Decision Procedures in Verification

Decision Procedures (3)

19.12.2013

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Until now:

Decidable subclasses of FOL

The Bernays-Schönfinkel class

(definition; decidability;tractable fragment: Horn clauses)

The Ackermann class

The monadic class

Decision problems/restrictions

Uninterpreted function symbols

Decision procedures for numeric domains

Difference logic

Positive difference logic

Syntax

The syntax of formulae in positive difference logic is defined as follows:

- Atomic formulae (also called difference constraints) are of the form:

  $$x - y \leq c$$

  where x, y are variables and c is a numerical constant.

- The set of formulae is:

  $$\begin{array}{rcll} F, G, H & ::= & A & \text{(atomic formula)} \\ & \mid & (F \wedge G) & \text{(conjunction)} \end{array}$$

Semantics:

Versions of difference logic exist, where the standard interpretation is $\mathbb{Q}$ or resp. $\mathbb{Z}$.

Positive difference logic

A decision procedure for positive difference logic ($\leq$ only)

Let S be a set (i.e. conjunction) of atoms in (positive) difference logic. $G(S) = (V, E, w)$, the inequality graph of S, is a weighted graph with:

- $V = X(S)$, the set of variables occurring in S
- $e = (x, y) \in E$ with $w(e) = c$ iff $x - y \leq c \in S$

Theorem 3.4.1.
Let S be a conjunction of difference constraints, and $G(S)$ the inequality graph of S. Then S is satisfiable iff there is no negative cycle in $G(S)$.

Searching for negative cycles in a graph can be done with the Bellman-Ford algorithm for finding the single-source shortest paths in a directed weighted graph in time $O(|V| \cdot |E|)$. (Side-effect of the algorithm exploited - if there exists a negative cycle in the graph then the algorithm finds it and aborts.)

Positive difference logic

Theorem 3.4.1.
Let S be a conjunction of difference constraints, and $G(S)$ the inequality graph of S. Then S is satisfiable iff there is no negative cycle in $G(S)$.

Proof: $(\Rightarrow)$ Assume S satisfiable. Let $\beta : X \rightarrow \mathbb{Z}$ satisfying assignment.
Let $v_1 \stackrel{c_{12}}{\rightarrow} v_2 \stackrel{c_{23}}{\rightarrow} \cdots \stackrel{c_{n-1,n}}{\rightarrow} v_n \stackrel{c_{n1}}{\rightarrow} v_1$ be a cycle in $G(S)$.

$$
\begin{array}{rlcl}
\text{Then:} & \beta(v_1) - \beta(v_2) & \leq & c_{12} \\
& \beta(v_2) - \beta(v_3) & \leq & c_{23} \\
& \cdots & & \\
g & \beta(v_n) - \beta(v_1) & \leq & c_{n1} \\
\hline
0 = & \beta(v_1) - \beta(v_1) & \leq & \sum_{i=1}^{n-1} c_{i,i+1} + c_{n1}
\end{array}
$$

Thus, for satisfiability it is necessary that all cycles are positive.

Positive difference logic

Theorem 3.4.1.
Let S be a conjunction of difference constraints, and $G(S)$ the inequality graph of S. Then S is satisfiable iff there is no negative cycle in $G(S)$.

Proof: $(\Leftarrow)$ Assume that there is no negative cycle.

Add a new vertex s and an 0-weighted edge from every vertex in V to s. (This does not introduce negative cycles.)

Let δ_{uv} denote the minimal weight of the paths from u to v.

- $\delta_{uv} = \infty$ if there is no path from u to v.
- well-defined since there are no negative cycles

Define $\beta : V \to \mathbb{Z}$ by $\beta(v) = \delta_{vs}$. Claim: β satisfying assignment for S.

Let $x - y \leq c \in S$. Consider the shortest paths from x to s and from y to s. By the triangle inequality, $\delta_{xs} \leq c + \delta_{ys}$, i.e. $\beta(x) - \beta(y) \leq c$.

Difference logic

Syntax

- Atomic formulae (difference constraints): $x - y \leq c$

  where x, y are variables and c is a numerical constant.

- Formulae: $F, G, H \quad ::= \quad A$ (atomic formula)

$$
\begin{array}{llll}
F, G, H & ::= & A & \text{(atomic formula)} \\
 & \mid & \neg A & \\
 & \mid & (F \wedge G) & \text{(conjunction)}
\end{array}
$$

Note: $\neg(x - y \leq c)$ is equivalent to $y - x < c$.

Difference logic

Syntax

- Atomic formulae (difference constraints): $x - y \leq c$

  where x, y are variables and c is a numerical constant.

- Formulae: $F, G, H \quad ::= \quad A$ (atomic formula)

  $\quad\quad\quad\quad\quad\quad\quad\quad | \quad \neg A$

  $\quad\quad\quad\quad\quad\quad\quad\quad | \quad (F \wedge G)$ (conjunction)

Note: $\neg(x - y \leq c)$ is equivalent to $y - x < c$.

