Decision Procedures for Verification

Combinations of Decision Procedures (1)

24.01.2022

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de
Until now:

Decision Procedures

- Uninterpreted functions
  - congruence closure
- Numerical domains
  - difference logic

$LI(\mathbb{R})$ and $LI(\mathbb{Q})$

Method of Fourier-Motzkin

Method of Weisspfenning-Loos
3.5. Combinations of theories

The combined validity problem

For $i = 1, 2$  
• let $\mathcal{T}_i$ be a first-order theory in signature $\Sigma_i$  
• let $\mathcal{L}_i$ be a class of (closed) $\Sigma$-formulae

Let $\mathcal{T}_1 \bigoplus \mathcal{T}_2$ be a combination of $\mathcal{T}_1$ and $\mathcal{T}_2$  
Let $\mathcal{L}_1 \bigoplus \mathcal{L}_2$ be a combination of $\mathcal{L}_1$ and $\mathcal{L}_2$

Problem: Given $\phi$ in $\mathcal{L}_1 \bigoplus \mathcal{L}_2$, is it the case that $\mathcal{T}_1 \bigoplus \mathcal{T}_2 \models \phi$?
The combined decidability problem I

For $i = 1, 2$  
• let $\mathcal{T}_i$ be a first-order theory in signature $\Sigma_i$  
• let $\mathcal{L}_i$ be a class of (closed) $\Sigma$-formulae  
• assume the $\mathcal{T}_i$-validity problem for $\mathcal{L}_i$ is decidable

Let $\mathcal{T}_1 \oplus \mathcal{T}_2$ be a combination of $\mathcal{T}_1$ and $\mathcal{T}_2$  
Let $\mathcal{L}_1 \oplus \mathcal{L}_2$ be a combination of $\mathcal{L}_1$ and $\mathcal{L}_2$  

**Question:** Is the $\mathcal{T}_1 \oplus \mathcal{T}_2$-validity problem for $\mathcal{L}_1 \oplus \mathcal{L}_2$ decidable?
The combined decidability problem II

For $i = 1, 2$  
- let $T_i$ be a first-order theory in signature $\Sigma_i$  
- let $L_i$ be a class of (closed) $\Sigma$-formulae  
- $P_i$ decision procedure for $T_i$-validity for $L_i$

Let $T_1 \oplus T_2$ be a combination of $T_1$ and $T_2$  
Let $L_1 \oplus L_2$ be a combination of $L_1$ and $L_2$

**Question:** Can we combine $P_1$ and $P_2$ modularly into a decision procedure for the $T_1 \oplus T_2$-validity problem for $L_1 \oplus L_2$?

**Main issue:** How are $T_1 \oplus T_2$ and $L_1 \oplus L_2$ defined?
Combinations of theories and models

Forgetting symbols

Let $\Sigma = (\Omega, \Pi)$ and $\Sigma' = (\Omega', \Pi')$ s.t. $\Sigma \subseteq \Sigma'$, i.e., $\Omega \subseteq \Omega'$ and $\Pi \subseteq \Pi'$

For $\mathcal{A} \in \Sigma'$-alg, we denote by $\mathcal{A}|_{\Sigma}$ the $\Sigma$-structure for which:

$$U_{\mathcal{A}|_{\Sigma}} = U_{\mathcal{A}}, \quad f_{\mathcal{A}|_{\Sigma}} = f_{\mathcal{A}} \quad \text{for } f \in \Omega;$$

$$P_{\mathcal{A}|_{\Sigma}} = P_{\mathcal{A}} \quad \text{for } P \in \Pi$$

(ignore functions and predicates associated with symbols in $\Sigma' \setminus \Sigma$)

$\mathcal{A}|_{\Sigma}$ is called the restriction (or the reduct) of $\mathcal{A}$ to $\Sigma$.

