the formula $\Gamma \land \Phi_1 \land \cdots \land \Phi_n \land \Delta$ is satisfiable. $\bot$ is unsatisfiable.

Satisfiability of a conjunction of literals $\Gamma$ is thus equivalent to the satisfiability of the initial configuration $\langle V \mid \emptyset \mid \Gamma \mid \emptyset \rangle$.

We write $C \Rightarrow C'$ if the configuration $C$ can be reduced to $C'$ by one of the inference rules. A configuration that cannot be reduced is called irreducible. It is proper if it is not $\bot$.

**Theorem [Correctness]**

A set of literals $\Gamma$ is satisfiable iff there exists an irreducible and proper configuration $C$ such that $\langle V \mid \emptyset \mid \Gamma \mid \emptyset \rangle \Rightarrow^* C$. 
Correctness: Termination

Theorem

The reduction relation \( \Rightarrow \) is terminating

Proof.

We consider the measure \((|\Gamma|, |\Delta|)\) on configurations defined by

- \(|\Gamma|\) is the sum total of sizes of its terms
- \(|\Delta|\) is the number of equivalence classes represented by \(\Delta\)

Notice that given a fix number of variables, the ordering on \(|\Delta|\) is well-founded (there is only a finite number of classes)

It is immediate to see that

- \(|\Gamma|\) decreases by **Abstract** and **Arrange**
- \(|\Gamma|\) is constant by **Deduct** and \(|\Delta|\) decreases
Correctness (III): Finite Models

Assume $\mathcal{T}_1$ is a $\Sigma_1$-theory whose models have at most 2 elements, and $\mathcal{T}_2$ is a $\Sigma_2$-theory admitting models of any cardinality

- Let $f \in \Sigma_1$ and $g \in \Sigma_2$ such that
  \[ \not\models_{\mathcal{T}_1} \forall x, y. f(x) = f(y) \quad \text{and} \quad \not\models_{\mathcal{T}_2} \forall x, y. g(x) = g(y) \]
  ($f$ and $g$ are not constant functions)
- Let $\Gamma = \{ f(x) \neq f(y), g(x) \neq g(z), g(y) \neq g(z) \}$

Running NO from $\langle V \mid \emptyset \mid \Gamma \mid \emptyset \rangle$, we reach the final configuration

\[ \langle V' \mid \Delta \mid \emptyset \mid \Phi_1, \Phi_2 \rangle \]

with
\[ \Delta = \{ x_1 \neq y_1, x_2 \neq z_2, y_2 \neq z_2 \} \]
\[ \Phi_1 = \{ x_1 = f(x), y_1 = f(y) \} \]
\[ \Phi_2 = \{ x_2 = g(x), y_2 = g(y), z_2 = g(z) \} \]
Correctness (III): Finite Models (cont)

$x$ and $y$ are the only variables shared variables, and only $x = y$ or $x \neq y$ can be shared

- $x = y$ is impossible since the procedure would have reach $\bot$
- with $x \neq y$, $\Delta \cup \Phi_1$ and $\Delta \cup \Phi_2$ are satisfiables, but it is straightforward to check that

$$\Gamma \models_{\mathcal{T}_1 \cup \mathcal{T}_2} x \neq y \land x \neq z \land y \neq z$$

$\Gamma$ is thus unsatisfiable since $\mathcal{T}_1 \cup \mathcal{T}_2$ (as $\mathcal{T}_1$) has only models with at most 2 elements.

NO is unsound for theories with finite models only
NO can only combine decision procedures for theories that have infinite models

**Stably Infinite Theories**

A theory $\mathcal{T}$ is stably infinite if every $\mathcal{T}$-satisfiable formula is satisfiable in an infinite model

Example. Theories with only finite models are not stably infinite

$$\forall x, y, z. x = y \lor x = z \lor y = z$$

This condition also ensures that union of two **consistent**, **disjoint**, stably infinite theories is **consistent**
Correctness (V): Tinelli-Harandi’s theorem