Satisfiability over $\mathbb{Z}$

$y - x < c$ iff $y - x \leq c - 1$

Natural reduction to positive difference logic.

Difference logic

Syntax

- Atomic formulae (difference constraints): $x - y \leq c$

  where x, y are variables and c is a numerical constant.

- Formulae: $F, G, H \quad ::= \quad A$ (atomic formula)

  $\quad\quad\quad\quad\quad\quad\quad\quad | \quad \neg A$

  $\quad\quad\quad\quad\quad\quad\quad\quad | \quad (F \wedge G)$ (conjunction)

Note: $\neg(x - y \leq c)$ is equivalent to $y - x < c$.

Theorem 3.4.2.

Let S be a conjunction of strict and non-strict difference constraints, and $G(S)$ the inequality graph of S. Then S is satisfiable iff there is no negative cycle in $G(S)$.

Difference logic

Theorem 3.4.2.
Let S be a conjunction of strict and non-strict difference constraints, and $G(S)$ the inequality graph of S. Then S is satisfiable iff there is no negative cycle in $G(S)$.

Proof:
Need to extend the graph construction and the unsatisfiability condition:

$x_1 - x_2 \prec_1 c_1, \ldots, x_n - x_1 \prec_n c_n$ unsatisfiable iff

- $\sum_{i=1}^n c_i < 0$, or
- $\sum_{i=1}^n c_i = 0$ and one $\prec_i$ is strict.

Consider pairs $(\prec, c)$ instead of numbers c

- $(\prec, c) <_B (\prec', c')$ iff $c < c'$ or ($c = c'$, $\prec_1 = <$ and $\prec_2 = \leq$)
- $(\prec, c) + (\prec', c') = (\prec'', c + c')$ where $\prec'' = <$ iff $\prec$ or $\prec'$ is $<$.

Linear arithmetic over $\mathbb{N}$ or $\mathbb{Z}$

1. Th($\mathbb{Z}_+$) $\quad$ $\mathbb{Z}_+ = (\mathbb{Z}, 0, s, +, <)$ the standard interpretation of integers.
2. **Presburger arithmetic.**

Axiomatization:

$\forall x\, \neg(x + 1 \approx 0)$	(zero)
$\forall x \forall y\, (x + 1 \approx y + 1 \rightarrow x \approx y$	(successor)
$F[0] \wedge (\forall x\, (F[x] \rightarrow F[x+1]) \rightarrow \forall x F[x])$	(induction)
$\forall x\, (x + 0 \approx x)$	(plus zero)
$\forall x, y\, (x + (y + 1) \approx (x + y) + 1)$	(plus successor)

Linear arithmetic over $\mathbb{N}$ or $\mathbb{Z}$

Presburger arithmetic decidable in 3EXPTIME [Presburger'29]

- automata theoretic method

Linear arithmetic over $\mathbb{Z}$:

check satisfiability of conjunctions of (in)equalities over $\mathbb{Z}$: NP-hard

- Integer linear programming

  use branch-and-bound/cutting planes

- The Omega test – use variable elimination

Linear arithmetic over $\mathbb{R}$ or $\mathbb{Q}$

- Th($\mathbb{R}$)
  $\mathbb{R} = (\mathbb{R}, \{0, 1, +\}, \{<\})$ the standard interpretation of real numbers;

- Th($\mathbb{Q}$)
  $\mathbb{Q} = (\mathbb{Q}, \{0, 1, +\}, \{<\})$ the standard interpretation of rational numbers.

Linear arithmetic over $\mathbb{R}$ or $\mathbb{Q}$

Axiomatization:

The equational part of linear rational arithmetic is described by the theory of divisible torsion-free abelian groups:

$$\forall x, y, z(x + (y + z) \approx (x + (y + z))) \qquad \text{(associativity)}$$

$$\forall x, y(x + y \approx y + x) \qquad \text{(commutativity)}$$

$$\forall x(x + 0 \approx x) \qquad \text{(identity)}$$

$$\forall x \exists y(x + y \approx 0) \qquad \text{(inverse)}$$

$$\text{For all } n \geq 1\text{: } \forall x(\underbrace{x + \cdots + x}_{n \text{ times}} \approx 0 \to x \approx 0) \qquad \text{(torsion-freeness)}$$

$$\text{For all } n \geq 1 : \forall x \exists y(\underbrace{y + \cdots + y}_{n \text{ times}} \approx x) \qquad \text{(divisibility)}$$

$$\neg 1 \approx 0 \qquad \text{(non-triviality)}$$

Note: Quantification over natural numbers is not part of our language. We really need infinitely many axioms for torsion-freeness and divisibility.