Example: $\Sigma' = (\{+/2, \ast/2, 1/0\}, \{\leq /2, \text{even}/1, \text{odd}/1\})$

$\Sigma = (\{+/2, 1/0\}, \{\leq /2\}) \subseteq \Sigma'$

$\mathcal{N} = (\mathbb{N}, +, \ast, 1, \leq, \text{even}, \text{odd}) \quad \mathcal{N}|_{\Sigma} = (\mathbb{N}, +, 1, \leq)$
One possibility of combining theories

Syntactic view: \( \mathcal{T}_1 + \mathcal{T}_2 = \mathcal{T}_1 \cup \mathcal{T}_2 \subseteq F_{\Sigma_1 \cup \Sigma_2}(X) \)

\[
\text{Mod}(\mathcal{T}_1 \cup \mathcal{T}_2) = \{ \mathcal{A} \in (\Sigma_1 \cup \Sigma_2)\text{-alg} \mid \mathcal{A} \models G, \text{ for all } G \text{ in } \mathcal{T}_1 \cup \mathcal{T}_2 \}
\]

where \( \Sigma_1 \cup \Sigma_2 = (\Omega_1, \Pi_1) \cup (\Omega_2, \Pi_2) = (\Omega_1 \cup \Omega_2, \Pi_1 \cup \Pi_2) \)
One possibility of combining theories

**Syntactic view:** \( \mathcal{T}_1 + \mathcal{T}_2 = \mathcal{T}_1 \cup \mathcal{T}_2 \subseteq F_{\Sigma_1 \cup \Sigma_2}(X) \)

\[
\text{Mod}(\mathcal{T}_1 \cup \mathcal{T}_2) = \{ A \in (\Sigma_1 \cup \Sigma_2)\text{-alg} \mid A \models G, \text{ for all } G \text{ in } \mathcal{T}_1 \cup \mathcal{T}_2 \}
\]

**Semantic view:** Let \( \mathcal{M}_i = \text{Mod}(\mathcal{T}_i), i = 1, 2 \)

\[
\mathcal{M}_1 + \mathcal{M}_2 = \{ A \in (\Sigma_1 \cup \Sigma_2)\text{-alg} \mid A|_{\Sigma_i} \in \mathcal{M}_i \text{ for } i = 1, 2 \}
\]
One possibility of combining theories

**Syntactic view:**  \( \mathcal{T}_1 + \mathcal{T}_2 = \mathcal{T}_1 \cup \mathcal{T}_2 \subseteq F_{\Sigma_1 \cup \Sigma_2}(X) \)

\[ \text{Mod}(\mathcal{T}_1 \cup \mathcal{T}_2) = \{ A \in (\Sigma_1 \cup \Sigma_2)\text{-alg} \mid A \models G, \text{ for all } G \in \mathcal{T}_1 \cup \mathcal{T}_2 \} \]

**Semantic view:** Let \( \mathcal{M}_i = \text{Mod}(\mathcal{T}_i), i = 1, 2 \)

\[ \mathcal{M}_1 + \mathcal{M}_2 = \{ A \in (\Sigma_1 \cup \Sigma_2)\text{-alg} \mid A|_{\Sigma_i} \in \mathcal{M}_i \text{ for } i = 1, 2 \} \]

\[ A \in \text{Mod}(\mathcal{T}_1 \cup \mathcal{T}_2) \text{ iff } A \models G, \text{ for all } G \in \mathcal{T}_1 \cup \mathcal{T}_2 \]

\[ \text{iff } A|_{\Sigma_i} \models G, \text{ for all } G \in \mathcal{T}_i, i = 1, 2 \]

\[ \text{iff } A|_{\Sigma_i} \in \mathcal{M}_i, i = 1, 2 \]

\[ \text{iff } A \in \mathcal{M}_1 + \mathcal{M}_2 \]
One possibility of combining theories

**Syntactic view:** \( \mathcal{T}_1 + \mathcal{T}_2 = \mathcal{T}_1 \cup \mathcal{T}_2 \subseteq F_{\Sigma_1 \cup \Sigma_2}(X) \)

\[ \text{Mod}(\mathcal{T}_1 \cup \mathcal{T}_2) = \{ \mathcal{A} \in (\Sigma_1 \cup \Sigma_2)\text{-alg} \mid \mathcal{A} \models G, \text{ for all } G \text{ in } \mathcal{T}_1 \cup \mathcal{T}_2 \} \]

