Seminar Decision Procedures and Applications

Background Information: Part I

Viorica Sofronie-Stokkermans
University Koblenz-Landau

25 June 2019
Topics for the talks

1. Matthias Becker: Decision Procedures for UTVPI Constraints
2. Delzar Habash: Automata approach to Presburger arithmetic
3. Denis Oldenburg: Quantifier elimination for linear arithmetic over the integers
4. Dominik Kohns: Reasoning about uninterpreted function symbols
5. Nico Bartmann: DPLL(T)
6. Stefan Strüder: Decision procedures for classical datatypes based on the superposition calculus
Structure

**Reasoning in standard theories**

- **Presburger arithmetic:** Delzar Habash, Denis Oldenburg
  - **Simpler fragments:** UTVPI Matthias Becker

- **Theory of uninterpreted function symbols:** Dominik Kohns

- **Conjunctive fragment** $\mapsto$ **clauses:** Nico Bartmann

- **Classical data types:** Stefan Strüder: Superposition
Reasoning in complex theories

Modular reasoning in combinations of theories
Disjoint signature: The Nelson-Oppen method

- **Applications: complex data types**
  - Fragment of theory of arrays: Tim Taubitz

  Fragment of theory of pointers: Jouliet Mesto
Logical theories

**Syntactic view**
Axiomatized by a set $\mathcal{F}$ of (closed) first-order $\Sigma$-formulae.
the models of $\mathcal{F}$: $\text{Mod}(\mathcal{F}) = \{ A \in \Sigma\text{-alg} | A \models G, \text{ for all } G \text{ in } \mathcal{F} \}$

$\mathcal{F} \subseteq \text{Th}(\text{Mod}(\mathcal{F}))$ (typically strict)
$\mathcal{M} \subseteq \text{Mod}(\text{Th}(\mathcal{M}))$ (typically strict)

**Semantic view**
given a class $\mathcal{M}$ of $\Sigma$-structures
the first-order theory of $\mathcal{M}$: $\text{Th}(\mathcal{M}) = \{ G \in F_\Sigma(X) \text{ closed} | \mathcal{M} \models G \}$

$\text{Th}(\text{Mod}(\mathcal{F}))$ the set of formulae true in all models of $\mathcal{F}$ represents exactly the set of consequences of $\mathcal{F}$
1. **Linear integer arithmetic.** $\Sigma = (\{0/0, s/1, +/2\}, \{\leq /2\})$

$\mathbb{Z}_+ = (\mathbb{Z}, 0, s, +, \leq)$ the standard interpretation of integers.

$\{\mathbb{Z}_+\} \subset \text{Mod}(\text{Th}(\mathbb{Z}_+))$

2. **Uninterpreted function symbols.** $\Sigma = (\Omega, \text{Pred})$

$\mathcal{M} = \Sigma\text{-alg}$: the class of all $\Sigma$-structures

The theory of uninterpreted function symbols is $\text{Th}(\Sigma\text{-alg})$ the family of all first-order formulae which are true in all $\Sigma$-structures.
Examples

3. Lists. $\Sigma = (\{\text{car/1, cdr/1, cons/2}\}, \emptyset)$

$\mathcal{F} = \left\{ \begin{array}{l}
\text{car}(\text{cons}(x, y)) \approx x \\
\text{cdr}(\text{cons}(x, y)) \approx y \\
\text{cons}(\text{car}(x), \text{cdr}(x)) \approx x
\end{array} \right.$

$\text{Mod}(\mathcal{F})$: the class of all models of $\mathcal{F}$

$\text{Th}_{\text{Lists}} = \text{Th}(\text{Mod}(\mathcal{F}))$ theory of lists (axiomatized by $\mathcal{F}$)
Decidable theories

\[ \Sigma = (\Omega, \text{Pred}) \] be a signature.

\( \mathcal{M} \): class of \( \Sigma \)-structures. \( \mathcal{T} = \text{Th}(\mathcal{M}) \) is decidable iff there is an algorithm which, for every closed first-order formula \( \phi \), can decide (after a finite number of steps) whether \( \phi \) is in \( \mathcal{T} \) or not.

\( \mathcal{F} \): class of (closed) first-order formulae.