Linear arithmetic over $\mathbb{R}$ or $\mathbb{Q}$

By adding the axioms of a compatible strict total ordering, we define ordered divisible abelian groups:

$$\forall x\ (\neg x < x) \qquad \text{(irreflexivity)}$$

$$\forall x, y, z\ (x < y \land y < z \rightarrow x < z) \qquad \text{(transitivity)}$$

$$\forall x, y\ (x < y \lor y < x \lor x \approx y) \qquad \text{(totality)}$$

$$\forall x, y, z\ (x < y \rightarrow x + z < y + z) \qquad \text{(compatibility)}$$

$$0 < 1 \qquad \text{(non-triviality)}$$

Note: The second non-triviality axiom renders the first one superfluous.

Moreover, as soon as we add the axioms of compatible strict total orderings, torsion-freeness can be omitted.

Every ordered divisible abelian group is obviously torsion-free. In fact the converse holds: Every torsion-free abelian group can be ordered [F.-W. Levi, 1913].

Examples: $\mathbb{Q}, \mathbb{R}, \mathbb{Q}^n, \mathbb{R}^n, \ldots$

Linear arithmetic over $\mathbb{R}$ or $\mathbb{Q}$

The signature can be extended by further symbols:

- $\leq/2, >/2, \geq/2, \not\approx/2$: defined using $<$ and $\approx$
- $-/1$: Skolem function for inverse axiom
- $-/2$: defined using $+/2$ and $-/1$
- $\mathrm{div_n}/1$: Skolem functions for divisibility axiom for all $n \geq 1$.
- $\mathrm{mult_n}/1$: defined by $\forall x(\mathrm{mult_n}(x) \approx \underbrace{x + \cdots + x}_{n \text{ times}}$ for all $n \geq 1$.
- $\mathrm{mult_q}/1$: defined using $\mathrm{mult_n}, \mathrm{div_n}, -$ for all $q \in \mathbb{Q}$.
  (We usually write $q \cdot t$ or qt instead of $\mathrm{mult_q}(t)$.)
- $q/0$ (for $q \in \mathbb{Q}$): defined by $q \approx q \cdot 1$.

Note: Every formula using the additional symbols is ODAG-equivalent to a formula over the base signature.

When $\cdot$ is considered as a binary operator, (ordered) divisible torsion-free abelian groups correspond to (ordered) rational vector spaces.

Linear arithmetic over $\mathbb{R}$ or $\mathbb{Q}$

Theorem.

(1) The satisfiability of any conjunction of (strict and non-strict) linear inequalities can be checked in PTIME [Khakian'79].

(2) The complexity of checking the satisfiability of sets of clauses in linear arithmetic is in NP [Sonntag'85].

Literature

[Khakian'79] L. Khachian. "A polynomial time algorithm for linear programming." *Soviet Math. Dokl.* 20:191-194, 1979.

[Sonntag'85] E.D. Sontag. "Real addition and the polynomial hierarchy". *Inf. Proc. Letters* 20(3):115-120, 1985.

Linear arithmetic over $\mathbb{R}$ or $\mathbb{Q}$

Methods The algorithms currently used are not PTIME.

- The simplex method
- The Fourier-Motzkin method – use variable elimination

Linear arithmetic: Comparison

Problem:

check satisfiability of conjunctions of equalities over a numerical domain D

Complexity: $D = \mathbb{R}$: PTIME; $D = \mathbb{Z}$: NP-hard

Methods

- The simplex method ($D = \mathbb{R}$)
- Integer linear programming ($D = \mathbb{Z}$)

  use branch-and-bound/cutting planes

- The Fourier-Motzkin method ($D = \mathbb{R}$)

  use variable elimination

- The Omega test ($D = \mathbb{Z}$)

  use variable elimination

Linear arithmetic: Comparison

Problem:

check satisfiability of conjunctions of equalities over a numerical domain D

Complexity: $D = \mathbb{R}$: PTIME; $D = \mathbb{Z}$: NP-hard

Methods

- The simplex method ($D = \mathbb{R}$)
- Integer linear programming ($D = \mathbb{Z}$)

  use branch-and-bound/cutting planes

- The Fourier-Motzkin method ($D = \mathbb{R}$)

  use variable elimination

- The Omega test ($D = \mathbb{Z}$)

  use variable elimination

Fourier-Motzkin Quantifier Elimination

Linear rational arithmetic permits quantifier elimination:
every formula $\exists x F$ or $\forall x F$ in linear rational arithmetic can be converted into an equivalent formula without the variable x.

The method was discovered in 1826 by J. Fourier and re-discovered by T. Motzkin in 1936.