**Semantic view:** Let \( \mathcal{M}_i = \text{Mod}(\mathcal{T}_i), i = 1, 2 \)

\[ \mathcal{M}_1 + \mathcal{M}_2 = \{ \mathcal{A} \in (\Sigma_1 \cup \Sigma_2)\text{-alg} \mid \mathcal{A}_{|\Sigma_i} \in \mathcal{M}_i \text{ for } i = 1, 2 \} \]

**Remark:** \( \mathcal{A} \in \text{Mod}(\mathcal{T}_1 \cup \mathcal{T}_2) \) iff \( \mathcal{A}_{|\Sigma_1} \in \text{Mod}(\mathcal{T}_1) \) and \( \mathcal{A}_{|\Sigma_2} \in \text{Mod}(\mathcal{T}_2) \)

**Consequence:** \( \text{Th}(\text{Mod}(\mathcal{T}_1 \cup \mathcal{T}_2)) = \text{Th}(\mathcal{M}_1 + \mathcal{M}_2) \)
Example

1. **Presburger arithmetic + UIF**

\[ \text{Th}(\mathbb{Z}_+) \cup \text{UIF} \quad \Sigma = (\Omega, \Pi) \]

Models: \((A, 0, s, +, \{ f_A \}_{f \in \Omega}, \leq, \{ P_A \}_{P \in \Pi})\)

where \((A, 0, s, +, \leq) \in \text{Mod}(\text{Th}(\mathbb{Z}_+)).\)

2. **The theory of reals + the theory of a monotone function \(f\)**

\[ \text{Th}(\mathbb{R}) \cup \text{Mon}(f) \quad \text{Mon}(f) : \forall x, y (x \leq y \rightarrow f(x) \leq f(y)) \]

Models: \((A, +, \ast, f_A, \{\leq\}), \text{where} \)

where \((A, +, \ast, \leq) \in \text{Mod}(\text{Th}(\mathbb{R})).\)

\((A, f_A, \leq) \models \text{Mon}(f), \text{i.e. } f_A : A \rightarrow A \text{ monotone.}\)

**Note:** The signatures of the two theories share the \(\leq\) predicate symbol
Combinations of theories

**Definition.** A theory is consistent if it has at least one model.

**Question:** Is the union of two consistent theories always consistent?

**Answer:** No. (Not even when the two theories have disjoint signatures)

**Example:**

\[ \Sigma_1 = (\Omega_1, \emptyset), \quad \Sigma_2 = (\{c/0, d/0\}, \emptyset), \quad c, d \notin \Omega_1 \]

\[ \mathcal{T}_1 = \{\exists x, y, z (x \not\approx y \land x \not\approx z \land y \not\approx z)\} \]

\[ \mathcal{T}_2 = \{\forall x (x \approx c \lor x \approx d)\} \]

\[ A \in \text{Mod}(\mathcal{T}_1) \iff |U_A| \geq 3. \]

\[ B \in \text{Mod}(\mathcal{T}_2) \iff |U_B| \leq 2. \]
Combinations of theories

The combined decidability problem

For $i = 1, 2$  
• let $\mathcal{T}_i$ be a first-order theory in signature $\Sigma_i$  
• assume the $\mathcal{T}_i$ ground satisfiability problem is decidable

Let $\mathcal{T}_1 \oplus \mathcal{T}_2$ be a combination of $\mathcal{T}_1$ and $\mathcal{T}_2$

**Question:**
Is the $\mathcal{T}_1 \oplus \mathcal{T}_2$ ground satisfiability problem decidable?
Goal: Modularity

Modular Reasoning

Example:

\( \mathcal{T}_0: \Sigma_0\)-theory.