An arrangement $\Delta(V)$ of a set of variables $V$ is a set of formulas of the form $x = y$ or $x \neq y$ such that for all pairs of variables $x, y \in V$ we have $\Delta(V) \models x = y$ or $\Delta(V) \models x \neq y$

Given two stably-infinite theories $\mathcal{T}_1$ and $\mathcal{T}_2$ over two disjoint signatures $\Sigma_1$ and $\Sigma_2$

**Theorem**[Tinelli-Harandi (1996)]

Let $\Phi_1$ and $\Phi_2$ two sets of $\Sigma_1$ and $\Sigma_2$-literals. Let $V$ be the set of the variables shared by $\Phi_1$ and $\Phi_2$, and $\Delta(V)$ an arrangement on $V$. If $\Phi_1 \land \Delta(V)$ is $\mathcal{T}_1$-satisfiable and $\Phi_2 \land \Delta(V)$ is $\mathcal{T}_2$-satisfiable then $\Phi_1 \land \Phi_2$ is $(\mathcal{T}_1 \cup \mathcal{T}_2)$-satisfiable
**Correctness (VI): Irreducible**

**Lemma** Every proper irreducible configuration is satisfiable

Proof. Let \( \langle V \mid \Delta \mid \Gamma \mid \Phi_1, \ldots, \Phi_n \rangle \) be such a configuration.

Since **Abstract** and **Arrange** cannot be applied, \( \Gamma \) is empty.
Since **Contradict** does not apply, \( \Delta \land \Phi_i \) is \( T_i \)-satisfiable. If \( \Delta(V) \) is an arrangement then Tinelli-Harandi’s Theorem finishes the proof. Otherwise, let \( \Delta' = \Delta \cup \{x_1 \neq y_1, \ldots, x_k \neq y_k\} \) the maximal satisfiable extension of \( \Delta \) such that \( \Delta \not\models x_i \neq y_i \). \( \Delta'(V) \) is an arrangement and \( \Delta' \models \Delta \).

If \( \Phi_i \land \Delta' \) is not \( T_i \)-satisfiable then \( \Phi_i \models_{T_i} \Delta^+ \rightarrow \neg \Delta^- \lor \delta \) where \( \delta \) is the clause \( x_1 = y_1 \lor \cdots \lor x_k = y_k \) and \( \Delta^+ \) (resp. \( \Delta^- \)) (dis)equations of \( \Delta \). Since \( T_i \) is convex, \( \Phi_i \models_{T_i} \Delta^+ \rightarrow x = y \) where \( x = y \in \neg \Delta^- \lor \delta \). Since **Deduct** does not apply, we have \( \Delta \models x = y \) and (since \( \Delta \models \Delta^- \)) \( \Delta \models \delta \), which contradict the satisfiability of \( \Delta' \)
Correctness (VII): Final Proof

Lemma [Equisatisfiability]

If \( C \Rightarrow C' \) then \( C \) and \( C' \) are equisatisfiables

Theorem [Correctness]

A set of literals \( \Gamma \) is satisfiable iff there exists an irreducible and proper configuration \( C \) such that \( \langle V \mid \emptyset \mid \Gamma \mid \emptyset \rangle \Rightarrow^* C \).

Proof.

It suffices to prove that a configuration \( C \) is satisfiable if and only if there exists a proper irreducible configuration \( C' \) such that \( C \Rightarrow^* C' \).

By induction over the terminating relation \( \Rightarrow \).

If \( C \) is irreducible, we conclude by the Lemma on irreducibility. If \( C \) reduces to \( C' \) then \( C' \) is equisatisfiable and we conclude by the induction hypothesis on \( C' \).
We first adapt **Deduction** for dealing with **disjunctions** of equalities

\[
\text{Deduction} \quad \frac{\langle V \mid \Delta \mid \Gamma \mid \Phi_0, \ldots, \Phi_n \rangle}{\langle V \mid \Delta \cup \delta \mid \Gamma \mid \Phi_0, \ldots, \Phi_n \rangle}
\]

if \( \Phi_i, \Delta \models \tau_i \delta \) and \( \Delta \not\models \delta \)

where \( \delta \) is a disjunction of equalities between variables
We also add a branching rule

\[ \text{Branch} \quad \langle V \mid \Delta \uplus \{x_1 = y_1 \lor \cdots \lor x_k = y_k\} \mid \Gamma \mid \Phi_0, \ldots, \Phi_n \rangle \]

\[ \langle V \mid \Delta \uplus \{x_i = y_i\} \mid \Gamma \mid \Phi_0, \ldots, \Phi_n \rangle \]

if \( \Delta \not \models x_i = y_i \quad (1 \leq i \leq k) \)