Observation: Every literal over the variables $x, y_1, \dots, y_n$ can be converted into an ODAG-equivalent atom $x \sim t[\overline{y}]$ or $0 \sim t[\overline{y}]$, where $\sim \in \{<, >, \leq, \geq, \approx, \not\approx\}$ and $t[\overline{y}]$ has the form $\sum_i q_i \cdot y_i + q_0$.

In other words, we can either eliminate x completely or isolate it on one side of the atom.

Moreover, we can convert every $\not\approx$ atom into an ODAG-equivalent disjunction of two $<$ atoms.

Fourier-Motzkin Quantifier Elimination

We first consider existentially quantified conjunctions of atoms.

(1) **If the conjunction contains an equation $x \approx t[\overline{y}]$**,
we can eliminate the quantifier $\exists x$ by substitution:

$$\exists x(x \approx t[\overline{y}] \wedge F)$$

is equivalent to

$$F\sigma, \text{ where } \sigma = [t[\overline{y}]/x]$$

Fourier-Motzkin Quantifier Elimination

We first consider existentially quantified conjunctions of atoms.

(2) **If x occurs only in inequations, then:**

$$\begin{aligned}\exists x \quad (&\textstyle\bigwedge_i x < s_i(\overline{y}) \wedge \bigwedge_j x \leq t_j(\overline{y}) \wedge \\ &\textstyle\bigwedge_k x > u_k(\overline{y}) \wedge \bigwedge_l x \geq v_l(\overline{y}) \wedge \\ &F(\overline{y}))\end{aligned}$$

is equivalent to:

$$\begin{aligned}&\textstyle\bigwedge_i \bigwedge_k s_i(\overline{y}) > u_k(\overline{y}) \wedge \bigwedge_j \bigwedge_k t_j(\overline{y}) > u_k(\overline{y}) \wedge \\ &\textstyle\bigwedge_i \bigwedge_l s_i(\overline{y}) > v_l(\overline{y}) \wedge \bigwedge_j \bigwedge_l t_j(\overline{y}) \geq v_l(\overline{y}) \wedge \\ &F(\overline{y})\end{aligned}$$

Proof: "$\Rightarrow$" follows by transitivity;

"$\Leftarrow$" Take $\frac{1}{2}(\min\{s_i, t_j\} + \max\{u_k, v_l\})$ as a witness.

Fourier-Motzkin Quantifier Elimination

Extension to arbitrary formulas:

- Transform into prenex formula;
- If innermost quantifier is $\exists$:
  transform matrix into DNF and move $\exists$ into disjunction;
- If innermost quantifier is $\forall$: replace $\forall x F$ by $\neg\exists x \neg F$, then eliminate $\exists$.

Consequences:

(1) Every closed formula over the signature of ODAGs is ODAG-equivalent to either $\top$ or $\bot$.

(2) ODAGs are a complete theory, i.e., every closed formula over the signature of ODAGs is either valid or unsatisfiable w.r.t. ODAGs.

(3) Every closed formula over the signature of ODAGs holds either in all ODAGs or in no ODAG.

ODAGs are indistinguishable by first-order formulas over the signature of ODAGs. (These properties do not hold for extended signatures!)

Fourier-Motzkin: Complexity

- **One FM-step for $\exists$:**

  formula size grows quadratically, therefore $O(n^2)$ runtime.

- m **quantifiers $\exists \ldots \exists$:**

  naive implementation needs $O(n^{2^m})$ runtime;

  It is unknown whether optimized implementation with simply exponential runtime is possible.

- m **quantifiers $\exists\forall\exists\forall \ldots \exists\forall$:**

  CNF/DNF conversion (exponential!) required after each step; therefore non-elementary runtime.

Fourier-Motzkin: Complexity

- **One FM-step for $\exists$:**

  formula size grows quadratically, therefore $O(n^2)$ runtime.

- ***m*** **quantifiers $\exists \ldots \exists$:**

  naive implementation needs $O(n^{2^m})$ runtime;

  It is unknown whether optimized implementation with simply exponential runtime is possible.

- ***m*** **quantifiers $\exists\forall\exists\forall \ldots \exists\forall$:**

  CNF/DNF conversion (exponential!) required after each step; therefore non-elementary runtime.

Improvement: Loos-Weispfenning Quantifier Elimination

Loos-Weispfenning Quantifier Elimination

A more efficient way to eliminate quantifiers in linear rational arithmetic was developed by R. Loos and V. Weispfenning (1993).

The method is also known as "test point method" or "virtual substitution method".

For simplicity, we consider only one particular ODAG, namely $\mathbb{Q}$ (as we have seen above, the results are the same for all ODAGs).

Loos-Weispfenning Quantifier Elimination

Let $F(x, \overline{y})$ be a positive boolean combination of linear (in-)equations of the form $x \sim_i s_i(\overline{y})$ and $0 \sim_j s_j(\overline{y})$ with $\sim_i, \sim_j \in \{\approx, \not\approx, <, \leq, >, \geq\}$, (i.e. a formula built from linear (in-) equations, $\vee$ and $\wedge$, but without $\neg$).