\( \mathcal{T}_i: \Sigma_i\)-theory; \( \mathcal{T}_0 \subseteq \mathcal{T}_i \) \( \Sigma_0 \subseteq \Sigma_i \).

Can use provers for \( \mathcal{T}_1, \mathcal{T}_2 \) as blackboxes to prove theorems in \( \mathcal{T}_1 \cup \mathcal{T}_2 \)?

Which information needs to be exchanged between the provers?
Combinations of theories

For $i = 1, 2$  
• let $\mathcal{T}_i$ be a first-order theory in signature $\Sigma_i$
• s.t. the ground satisfiability problem for $\mathcal{T}_i$ is decidable

**Question:** Is the ground satisfiability problem for $\mathcal{T}_1 \cup \mathcal{T}_2$ decidable?
**Combinations of theories**

For $i = 1, 2$  
- let $T_i$ be a first-order theory in signature $\Sigma_i$  
- s.t. the ground satisfiability problem for $T_i$ is decidable

**Question:** Is the ground satisfiability problem for $T_1 \cup T_2$ decidable?

**In general:** No (restrictions needed for affirmative answer)

**Example.** Word problem for $T$: Decide if $T \models \forall x (s \approx t)$

- $A$: theory of associativity
- $G$: finite set of ground equations
  (presentation for semigroup with undecidable word problem)
  (There exists finitely-presented semigroup with undecidable word problem [Matijasevic’67])

**Word problem:** decidable for $A, G$; undecidable for $A \cup G$
Combinations of theories

For $i = 1, 2$  
• let $\mathcal{T}_i$ be a first-order theory in signature $\Sigma_i$  
• s.t. the ground satisfiability problem for $\mathcal{T}_i$ is decidable

**Question:** Is the ground satisfiability problem for $\mathcal{T}_1 \cup \mathcal{T}_2$ decidable?

**In general:** No (restrictions needed for affirmative answer)

**Example.** Word problem for $\mathcal{T}$: Decide if $\mathcal{T} \models \forall x (s \approx t)$

**Simpler instances:** combinations of theories over disjoint signatures, theories sharing constructors, compatibility with shared theory ...
Combinations of theories

For \( i = 1, 2 \)
\[ \bullet \text{let } \mathcal{T}_i \text{ be a first-order theory in signature } \Sigma_i \]
\[ \bullet \text{s.t. the ground satisfiability problem for } \mathcal{T}_i \text{ is decidable} \]

**Question:** Is the ground satisfiability problem for \( \mathcal{T}_1 \cup \mathcal{T}_2 \) decidable?

**In general:** No (restrictions needed for affirmative answer)

**Theorem [Bonacina, Ghilardi et.al, IJCAR 2006]**
There are theories \( \mathcal{T}_1, \mathcal{T}_2 \) with disjoint signatures and decidable ground satisfiability problem such that ground satisfiability in \( \mathcal{T}_1 \cup \mathcal{T}_2 \) is undecidable.

**Idea:** Construct \( \mathcal{T}_1 \) such that ground satisfiability is decidable, but it is undecidable whether a constraint \( \Gamma_1 \) is satisfiable in an infinite model of \( \mathcal{T}_1 \). (Construction uses Turing Machines). Let \( \mathcal{T}_2 \) having only infinite models.
Combination of theories over disjoint signatures

The Nelson/Oppen procedure

**Given:** \( T_1, T_2 \) first-order theories with signatures \( \Sigma_1, \Sigma_2 \)

Assume that \( \Sigma_1 \cap \Sigma_2 = \emptyset \) (share only \( \approx \))

\( P_i \) decision procedures for satisfiability of ground formulae w.r.t. \( T_i \)

\( \phi \) quantifier-free formula over \( \Sigma_1 \cup \Sigma_2 \)

**Task:** Check whether \( \phi \) is satisfiable w.r.t. \( T_1 \cup T_2 \)

**Note:** Restrict to conjunctive quantifier-free formulae

\( \phi \mapsto DNF(\phi) \)