The correctness proof requires only a small number of modifications
Case splits for non-convex theories can be lifted to the SAT solver

When \( \text{Mode} = \text{search} \)

\[
M \models_T \bigvee x_i = y_i \quad \text{\(x_i\) and \(y_i\) are shared variables}
\]

\[
F := F \cup \{ \bigvee x_i = y_i \}
\]

\[\text{T-Learn} \]
Delayed Theory Propagation

- Nondeterministic NO
- Create a set of interface equalities $x = y$ between shared variables
- Use SAT solver to guess the partition

Main disadvantages

- The number of additional equality literals is quadratic in the number of shared variables
- Extension to quantified formulas
Model-Based Theory Combination

Instead of propagating disjunctions when

$$\Gamma \models \mathcal{T} \ u_1 = v_1 \lor \ldots u_n = v_n$$

Use a candidate model $\mathcal{M}$ of $\mathcal{T}$ that satisfies $\Gamma$ and propagate all equalities implied by $\mathcal{M}$

if $\mathcal{M} \models \mathcal{T} \land \Gamma \land u_i = v_i$ then propagate $u_i = v_i$

If other theories do not agree with that choice, then backtrack to fix the model

- In practice, the number of inter-theory equalities that matter is small, but intra-theory equalities does matter
- Backtracking is usually cheap
- Limit the number of equalities implied by $\mathcal{M}$
Efficient Deduction of Implied Equalities

In rule `Deduct`, when the theory is convex, we have to find a pair of new variables \( (x, y) \) such that

\[
\Delta, \Phi_i \models_{T_i} x = y
\]

Generic Solution:

For every pair of variables \( (x, y) \), use the decision procedure of \( T_i \) to check whether \( \Delta \cup \Phi_i \cup \{x \neq y\} \) is \( T_i \)-satisfiable

This solution may be very expensive; for some convex theories there exists efficient algorithms for computing new implied equalities.

For that, the decision procedures maintain a **union-find** data structure on terms such that a new equality \( x = y \) can be efficiently deduced by checking that \( \text{find}(x) = \text{find}(y) \) is true.
State Normalization

To maintain its union-find data structure the decision procedure uses a normalization function (which is theory dependent). A normalization step is represented by the relation

\[(\Delta, \Phi_i) \gg (\Delta, \Phi'_i)\]

Intuitively, \(\Phi_i\) can be simplified, with the use of \(\Delta\), into a “more” normalized equivalent set \(\Phi'_i\).

Three conditions are necessary

1. The relation \(\gg\) must be **terminating**
State Normalization

To maintain its *union-find* data structure the decision procedure uses a normalization function (which is theory dependent). A normalization step is represented by the relation

\[(\Delta, \Phi_i) \gg (\Delta, \Phi'_i)\]

Intuitively, \(\Phi_i\) can be simplified, with the use of \(\Delta\), into a “more” normalized equivalent set \(\Phi'_i\).

Three conditions are necessary

1. The relation \(\gg\) must be **terminating**
2. \(\Phi_i \land \Delta\) and \(\Phi'_i \land \Delta\) must be equisatisfiable
State Normalization

To maintain its *union-find* data structure the decision procedure uses a *normalization function* (which is theory dependent). A normalization step is represented by the relation

\[(\Delta, \Phi_i) \gg (\Delta, \Phi'_i)\]

Intuitively, \(\Phi_i\) can be simplified, with the use of \(\Delta\), into a “more” normalized equivalent set \(\Phi'_i\).