The theory \( \mathcal{T} = \text{Th}(\text{Mod}(\mathcal{F})) \) is decidable iff there is an algorithm which, for every closed first-order formula \( \phi \), can decide (in finite time) whether \( \mathcal{F} \models \phi \) or not.
Examples

Undecidable theories

• Peano arithmetic

    Axiomatized by:

    \[ \forall x \neg(x + 1 \approx 0) \]  (zero)

    \[ \forall x \forall y (x + 1 \approx y + 1 \rightarrow x \approx y) \]  (successor)

    \[ F[0] \land (\forall x (F[x] \rightarrow F[x + 1]) \rightarrow \forall x F[x]) \]  (induction)

    \[ \forall x (x + 0 \approx x) \]  (plus zero)

    \[ \forall x, y (x + (y + 1) \approx (x + y) + 1) \]  (plus successor)

    \[ \forall x, y (x \times 0 \approx 0) \]  (times zero)

    \[ \forall x, y (x \times (y + 1) \approx x \times y + x) \]  (times successor)

    \[ 3 \times y + 5 > 2 \times y \] expressed as \( \exists z (z \neq 0 \land 3 \times y + 5 \approx 2 \times y + z) \)

    Intended interpretation: \( (\mathbb{N}, \{0, 1, +, \times\}, \{\approx, \leq\}) \)

    (does not capture true arithmetic by Gödel’s incompleteness theorem)

• \( \text{Th}((\mathbb{Z}, \{0, 1, +, \times\}, \{\leq\})) \)

• \( \text{Th}(\Sigma\text{-alg}) \)
Examples

In order to obtain decidability results:

- Restrict the signature
- Enrich axioms
- Look at certain fragments
Examples

In order to obtain decidability results:

• Restrict the signature

• Enrich axioms

• Look at certain fragments

Decidable theories

• Presburger arithmetic decidable in 3EXPTIME [Presburger’29]
  Signature: (\{0, 1, +\}, \{\approx, \leq\}) (no *)
  Axioms \{ (zero), (successor), (induction), (plus zero), (plus successor) \}

A decision procedure will be presented by Delzar Habash

A quantifier-elimination method with be presented by Denis Oldenburg

A simple fragment (UTVPI) with be presented by Matthias Becker
Examples

In order to obtain decidability results:

• Restrict the signature

• Enrich axioms

• Look at certain fragments

Decidable theories

• The theory of real numbers (with addition and multiplication) is decidable in 2EXPTIME [Tarski’30]
Examples

In order to obtain decidability results:

- Restrict the signature
- Enrich axioms
- Look at certain fragments $\mathcal{L} \subseteq \text{Fma}(\Sigma)$

“Simpler” task: Given $\phi$ in $\mathcal{L}$, is it the case that $\mathcal{T} \models \phi$?

Common restrictions on $\mathcal{L}$

<table>
<thead>
<tr>
<th>$\mathcal{L}$</th>
<th>Pred = $\emptyset$</th>
<th>${ \phi \in \mathcal{L} \mid \mathcal{T} \models \phi }$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${ \forall x A(x) \mid A \text{ atomic} }$</td>
<td>word problem</td>
<td></td>
</tr>
<tr>
<td>${ \forall x (A_1 \land \ldots \land A_n \rightarrow B) \mid A_i, B \text{ atomic} }$</td>
<td>uniform word problem $\text{Th}_{\forall \text{Horn}}$</td>
<td></td>
</tr>
<tr>
<td>${ \forall x C(x) \mid C(x) \text{ clause} }$</td>
<td>clausal validity problem $\text{Th}_{\forall,cl}$</td>
<td></td>
</tr>
<tr>
<td>${ \forall x \phi(x) \mid \phi(x) \text{ unquantified} }$</td>
<td>universal validity problem $\text{Th}_{\forall}$</td>
<td></td>
</tr>
</tbody>
</table>
Validity of $\forall$ formulae vs. ground satisfiability

The following are equivalent:

(1) $\mathcal{T} \models \forall x (L_1(x) \lor \cdots \lor L_n(x))$

(2) There is no model of $\mathcal{T}$ which satisfies $\exists x (\neg L_1(x) \land \cdots \land \neg L_n(x))$

(3) There is no model of $\mathcal{T}$ and no valuation for the constants $c$ for which $(\neg L_1(c) \land \cdots \land \neg L_n(c))$ becomes true

(4) (notation: $(\neg L_1(c) \land \cdots \land \neg L_n(c)) \models_{\mathcal{T}} \perp$)

Can reduce any validity problem to a ground satisfiability problem
Useful theories

Many example of theories in which ground satisfiability is decidable:

- The empty theory (no axioms) $UIF(\Sigma)$: Dominik Kohns
- theories axiomatizing common datatypes: Stefan Strüder
Combination of theories
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 $A \in \Sigma'$-alg, we denote by $A|_{\Sigma}$ the $\Sigma$-structure for which:

$U_{A|_{\Sigma}} = U_A$, $f_{A|_{\Sigma}} = f_A$ for $f \in \Omega$; $P_{A|_{\Sigma}} = P_A$ for $P \in \Pi$

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

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

**Example:**

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

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

$\mathcal{N} = (\mathbb{N}, +, *, 1, \leq, \text{even}, \text{odd})$  $\mathcal{N}|_{\Sigma} = (\mathbb{N}, +, 1, \leq)$
Combining theories

**Syntactic view:** \( T_1 + T_2 = T_1 \cup T_2 \subseteq F_{\Sigma_1 \cup \Sigma_2}(X) \)

\[
\text{Mod}(T_1 \cup T_2) = \{ A \in (\Sigma_1 \cup \Sigma_2)\text{-alg} \mid A \models G, \text{ for all } G \text{ in } T_1 \cup T_2 \}
\]

**Semantic view:** Let \( M_i = \text{Mod}(T_i), i = 1, 2 \)

\[
M_1 + M_2 = \{ A \in (\Sigma_1 \cup \Sigma_2)\text{-alg} \mid A|_{\Sigma_i} \in M_i \text{ for } i = 1, 2 \}
\]

\[
A \in \text{Mod}(T_1 \cup T_2) \iff A \models G, \text{ for all } G \text{ in } T_1 \cup T_2
\]

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

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

\[
\text{iff } A \in M_1 + M_2
\]
Example

1. Presburger arithmetic + UIF

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

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

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

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

**Goal: Modular Reasoning**

Example:
\[
\text{lists}(\mathbb{R}) \cup \text{arrays}(\mathbb{R})
\]

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

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?
Combination of theories over disjoint signatures

The Nelson/Oppen procedure

**Given:** $\mathcal{T}_1, \mathcal{T}_2$ 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. $\mathcal{T}_i$

$\phi$ quantifier-free formula over $\Sigma_1 \cup \Sigma_2$

**Task:** Check whether $\phi$ is satisfiable w.r.t. $\mathcal{T}_1 \cup \mathcal{T}_2$

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

$\phi \mapsto DNF(\phi)$

$DNF(\phi)$ satisfiable in $\mathcal{T}$ iff one of the disjuncts satisfiable in $\mathcal{T}$
Example

[Nelson & Oppen, 1979]

Theories

<table>
<thead>
<tr>
<th>Theory</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{R} ) theory of rationals</td>
<td>( \Sigma_\mathcal{R} = { \leq, +, -, 0, 1 } ) ≈</td>
</tr>
<tr>
<td>( \mathcal{L} ) theory of lists</td>
<td>( \Sigma_\mathcal{L} = { \text{car, cdr, cons} } ) ≈</td>
</tr>
<tr>
<td>( \mathcal{E} ) theory of equality (UIF)</td>
<td>( \Sigma: ) free function and predicate symbols ≈</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></th>
<th>(\mathcal{R})</th>
<th>(\mathcal{L})</th>
<th>(\mathcal{E})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Sigma)</td>
<td>({\leq, +, −, 0, 1})</td>
<td>{car, cdr, cons}</td>
<td>(F \cup P)</td>
</tr>
<tr>
<td>Axioms</td>
<td>(x + 0 \approx x)</td>
<td>car(cons((x, y))) \approx x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x − x \approx 0)</td>
<td>cdr(cons((x, y))) \approx y</td>
<td></td>
</tr>
<tr>
<td>(univ. quantif.)</td>
<td>(+ is A, C)</td>
<td>at((x)) \lor \text{cons}(\text{car}(x), \text{cdr}(x)) \approx x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\leq is R, T, A)</td>
<td>(\neg at(\text{cons}(x, y)))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x \leq y \lor y \leq x)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x \leq y \rightarrow x + z \leq y + z)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Is the following conjunction:

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

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 $\phi$

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

where $t \approx f(t_1, \ldots, t_n)$

Termination: Obvious

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

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

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

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

<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>( 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>
</tbody>
</table>
| \( c_5 \approx 0 \) | \( c_4 \approx h(d) \) | \n
36
Step 1: Purification