\( DNF(\phi) \) satisfiable in \( T \) iff one of the disjuncts satisfiable in \( T \)
Example

[Nelson & Oppen, 1979]

**Theories**

<table>
<thead>
<tr>
<th>Theory</th>
<th>Description</th>
<th>Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{R})</td>
<td>theory of rationals</td>
<td>(\Sigma_{\mathcal{R}} = {\leq, +, -, 0, 1})</td>
</tr>
<tr>
<td>(\mathcal{L})</td>
<td>theory of lists</td>
<td>(\Sigma_{\mathcal{L}} = {\text{car, cdr, cons}})</td>
</tr>
<tr>
<td>(\mathcal{E})</td>
<td>theory of equality (UIF)</td>
<td>(\Sigma: \text{free function and predicate symbols})</td>
</tr>
</tbody>
</table>
Example

[Nelson & Oppen, 1979]

**Theories**

<table>
<thead>
<tr>
<th>Theory</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{R} )</td>
<td>theory of rationals ( \Sigma_{\mathcal{R}} = {\leq, +, -, 0, 1} \approx )</td>
</tr>
<tr>
<td>( \mathcal{L} )</td>
<td>theory of lists ( \Sigma_{\mathcal{L}} = {\text{car, cdr, cons}} \approx )</td>
</tr>
<tr>
<td>( \mathcal{E} )</td>
<td>theory of equality (UIF) ( \Sigma: ) free function and predicate symbols ( \approx )</td>
</tr>
</tbody>
</table>

**Problems:**

1. \( \mathcal{R} \cup \mathcal{L} \cup \mathcal{E} \models \forall x, y (x \leq y \land y \leq x + \text{car} (\text{cons}(0, x)) \land P(h(x) - h(y)) \rightarrow P(0)) \)

2. Is the following conjunction:

\[
  c \leq d \land d \leq c + \text{car} (\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0)
\]

satisfiable in \( \mathcal{R} \cup \mathcal{L} \cup \mathcal{E} \)?
An Example

<table>
<thead>
<tr>
<th>Σ</th>
<th>( \Sigma )</th>
<th>( \mathcal{R} )</th>
<th>( \mathcal{L} )</th>
<th>( \mathcal{E} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axioms</td>
<td>( x + 0 \approx x )</td>
<td>{≤, +, −, 0, 1}</td>
<td>{car, cdr, cons}</td>
<td>( F \cup P )</td>
</tr>
<tr>
<td>(univ. quantif.)</td>
<td>( x - x \approx 0 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( + \text{ is } A, C )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \leq \text{ is } R, T, A )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( x \leq y \lor y \leq x )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( x \leq y \rightarrow x + z \leq y + z )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Is the following conjunction:

\[
c \leq d \land d \leq c + \text{car}(\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0)
\]

satisfiable in \( \mathcal{R} \cup \mathcal{L} \cup \mathcal{E} \)?
Step 1: Purification

Given: $\phi$ conjunctive quantifier-free formula over $\Sigma_1 \cup \Sigma_2$

Task: Find $\phi_1, \phi_2$ s.t. $\phi_i$ is a pure $\Sigma_i$-formula and $\phi_1 \land \phi_2$ equivalent with $\phi$

\[
\begin{align*}
f(s_1, \ldots, s_n) \approx g(t_1, \ldots, t_m) & \quad \mapsto \quad u \approx f(s_1, \ldots, s_n) \land u \approx g(t_1, \ldots, t_m) \\
f(s_1, \ldots, s_n) \not\approx g(t_1, \ldots, t_m) & \quad \mapsto \quad u \approx f(s_1, \ldots, s_n) \land v \approx g(t_1, \ldots, t_m) \land u \not\approx v \\
(\neg)P(\ldots, s_i, \ldots) & \quad \mapsto \quad (\neg)P(\ldots, u, \ldots) \land u \approx s_i \\
(\neg)P(\ldots, s_i[t], \ldots) & \quad \mapsto \quad (\neg)P(\ldots, s_i[t \mapsto u], \ldots) \land u \approx t \\
\text{where } t \approx f(t_1, \ldots, t_n)
\end{align*}
\]