Three conditions are necessary

1. The relation \(\gg\) must be **terminating**
2. \(\Phi_i \land \Delta\) and \(\Phi'_i \land \Delta\) must be equisatisfiable
3. Completeness of \(\gg\):
   
   if \(\Phi_i, \Delta \models_{T_i} x = y\) and \(\Delta \not\models x = y\) then there exists \(\Phi'_i\) such that \((\Phi_i, \Delta) \gg^* (\Phi'_i, \Delta)\) and \(\{x = t, y = t\} \subseteq \Phi'_i\)
State Normalization

To maintain its *union-find* data structure the decision procedure uses a normalization function (which is theory dependent). A normalization step is represented by the relation

\[(\Delta, \Phi_i) \gg (\Delta, \Phi'_i)\]

Intuitively, \(\Phi_i\) can be simplified, with the use of \(\Delta\), into a “more” normalized equivalent set \(\Phi'_i\).

Three conditions are necessary

1. The relation \(\gg\) must be terminating
2. \(\Phi_i \land \Delta\) and \(\Phi'_i \land \Delta\) must be equisatisfiable
3. Completeness of \(\gg\):

   if \(\Phi_i, \Delta \models_{\mathcal{T}_i} x = y\) and \(\Delta \not\models x = y\) then there exists \(\Phi'_i\) such that \((\Phi_i, \Delta) \gg^\ast (\Phi'_i, \Delta)\) and \(\{x = t, y = t\} \subseteq \Phi'_i\)

\[
\begin{align*}
\text{Norm} & \quad \frac{\langle V \mid \Delta \mid \Gamma \mid \ldots, \Phi_i, \ldots \rangle}{\langle V \mid \Delta \mid \Gamma \mid \ldots, \Phi'_i, \ldots \rangle} \quad \text{if } (\Delta, \Phi) \gg (\Delta, \Phi')
\end{align*}
\]
Efficient Deduction Rule

We implement $\Delta$ as a *union-find* data structure and we write $\Delta(x)$ the representative of the variable $x$.

Using state normalization, new equalities between variables can be directly extracted from the union-find data structure.

$$\text{TDeduct} \frac{\langle V \mid \Delta \mid \Gamma \mid \ldots, \Phi_i \cup \{x = a, y = a\}, \ldots \rangle}{\langle V \mid \Delta \cup \{x = y\} \mid \Gamma \mid \ldots, \Phi_i \cup \{x = a, y = a\}, \ldots \rangle}$$

if $\Delta(x) \neq \Delta(y)$

NO is correct if *Deduct* is replaces by *Norm* and *TDeduct*.
A Shostak theory $\mathcal{T}$ is a convex theory for which there exists a canonizer and a solver.

A canonizer $\sigma$ is a function that for every term $u$ returns a unique representative $\sigma(u)$ in the equivalence class of $u$. A canonizer must satisfy the following conditions:

1. $\mathcal{T} \models u = v$ iff $\sigma(u) = \sigma(v)$
2. $\sigma(\sigma(u)) = \sigma(u)$
3. if $x$ occurs in $\sigma(u)$ then $x$ occurs in $u$
4. if $\sigma(u) = u$ then $\sigma(v) = v$ for every sub-terms $v$ of $u$
A general solution of a satisfiable equation \( u = v \) is a set of equations of the triangular form

\[
x_1 = t_1, \ldots, x_k = t_k
\]

where \( x_i \) are variables occurring in \( u \) and \( v \) but not in \( t_i \), such that

\[
\models \tau \; u = v \iff (\exists y_1 \ldots y_m) \; (x_1 = t_1 \land \cdots \land x_k = t_k)
\]

where \( y_1, \ldots, y_m \) are the \( t_i \)'s variables
A solver for a theory $\mathcal{T}$ is an algorithm that takes a $\mathcal{T}$-equation $u = v$ as input, and returns unsat if this equation is not $\mathcal{T}$-satisfiable, and its general solution if it is $\mathcal{T}$-satisfiable.