Goal: Find a finite set T of "test points" so that

$$\exists x F(x, \overline{y}) \models\!\!\mid \bigvee_{t \in T} F(x, \overline{y})[t/x].$$

In other words:

We want to replace the infinite disjunction $\exists x$ by a finite disjunction.

Loos-Weispfenning Quantifier Elimination

If we keep the values of the variables $\overline{y}$ fixed, we can regard F as a function

$$F : \mathbb{Q} \to \{0, 1\} \quad \text{defined by } x \mapsto F(x, \overline{y})$$

Remarks:

(1) The value of each of the atoms $s_i(\overline{y}) \sim_i x$ changes only at $s_i(\overline{y})$,
(2) The value of F can only change if the value of one of its atoms changes.

(3) F is a piecewise constant function; more precisely:
the set of all x with $F(x, \overline{y}) = 1$ is a finite union of intervals.

(The union may be empty, the individual intervals may be finite or infinite and open or closed.)

Let $\delta(\overline{y}) = \min\{|s_i(\overline{y}) - s_j(\overline{y})| \mid s_i(\overline{y}) \neq s_j(\overline{y})\}$.

Each of the intervals has either length 0 (i.e., it consists of one point), or its length is at least $\delta(\overline{y})$.

Loos-Weispfenning Quantifier Elimination

If the set of all x for which $F(x, \overline{y})$ is 1 is non-empty, then

(i) $F(x, \overline{y}) = 1$ for all $x \leq r(\overline{y})$ for some $r(\overline{y}) \in \mathbb{Q}$

(ii) or there is some point where the value of $F(x, \overline{y})$ switches from 0 to 1 when we traverse the real axis from $-\infty$ to $+\infty$.

We use this observation to construct a set of test points.

Loos-Weispfenning Quantifier Elimination

We start with a "sufficiently small" test point $r(\overline{y})$ to take care of case (i).

For case (ii), we observe that $F(x, \overline{y})$ can only switch from 0 to 1 if one of the atoms switches from 0 to 1. (We consider only positive boolean combinations of atoms and $\wedge$ and $\vee$ are monotonic w.r.t. truth values.)

- $x \leq s_i(\overline{y})$ and $x < s_i(\overline{y})$ do not switch from 0 to 1 when x grows.
- $x \geq s_i(\overline{y})$ and $x \approx s_i(\overline{y})$ switch from 0 to 1 at $s_i(\overline{y})$
  $\Rightarrow$ $s_i(\overline{y})$ is a test point.
- $x > s_i(\overline{y})$ and $x \not\approx s_i(\overline{y})$ switch from 0 to 1 "right after" $s_i(\overline{y})$
  $\Rightarrow$ $s_i(\overline{y}) + \epsilon$ (for some $0 < \epsilon < \delta(\overline{y})$) is a test point.

Loos-Weispfenning Quantifier Elimination

We start with a "sufficiently small" test point $r(\overline{y})$ to take care of case (i).

For case (ii), we observe that $F(x, \overline{y})$ can only switch from 0 to 1 if one of the atoms switches from 0 to 1. (We consider only positive boolean combinations of atoms and $\wedge$ and $\vee$ are monotonic w.r.t. truth values.)

- $x \leq s_i(\overline{y})$ and $x < s_i(\overline{y})$ do not switch from 0 to 1 when x grows.
- $x \geq s_i(\overline{y})$ and $x \approx s_i(\overline{y})$ switch from 0 to 1 at $s_i(\overline{y})$
  $\Rightarrow$ $s_i(\overline{y})$ is a test point.
- $x > s_i(\overline{y})$ and $x \not\approx s_i(\overline{y})$ switch from 0 to 1 "right after" $s_i(\overline{y})$
  $\Rightarrow$ $s_i(\overline{y}) + \epsilon$ (for some $0 < \epsilon < \delta(\overline{y})$) is a test point.

If $r(\overline{y})$ is sufficiently small and $0 < \epsilon < \delta(\overline{y})$, then

$$T := \{r(\overline{y})\} \cup \{s_i(\overline{y}) \mid \sim_i \in \{\geq, \approx\}\} \cup \{s_i(\overline{y}) + \epsilon \mid \sim_i \in \{>, \not\approx\}\}.$$

is a set of test points.

Loos-Weispfenning Quantifier Elimination

Problems:

(1) We don't know how small $r(\overline{y})$ has to be for case (i).

(2) We don't know $\delta(\overline{y})$ for case (ii).

Idea: We consider the limits for $r \to -\infty$ and for $\epsilon \to 0$ (but positive), that is, we redefine

$$T := \{-\infty\} \cup \{s_i(\overline{y}) \mid \sim_i \in \{\geq, \approx\}\} \cup \{s_i(\overline{y}) + \epsilon \mid \sim_i \in \{>, \not\approx\}\}.$$

New problem:

How can we eliminate the infinitesimals $-\infty$ and ϵ when we substitute elements of T for x?