Termination: Obvious

Correctness: $\phi_1 \land \phi_2$ and $\phi$ equisatisfiable.
Step 1: Purification

\[ c \leq d \land d \leq c + \text{car}(\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0) \]
Step 1: Purification

\[ c \leq d \land d \leq c + \text{car(cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0) \]
Step 1: Purification

\[ c \leq d \land d \leq c + \text{car} \left( \text{cons}(0, c) \right) \land P\left( h(c) - h(d) \right) \land \neg P(0) \]
Step 1: Purification

\[ c \leq d \land d \leq c + \text{car}(\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0) \]
## Step 1: Purification

\[ c \leq d \land d \leq c + \text{car}(\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>( \mathcal{R} )</th>
<th>( \mathcal{L} )</th>
<th>( \mathcal{E} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c \leq d )</td>
<td>( c_1 \approx \text{car(cons}(c_5, c)) )</td>
<td>( P(c_2) )</td>
</tr>
<tr>
<td>( d \leq c + c_1 )</td>
<td>( c_2 \approx c_3 - c_4 )</td>
<td>( \neg P(c_5) )</td>
</tr>
<tr>
<td>( c_2 \approx c_3 - c_4 )</td>
<td>( c_5 \approx 0 )</td>
<td>( c_3 \approx h(c) )</td>
</tr>
<tr>
<td>( c_5 \approx 0 )</td>
<td>( c_4 \approx h(d) )</td>
<td></td>
</tr>
</tbody>
</table>
Step 1: Purification

\[ c \leq d \land d \leq c + \text{car}(\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>( \mathcal{R} )</th>
<th>( \mathcal{L} )</th>
<th>( \mathcal{E} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c \leq d )</td>
<td>( c_1 \approx \text{car}(\text{cons}(c_5, c)) )</td>
<td>( P(c_2) )</td>
</tr>
<tr>
<td>( d \leq c + c_1 )</td>
<td></td>
<td>( \neg P(c_5) )</td>
</tr>
<tr>
<td>( c_2 \approx c_3 - c_4 )</td>
<td></td>
<td>( c_3 \approx h(c) )</td>
</tr>
<tr>
<td>( c_5 \approx 0 )</td>
<td></td>
<td>( c_4 \approx h(d) )</td>
</tr>
<tr>
<td>satisfiable</td>
<td>satisfiable</td>
<td>satisfiable</td>
</tr>
</tbody>
</table>
Step 2: Propagation

\[
c \leq d \land d \leq c + \text{car}(\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0)
\]

<table>
<thead>
<tr>
<th>(R)</th>
<th>(L)</th>
<th>(E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (c \leq d)</td>
<td>2. (d \leq c + c_1)</td>
<td>3. (c_2 \approx c_3 - c_4)</td>
</tr>
<tr>
<td></td>
<td>4. (c_5 \approx 0)</td>
<td>5. (c_3 \approx h(c))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. (c_4 \approx h(d))</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. (c_1 \approx \text{car}(\text{cons}(c_5, c)))</td>
<td></td>
<td>8. (P(c_2))</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(\neg P(c_5))</td>
</tr>
</tbody>
</table>

**deduce and propagate equalities between constants entailed by components**
Step 2: Propagation

\[ c \leq d \land d \leq c + \text{car}(\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>( R )</th>
<th>( L )</th>
<th>( E )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c \leq d )</td>
<td>( c_1 \approx \text{car}(\text{cons}(c_5, c)) )</td>
<td>( P(c_2) )</td>
</tr>
<tr>
<td>( d \leq c + c_1 )</td>
<td></td>
<td>( \neg P(c_5) )</td>
</tr>
<tr>
<td>( c_2 \approx c_3 - c_4 )</td>
<td></td>
<td>( c_3 \approx h(c) )</td>
</tr>
<tr>
<td>( c_5 \approx 0 )</td>
<td></td>
<td>( c_4 \approx h(d) )</td>
</tr>
<tr>
<td>( c_1 \approx c_5 )</td>
<td></td>
<td>( c_1 \approx c_5 )</td>
</tr>
</tbody>
</table>
Step 2: Propagation