Examples of theories equipped with solvers found in practice: linear arithmetic over the rationals, theory of records.
We use the canonizer and the solvers of a Shostak theory to bring $\Phi$ into a triangular form $\{x_1 = t_1, \ldots, x_k = t_k\}$.
Normalization for Shostak Theories (II)

We use the canonizer and the solvers of a Shostak theory to bring $\Phi$ into a triangular form \( \{ x_1 = t_1, \ldots, x_k = t_k \} \)

**Example:**

1. Assume $\Phi$ and $\Delta$ are of the form
   \[
   \Phi = \{ x_1 = u - v, x_2 = 2v - u, x_3 = 2u - v, x_4 = 2v \} \\
   \Delta = \{ x_1 = x_2 \}
   \]

2. Solve $x_1 = x_2$, that is $u - v = 2v - u
   \[
   u - v = 2v - u \quad \Rightarrow \quad \{ u = 3t, v = 2t \}
   \]

3. Substitute $u$ and $v$ in $\Phi$, and canonize the right parts
   \[
   \Phi' = \{ x_1 = t, x_2 = t, x_3 = 4t, x_4 = 4t \}
   \]

4. Apply TDeduct to infer $x_3 = x_4$
Rule **Norm** for a Shostak theory $\mathcal{T}_i$ is implemented by a combination of the following rules

**Canon**

\[
\begin{align*}
\langle V \mid \Delta \mid \Gamma \mid \ldots, \Phi_i \cup \{x = a\}, \ldots \rangle & \\
\langle V \mid \Delta \mid \Gamma \mid \ldots, \Phi_i \cup \{x = \text{canon}_i(a)\}, \ldots \rangle \\
\text{if } a \neq \text{canon}_i(a)
\end{align*}
\]

**Solve**

\[
\begin{align*}
\langle V \mid \Gamma \mid \Delta \mid \ldots, \Phi_i \cup \{x = a, y = b\}, \ldots \rangle & \\
\langle V \mid \Gamma \mid \Delta \mid \ldots, (\Phi_i \cup \{x = a, y = b\} \cup \text{solve}(a = b))^2, \ldots \rangle \\
\text{if } \Delta(x) = \Delta(y) \text{ and } a \neq b \text{ and } a = b \text{ is } \mathcal{T}_i\text{-satisfiable}
\end{align*}
\]
QUANTIFIERS
Consider the following axiomatization (in Alt-Ergo’s syntax) for an ordering relation \( \texttt{le} \)

```plaintext
logic \texttt{le: int,int \to prop}
axiom \texttt{refl: forall x:int. le(x,x)}
axiom \texttt{trans:}
  forall x,y,z:int. le(x,y) and le(y,z) \to le(x,z)
axiom \texttt{antisym:}
  forall x,y:int. le(x,y) and le(y,x) \to x = y
```
Consider the following axiomatization (in Alt-Ergo’s syntax) for an ordering relation \texttt{le}

\begin{verbatim}
logic le:int,int -> prop
axiom refl: forall x:int. le(x,x)
axiom trans:
    forall x,y,z:int. le(x,y) and le(y,z) -> le(x,z)
axiom antisym:
    forall x,y:int. le(x,y) and le(y,x) -> x = y
\end{verbatim}

and some goals we want to prove:

\begin{verbatim}
goal g1: le(2,5) and le(5,10) -> le(2,10)
goal g2:
    forall a:int.
        le(a,5) and le(5,8) and le(8,a) -> a=5
\end{verbatim}
Many SMT solvers handle universal formulas through an instantiation mechanism.
Guiding Quantifier Instantiation

Many SMT solvers handle universal formulas through an instantiation mechanism

Questions:

- How to find good instances to prove a goal?
- How to limit the (prohibitive) number of instances?
Many SMT solvers handle universal formulas through an instantiation mechanism

Questions:

▶ How to find good instances to prove a goal?
▶ How to limit the (prohibitive) number of instances?

A possible answer: find good heuristics!
Guiding Quantifier Instantiation

Many SMT solvers handle universal formulas through an instantiation mechanism

Questions:

▶ How to find good instances to prove a goal?
▶ How to limit the (prohibitive) number of instances?