Loos-Weispfenning Quantifier Elimination

Virtual substitution:

$(x < s(\overline{y}))[-\infty/x] := \lim_{r \to -\infty}(r < s(\overline{y})) = \top$

$(x \leq s(\overline{y}))[-\infty/x] := \lim_{r \to -\infty}(r \leq s(\overline{y})) = \top$

$(x > s(\overline{y}))[-\infty/x] := \lim_{r \to -\infty}(r > s(\overline{y})) = \bot$

$(x \geq s(\overline{y}))[-\infty/x] := \lim_{r \to -\infty}(r \geq s(\overline{y})) = \bot$

$(x \approx s(\overline{y}))[-\infty/x] := \lim_{r \to -\infty}(r \approx s(\overline{y})) = \bot$

$(x \not\approx s(\overline{y}))[-\infty/x] := \lim_{r \to -\infty}(r \not\approx s(\overline{y})) = \top$

Loos-Weispfenning Quantifier Elimination

Virtual substitution:

$$(x < s(\bar{y}))[u + \epsilon / x] := \lim_{\substack{\epsilon \to 0 \\ \epsilon > 0}} (u + \epsilon < s(\bar{y})) = (u < s(\bar{y}))$$

$$(x \leq s(\bar{y}))[u + \epsilon / x] := \lim_{\substack{\epsilon \to 0 \\ \epsilon > 0}} (u + \epsilon \leq s(\bar{y})) = (u < s(\bar{y}))$$

$$(x > s(\bar{y}))[u + \epsilon / x] := \lim_{\substack{\epsilon \to 0 \\ \epsilon > 0}} (u + \epsilon > s(\bar{y})) = (u \geq s(\bar{y}))$$

$$(x \geq s(\bar{y}))[u + \epsilon / x] := \lim_{\substack{\epsilon \to 0 \\ \epsilon > 0}} (u + \epsilon \geq s(\bar{y})) = (u \geq s(\bar{y}))$$

$$(x \approx s(\bar{y}))[u + \epsilon / x] := \lim_{\substack{\epsilon \to 0 \\ \epsilon > 0}} (u + \epsilon \approx s(\bar{y})) = \perp$$

$$(x \not\approx s(\bar{y}))[u + \epsilon / x] := \lim_{\substack{\epsilon \to 0 \\ \epsilon > 0}} (u + \epsilon \not\approx s(\bar{y})) = \top$$

We have traversed the real axis from $-\infty$ to $+\infty$.

Loos-Weispfenning Quantifier Elimination

Virtual substitution:

Alternatively, we can traverse it from $+\infty$ to $-\infty$.

In this case, the test points are

$$T' := \{+\infty\} \cup \{s_i(\overline{y}) \mid \sim_i \in \{\leq, \approx\}\} \cup \{s_i(\overline{y}) - \epsilon \mid \sim_i \in \{<, \not\approx\}\}.$$

Infinitesimals are eliminated in a similar way as before.

In practice: Compute both T and T' and take the smaller set.

For a universally quantified formula $\forall x F$, we replace it by $\neg \exists x \neg F$, push inner negation downwards, and then continue as before.

Note that there is no CNF/DNF transformation required.

Loos-Weispfenning quantifier elimination works on arbitrary positive formulas.

Loos-Weispfenning: Complexity

- **One LW-step for $\exists$ or $\forall$:**

  As the number of test points is at most equal to the number of atoms, the formula size grows quadratically; therefore $O(n^2)$ runtime.

- **Multiple quantifiers of the same kind:**

$$
\begin{aligned}
& \exists x_2 \exists x_1 . F(x_1, x_2, \overline{y}) \\
\mapsto \quad & \exists x_2 . \bigvee_{t_1 \in T_1} F(x_1, x_2, \overline{y})[t_1/x_1] \\
\mapsto \quad & \bigvee_{t_1 \in T_1} (\exists x_2 . F(x_1, x_2, \overline{y})[t_1/x_1]) \\
\mapsto \quad & \bigvee_{t_1 \in T_1} \bigvee_{t_2 \in T_2} F(x_1, x_2, \overline{y})[t_1/x_1][t_2/x_2]
\end{aligned}
$$

- ***m* quantifiers $\exists$ . . . $\exists$ or $\forall$...$\forall$:**
  formula size is multiplied by n in each step $\Rightarrow O(n^{m+1})$ runtime.

- ***m* quantifiers $\exists\forall\exists\forall \ldots \forall$:** doubly exponential runtime.

Note: The formula resulting from a LW-step is usually highly redundant. An efficient implementation must make use of simplification techniques.