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

**R** | **L** | **E**
--- | --- | ---
\( c \leq d \) | \( c_1 \approx \text{car}(\text{cons}(c_5, c)) \) | \( P(c_2) \)
\( d \leq c + c_1 \) |  | \( \neg P(c_5) \)
\( c_2 \approx c_3 - c_4 \) |  | \( c_3 \approx h(c) \)
\( c_5 \approx 0 \) |  | \( c_4 \approx h(d) \)
satisfiable | satisfiable | satisfiable
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>
</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>( c_2 \approx c_3 - c_4 )</td>
<td>( \neg P(c_5) )</td>
</tr>
<tr>
<td>( c_5 \approx 0 )</td>
<td>( c_3 \approx h(c) )</td>
<td>( c_4 \approx h(d) )</td>
</tr>
</tbody>
</table>

\( c_1 \approx c_5 \)
Step 2: Propagation

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

\[ \begin{array}{ccc}
\text{R} & \text{L} & \mathcal{E} \\
\hline
\text{c} \leq \text{d} & \text{c}_1 \approx \text{car}(\text{cons}(\text{c}_5, \text{c})) & P(\text{c}_2) \\
\text{d} \leq \text{c} + \text{c}_1 & & \neg P(\text{c}_5) \\
\text{c}_2 \approx \text{c}_3 - \text{c}_4 & & \text{c}_3 \approx h(\text{c}) \\
\text{c}_5 \approx 0 & & \text{c}_4 \approx h(\text{d}) \\
\text{c}_1 \approx \text{c}_5 & & \\
\text{c} \approx \text{d} & & \\
\end{array} \]
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>( c_2 \approx c_3 - c_4 )</td>
<td>( \neg P(c_5) )</td>
</tr>
<tr>
<td>( c_5 \approx 0 )</td>
<td>( c_3 \approx h(c) )</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>( c_3 \approx c_4 )</td>
<td></td>
</tr>
<tr>
<td>( c_2 \approx c_5 )</td>
<td>( \perp )</td>
<td></td>
</tr>
</tbody>
</table>
The Nelson-Oppen algorithm

ϕ conjunction of literals

**Step 1.** Purification \( T_1 \cup T_2 \cup \phi \rightarrow (T_1 \cup \phi_1) \cup (T_2 \cup \phi_2) \):

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 \( T_1 \cup T_2 \cup \phi \mapsto (T_1 \cup \phi_1) \cup (T_2 \cup \phi_2) \):
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.

Sound: if inconsistency detected input unsatisfiable
Complete: under additional assumptions

not problematic; requires linear time
not problematic; termination guaranteed
The Nelson-Oppen algorithm

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

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

**Completeness:** Under additional hypotheses

Consider stably infinite theories.

$\mathcal{T}$ is stably infinite iff for every quantifier-free formula $\phi$

$\phi$ satisfiable in $\mathcal{T}$ iff $\phi$ satisfiable in an infinite model of $\mathcal{T}$.

**Note:** This restriction is not mentioned in [Nelson Oppen 1979]; introduced by Oppen in 1980.

With this additional condition completeness can be proved.
Applications

1. Decision Procedures for data types

- A decidable fragment of the theory of arrays
  \(\mapsto\) reduction to reasoning in the combination of Presburger arithmetic and uninterpreted function symbols
  
  Tim Taubitz

- A decidable fragment of the theory of pointer structures
  \(\mapsto\) reduction to reasoning in the combination of the theory uninterpreted function symbols and the \(\beta\)calartheories.
  
  Jouliet Mesto
2. Program Verification

\[
T = (\Sigma, \text{Init}, \text{Update}(\Sigma, \Sigma'))
\]

Safety Property \mapsto \text{Formula Safe}

**Task:** Prove that the safety property always holds (in general difficult)

**Invariant checking**

\[
\text{Init} \models \text{Safe}
\]

\[
\text{Safe} \land \text{Update}(\Sigma, \Sigma') \models \text{Safe}'
\]

**Bounded model checking:** given \( k \in \mathbb{N} \). Prove that for all \( n \leq k \):

\[
\text{Init}(\Sigma^0) \land \text{Update}|(\Sigma^0, \Sigma^1) \land \cdots \land \text{Update}|(\Sigma^{n-1}, \Sigma^n) \models \text{Safe}(\Sigma^n)
\]
Summary

- Logical Theories
- Decidability/Undecidability
- Combination of Logical Theories
  - The Nelson/Oppen Method for reasoning in combinations of theories with disjoint signatures
- Applications