\[ c \leq d \land d \leq c + \text{car}(\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>( \mathcal{R} )</th>
<th>( \mathcal{L} )</th>
<th>( \mathcal{E} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c \leq d )</td>
<td>( c_1 \approx \text{car}(\text{cons}(c_5, c)) )</td>
<td>( P(c_2) )</td>
</tr>
<tr>
<td>( d \leq c + c_1 )</td>
<td></td>
<td>( \neg P(c_5) )</td>
</tr>
<tr>
<td>( c_2 \approx c_3 - c_4 )</td>
<td></td>
<td>( c_3 \approx h(c) )</td>
</tr>
<tr>
<td>( c_5 \approx 0 )</td>
<td></td>
<td>( c_4 \approx h(d) )</td>
</tr>
<tr>
<td>( c_1 \approx c_5 )</td>
<td>( c_1 \approx c_5 )</td>
<td></td>
</tr>
<tr>
<td>( c \approx d )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Step 2: Propagation

\[ c \leq d \land d \leq c + \text{car}(\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>( \mathcal{R} )</th>
<th>( \mathcal{L} )</th>
<th>( \mathcal{E} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c \leq d )</td>
<td>( c_1 \approx \text{car}(\text{cons}(c_5, c)) )</td>
<td>( P(c_2) )</td>
</tr>
<tr>
<td>( d \leq c + c_1 )</td>
<td></td>
<td>( \neg P(c_5) )</td>
</tr>
<tr>
<td>( c_2 \approx c_3 - c_4 )</td>
<td></td>
<td>( c_3 \approx h(c) )</td>
</tr>
<tr>
<td>( c_5 \approx 0 )</td>
<td></td>
<td>( c_4 \approx h(d) )</td>
</tr>
<tr>
<td>( c_1 \approx c_5 )</td>
<td>( c_1 \approx c_5 )</td>
<td>( c \approx d )</td>
</tr>
<tr>
<td>( c \approx d )</td>
<td></td>
<td>( c_3 \approx c_4 )</td>
</tr>
</tbody>
</table>
## Step 2: Propagation

\[ c \leq d \land d \leq c + \text{car}(\text{cons}(0, c)) \land P(h(c) - h(d)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>(\mathcal{R})</th>
<th>(\mathcal{L})</th>
<th>(\mathcal{E})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c \leq d)</td>
<td>(c_1 \approx \text{car}(\text{cons}(c_5, c)))</td>
<td>(P(c_2))</td>
</tr>
<tr>
<td>(d \leq c + c_1)</td>
<td></td>
<td>(\neg P(c_5))</td>
</tr>
<tr>
<td>(c_2 \approx c_3 - c_4)</td>
<td></td>
<td>(c_3 \approx h(c))</td>
</tr>
<tr>
<td>(c_5 \approx 0)</td>
<td></td>
<td>(c_4 \approx h(d))</td>
</tr>
<tr>
<td>(c_1 \approx c_5)</td>
<td>(c_1 \approx c_5)</td>
<td></td>
</tr>
<tr>
<td>(c \approx d)</td>
<td></td>
<td>(c \approx d)</td>
</tr>
<tr>
<td>(c_2 \approx c_5)</td>
<td></td>
<td>(c_3 \approx c_4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(\perp)</td>
</tr>
</tbody>
</table>
The Nelson-Oppen algorithm

ϕ conjunction of literals

Step 1. Purification  \( T_1 \cup T_2 \cup \phi \mapsto (T_1 \cup \phi_1) \cup (T_2 \cup \phi_2) \):
where \( \phi_i \) is a pure \( \Sigma_i \)-formula and \( \phi_1 \land \phi_2 \) is equisatisfiable with \( \phi \).