A possible answer: find good heuristics!

▶ In practice, heuristics for choosing new instances are based on triggers: lists of patterns (terms with variables) that guide (or restrict) instantiations to known ground terms that have a given form
If $P(x)$ is used as trigger in the following axiom $ax1$

logic $P,Q,R$: int -> prop
axiom $ax1$: forall $x$:int. $(P(x) \text{ or } Q(x)) \rightarrow R(x)$
goal $g3$: $P(1) \rightarrow R(1)$
goal $g4$: $Q(2) \rightarrow R(2)$

then, among the set of known terms $\{P(1), R(1), P(2), R(2)\}$, only $P(1)$ can be used to create the following instance of $ax1$

$$((P(1) \text{ or } Q(1)) \rightarrow R(1))$$

which implies that only goal $g3$ is proved
Explicit Triggers

SMT solvers’ input syntax provides the possibility for a user to specify its own triggers.

For instance, in Alt-Ergo, the list of terms \([f(x), \, Q(y)]\) is an explicit trigger for the following axiom \(\text{ax2}\):

\[
\text{logic } P, Q, R: \text{int} \rightarrow \text{prop} \\
\text{logic } f: \text{int} \rightarrow \text{int} \\
\text{axiom } \text{ax2}: \\
\quad \forall x, y:\text{int} \quad [f(x), \, Q(y)]. \\
\quad P(f(x)) \ \text{and} \ Q(y) \rightarrow R(x)
\]
We use a matching algorithm to create new instances of universal formulas.

Given a ground term $t$ and a pattern $p$, the matching algorithm returns a set $S$ of substitutions over the variables of $p$ such that

$$t = \sigma(p) \quad \text{for all} \quad \sigma \in S$$
Limitation of Matching

Purely syntactic matching is very limited!

Consider for instance the following formulas:

```
logic P,R : int -> prop
logic f : int -> int
axiom ax : forall x:int [P(f(x))]. P(f(x)) -> R(x)
goal g1 : forall a:int. P(a) -> a = f(2) -> R(2)
```

The trigger $P(f(x))$ prevents the creation of instances of axiom $ax$ since there is no ground term of the form $P(f(\_))$ in the problem.

To prove such goals, we need to extend the matching algorithm to find substitutions modulo (ground) equalities.
E-Matching

Given a set of ground equations $E$, a ground term $t$ and a pattern $p$, the e-matching algorithm returns a set $S$ of substitutions over the variables of $p$ such that

$$E \models t = \sigma(p) \text{ for all } \sigma \in S$$

In the previous example

logic $P,R : \text{int} \to \text{prop}$
logic $f : \text{int} \to \text{int}$
axiom $ax : \forall x: \text{int} \ [P(f(x))]. \ P(f(x)) \to R(x)$
goal $g1 : \forall a: \text{int}. \ P(a) \to a = f(2) \to R(2)$

e-matching takes advantage of ground equality $a = f(2)$ and returns the substitution $\sigma = \{x \mapsto 2\}$ which is used to create the instance $P(f(2)) \to R(2)$ of axiom $ax$
Ground Terms

Known ground terms are extracted from literals assumed or implied by the SAT solver.

Instantiation based mechanisms are strongly impacted by the number and the relevance of known ground terms:

- more ground terms, more instances of lemmas
- irrelevant ground terms, irrelevant instances
Ground Terms and Linear CNF

The shape of formulas to be proved, and in particular the conversion process used to produce a CNF, has a strong impact on the number of known ground terms.

Consider for instance the following formula

\[ A \lor (B \land C) \]

When \( A \) is assumed to be true, terms of \( A \) become known and the rest of the (terms of the) formula \( (B \land C) \) can be ignored.
Ground Terms and Linear CNF

The shape of formulas to be proved, and in particular the conversion process used to produce a CNF, has a strong impact on the number of known ground terms.

Consider for instance the following formula

\[ A \lor (B \land C) \]

When \( A \) is assumed to be true, terms of \( A \) become known and the rest of the (terms of the) formula \((B \land C)\) can be ignored.