Until now

Decidable fragments of first-order logic

Decision procedures for single theories

- UIF
- Numeric domains

  Here:

  Difference logic

  Linear arithmetic over $\mathbb{R}$, $\mathbb{Q}$

Next: Reasoning in combinations of theories

Combinations of decision procedures

3.5. Combinations of theories

The combined validity problem

For $i = 1, 2$
- let $\mathcal{T}_i$ be a first-order theory in signature Σ_i
- let $\mathcal{L}_i$ be a class of (closed) Σ-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 ϕ in $\mathcal{L}_1 \bigoplus \mathcal{L}_2$, is it the case that $\mathcal{T}_1 \bigoplus \mathcal{T}_2 \models \phi$?

Problems

The combined decidability problem I

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

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 $\mathcal{L}_2$

Question: Is the $\mathcal{T}_1 \bigoplus \mathcal{T}_2$-validity problem for $\mathcal{L}_1 \bigoplus \mathcal{L}_2$ decidable?

Problems

The combined decidability problem II

For $i = 1, 2$
- let $\mathcal{T}_i$ be a first-order theory in signature Σ_i
- let $\mathcal{L}_i$ be a class of (closed) Σ-formulae
- P_i decision procedure for $\mathcal{T}_i$-validity for $\mathcal{L}_i$

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 $\mathcal{L}_2$

Question: Can we combine P_1 and P_2 modularly into a decision procedure for the $\mathcal{T}_1 \bigoplus \mathcal{T}_2$-validity problem for $\mathcal{L}_1 \bigoplus \mathcal{L}_2$?

Main issue: How are $\mathcal{T}_1 \bigoplus \mathcal{T}_2$ and $\mathcal{L}_1 \bigoplus \mathcal{L}_2$ 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 Σ-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 $\Sigma' \backslash \Sigma$)

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

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

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

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

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

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

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

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

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 $\mathcal{M}_i = \text{Mod}(\mathcal{T}_i), i = 1, 2$

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

$$\begin{aligned}
\mathcal{A} \in \text{Mod}(\mathcal{T}_1 \cup \mathcal{T}_2) \quad & \text{iff} \quad \mathcal{A} \models G, \text{ for all } G \text{ in } \mathcal{T}_1 \cup \mathcal{T}_2 \\
& \text{iff} \quad \mathcal{A}_{|\Sigma_i} \models G, \text{ for all } G \text{ in } \mathcal{T}_i, i = 1, 2 \\
& \text{iff} \quad \mathcal{A}_{|\Sigma_i} \in \mathcal{M}_i, i = 1, 2 \\
& \text{iff} \quad \mathcal{A} \in \mathcal{M}_1 + \mathcal{M}_2
\end{aligned}$$

One possibility of combining theories

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

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

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

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

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

Consequence: $\mathrm{Th}(\mathrm{Mod}(\mathcal{T}_1 \cup \mathcal{T}_2)) = \mathrm{Th}(\mathcal{M}_1 + \mathcal{M}_2)$

Example

1. Presburger arithmetic + UIF

$\mathsf{Th}(\mathbb{Z}_+) \cup UIF \qquad \Sigma = (\Omega, \Pi)$

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

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

2. The theory of reals + the theory of a monotone function f

$\mathsf{Th}(\mathbb{R}) \cup \mathsf{Mon}(f) \qquad \mathsf{Mon}(f) : \forall x, y (x \leq y \rightarrow f(x) \leq f(y))$

Models: $(A, +, *, f_A, \{\leq\})$, where

where $(A, +, *, \leq) \in \mathsf{Mod}(\mathsf{Th}(\mathbb{R}))$.

$(A, f_A, \leq) \models \mathsf{Mon}(f)$, i.e. $f_A : A \rightarrow A$ 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: $\Sigma_1 = (\Omega_1, \emptyset), \quad \Sigma_2 = (\{c/0, d/0\}, \emptyset), \quad c, d \not\in \Omega_1$

$$\mathcal{T}_1 = \{\exists x, y, z(x \not\approx y \ \wedge \ x \not\approx z \ \wedge \ y \not\approx z)\}$$

$$\mathcal{T}_2 = \{\forall x(x \approx c \ \vee \ x \approx d)\}$$

$$\mathcal{A} \in \mathrm{Mod}(\mathcal{T}_1) \quad \text{iff} \quad |U_{\mathcal{A}}| \geq 3.$$

$$\mathcal{B} \in \mathrm{Mod}(\mathcal{T}_2) \quad \text{iff} \quad |U_{\mathcal{B}}| \leq 2.$$

Combinations of theories

The combined decidability problem

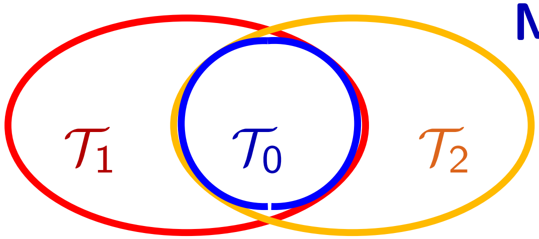
For $i = 1, 2$
- let $\mathcal{T}_i$ be a first-order theory in signature Σ_i
- assume the $\mathcal{T}_i$ ground satisfiability problem is decidable

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

Question:

Is the $\mathcal{T}_1 \bigoplus \mathcal{T}_2$ ground satisfiability problem decidable?