Step 2. Propagation.
The decision procedure for ground satisfiability for \( T_1 \) and \( T_2 \) fairly exchange information concerning entailed unsatisfiability of constraints in the shared signature i.e. clauses over the shared variables.

until an inconsistency is detected or a saturation state is reached.
The Nelson-Oppen algorithm

\( \phi \) conjunction of literals

**Step 1.** Purification \( \mathcal{T}_1 \cup \mathcal{T}_2 \cup \phi \mapsto (\mathcal{T}_1 \cup \phi_1) \cup (\mathcal{T}_2 \cup \phi_2) \):

where \( \phi_i \) is a pure \( \Sigma_i \)-formula and \( \phi_1 \land \phi_2 \) is equisatisfiable with \( \phi \).

**Step 2.** Propagation.

The decision procedure for ground satisfiability for \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \) fairly exchange information concerning entailed unsatisfiability of constraints in the shared signature i.e. clauses over the shared variables.

until an inconsistency is detected or a saturation state is reached.

**not problematic; requires linear time**

**not problematic; termination guaranteed**

**Sound:** if inconsistency detected input unsatisfiable

**Complete:** under additional assumptions
Implementation

ϕ conjunction of literals

**Step 1. Purification:** \( \mathcal{T}_1 \cup \mathcal{T}_2 \cup \phi \mapsto (\mathcal{T}_1 \cup \phi_1) \cup (\mathcal{T}_2 \cup \phi_2) \),
where \( \phi_i \) is a pure \( \Sigma_i \)-formula and \( \phi_1 \land \phi_2 \) is equisatisfiable with \( \phi \).

**Step 2. Propagation:** The decision procedure for ground satisfiability for \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \) fairly exchange information concerning entailed unsatisfiability of constraints in the shared signature i.e. clauses over the shared variables.

until an inconsistency is detected or a saturation state is reached.

**How to implement Propagation?**

**Guessing:** guess a maximal set of literals containing the shared variables; check it for \( \mathcal{T}_i \cup \phi_i \) consistency.

**Backtracking:** identify disjunction of equalities between shared variables entailed by \( \mathcal{T}_i \cup \phi_i \); make case split by adding some of these equalities to \( \phi_1, \phi_2 \). Repeat as long as possible.
Implementation of propagation

Guessing variant

Guess a maximal set of literals containing the shared variables $V$
(arrangement: $\alpha(V, E) = (\bigwedge_{(u,v) \in E} u \approx v \land \bigwedge_{(u,v) \notin E} u \not\approx v)$, where $E$ equivalence relation); check it for $T_i \cup \phi_i$ consistency.

On the blackboard: Example 10.5 and 10.7 pages 272, 273
Example 10.6 and 10.9 pages 272, 275
from the book “The Calculus of Computation” by A. Bradley and Z. Manna

**Advantage:** Whenever constraints are represented as Boolean combinations of atoms, one may combine heuristics of SMT solvers with specific features of the theories to be combined to produce the right arrangement efficiently.
Implementation of propagation

Backtracking variant

Identify disjunction of equalities between shared variables entailed by $\mathcal{T}_i \cup \phi_i$; make case split by adding some of these equalities to $\phi_1, \phi_2$.

Repeat as long as possible.


Advantages:

- it works on the non-disjoint case as well

- can be made deterministic for combinations of convex theories

$\mathcal{T}$ convex iff whenever $\mathcal{T} \models \bigwedge_{i=1}^{n} A_i \rightarrow \bigvee_{j=1}^{m} B_j$

there exists $j$ s.t. $\mathcal{T} \models \bigwedge_{i=1}^{n} A_i \rightarrow B_j$
Next Time

**Termination:** only finitely many shared variables to be identified

**Soundness:** If procedure answers “unsatisfiable” then $\phi$ is unsatisfiable

**Completeness:** If procedure answers “satisfiable” then $\phi$ is satisfiable

$\iff$ For stably infinite theories (to be defined next time)