However, because of the shape of the CNF conversion

\[ (A \lor X) \land (X \iff (B \land C)) \]

the SMT solver will assign a value to \( X \) (even when \( A \) is true) and terms from \( B \) and \( C \) will be considered has known terms.
EXTRA MATERIAL
A signature $\Sigma$ is a finite set of function and predicate symbols with an arity

- **Constants** are just function symbols of arity 0

- We assume that $\Sigma$ contains the binary predicate $=\!

- We assume a set $\mathcal{V}$ of variables, distinct from $\Sigma$

$T(\Sigma, \mathcal{V})$ is the set of terms, *i.e.* the smallest set which contains $\mathcal{V}$ and such that $f(t_1, \ldots, t_n) \in T(\Sigma, \mathcal{V})$ whenever $t_1, \ldots, t_n \in T(\Sigma, \mathcal{V})$ and $f \in \Sigma$

$T(\Sigma, \emptyset)$ is the set of ground terms

Terms are just trees. Given a term $t$ and a position $\pi$ in a tree, we write $t_\pi$ for the sub-term of $t$ at position $\pi$. We also write $t[\pi \mapsto t']$ for the replacement of the sub-term of $t$ at position $\pi$ by the term $t'$
First-Order Logic: Formulas

- An atomic formula is $P(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms in $T(\Sigma, \mathcal{V})$ and $P$ is a predicate symbol of $\Sigma$

- Literals are atomic formulas or their negation

- Formulas are inductively constructed from atomic formulas with the help of Boolean connectives and quantifiers $\forall$ and $\exists$

- Ground formulas contain only ground terms

- A variable is free if it is not bound by a quantifier

- A sentence is a formula with no free variables
A model $\mathcal{M}$ for a signature $\Sigma$ is defined by

- a domain $\mathcal{D}_M$
- an interpretation $f^\mathcal{M}$ for each function symbol $f \in \Sigma$
- a subset $P^\mathcal{M}$ of $\mathcal{D}_M^n$ for each predicate $P \in \Sigma$ of arity $n$
- an assignment $\mathcal{M}(x)$ for each variable $x \in \mathcal{V}$

The cardinality of model $\mathcal{M}$ is the the cardinality of $\mathcal{D}_M$
First-Order Logic : Semantics

Interpretation of terms:

\[ M[x] = M(x) \]
\[ M[f(t_1, \ldots, t_n)] = f^M(M[t_1], \ldots, M[t_n]) \]

Interpretation of formulas:

\[ M \models t_1 = t_2 \iff M[t_1] = M[t_2] \]
\[ M \models P(t_1, \ldots, t_n) \iff (M[t_1], \ldots, M[t_n]) \in P^M \]
\[ M \models \neg F \iff M \not\models F \]
\[ M \models F_1 \land F_2 \iff M \models F_1 \text{ and } M \models F_2 \]
\[ M \models F_1 \lor F_2 \iff M \models F_1 \text{ or } M \models F_2 \]
\[ M \models \forall x.F \iff M\{x \mapsto v\} \models F \text{ for all } v \in D_M \]
\[ M \models \exists x.F \iff M\{x \mapsto v\} \models F \text{ for some } v \in D_M \]
A formula $F$ is **satisfiable** if there a model $\mathcal{M}$ such that $\mathcal{M} \models F$, otherwise $F$ is **unsatisfiable**.

A formula $F$ is **valid** if $\neg F$ is unsatisfiable.
A first-order theory $T$ over a signature $\Sigma$ is a set of sentences.

A theory is consistent if it has (at least) a model.

A formula $F$ is satisfiable in $T$ (or $T$-satisfiable) if there exists a model $\mathcal{M}$ for $T \land F$, written $\mathcal{M} \models_T F$.

A formula $F$ is $T$-validity, denoted $\models_T F$, if $\neg F$ is $T$-unsatisfiable.
A decision procedure is an algorithm used to determine whether a formula $F$ in a theory $T$ is satisfiable.

Many decision procedures work on conjunctions of (ground) literals.