Goal: Modularity



Modular Reasoning

$\mathcal{T}_0$: Σ_0-theory.

$\mathcal{T}_i$: Σ_i-theory; $\mathcal{T}_0 \subseteq \mathcal{T}_i$ $\Sigma_0 \subseteq \Sigma_i$.

Example:

lists$(\mathbb{R}) \cup$ **arrays**$(\mathbb{R})$

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 Σ_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 $\mathcal{T}_i$ be a first-order theory in signature Σ_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 $\mathcal{T} \models \forall x(s \approx t)$

$\mathcal{A}$: theory of associativity

$\mathcal{G}$ finite set of ground equations
(presentation for semigroup
with undecidable word problem)

$\uparrow$

($\exists$ finitely-presented semigroup with
undecidable word problem [Matijasevic'67])

Word problem: decidable for $\mathcal{A}, \mathcal{G}$; undecidable for $\mathcal{A} \cup \mathcal{G}$

Combinations of theories

For $i = 1, 2$ • let $\mathcal{T}_i$ be a first-order theory in signature Σ_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 $i = 1, 2$
- let $\mathcal{T}_i$ be a first-order theory in signature Σ_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)

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 Γ_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: $\mathcal{T}_1, \mathcal{T}_2$ first-order theories with signatures Σ_1, Σ_2

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

P_i decision procedures for satisfiability of ground formulae w.r.t. $\mathcal{T}_i$

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

Task: Check whether ϕ 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

$\mathcal{R}$	theory of rationals	$\Sigma_{\mathcal{R}} = \{\leq, +, -, 0, 1\}$	$\approx$
$\mathcal{L}$	theory of lists	$\Sigma_{\mathcal{L}} = \{\mathsf{car}, \mathsf{cdr}, \mathsf{cons}\}$	$\approx$
$\mathcal{E}$	theory of equality (UIF)	Σ: free function and predicate symbols	$\approx$

Example

[Nelson & Oppen, 1979]

Theories

$\mathcal{R}$	theory of rationals	$\Sigma_{\mathcal{R}} = \{\leq, +, -, 0, 1\}$	$\approx$
$\mathcal{L}$	theory of lists	$\Sigma_{\mathcal{L}} = \{\text{car}, \text{cdr}, \text{cons}\}$	$\approx$
$\mathcal{E}$	theory of equality (UIF)	Σ: free function and predicate symbols	$\approx$

Problems:

1. $\mathcal{R} \cup \mathcal{L} \cup \mathcal{E} \models \forall x, y (x \leq y \wedge y \leq x + \text{car}(\text{cons}(0, x)) \wedge P(h(x) - h(y)) \rightarrow P(0))$
2. Is the following conjunction:

$$c \leq d \;\wedge\; d \leq c + \text{car}(\text{cons}(0, c)) \;\wedge\; P(h(c) - h(d)) \;\wedge\; \neg P(0)$$

satisfiable in $\mathcal{R} \cup \mathcal{L} \cup \mathcal{E}$?

An Example

	$\mathcal{R}$	$\mathcal{L}$	$\mathcal{E}$
Σ	$\{\leq, +, -, 0, 1\}$	$\{\mathsf{car}, \mathsf{cdr}, \mathsf{cons}\}$	$F \cup P$
Axioms	$x + 0 \approx x$	$\mathsf{car}(\mathsf{cons}(x, y)) \approx x$	
	$x - x \approx 0$	$\mathsf{cdr}(\mathsf{cons}(x, y)) \approx y$	
(univ.	$+$ is A, C	$\mathsf{at}(x) \vee \mathsf{cons}(\mathsf{car}(x), \mathsf{cdr}(x)) \approx x$	
quantif.)	$\leq$ is R, T, A	$\neg \mathsf{at}(\mathsf{cons}(x, y))$	
	$x \leq y \vee y \leq x$		
	$x \leq y \rightarrow x + z \leq y + z$		

Is the following conjunction:

$$c \leq d \ \wedge \ d \leq c + \mathsf{car}(\mathsf{cons}(0, c)) \ \wedge \ P(h(c) - h(d)) \ \wedge \ \neg P(0)$$

satisfiable in $\mathcal{R} \cup \mathcal{L} \cup \mathcal{E}$?

Step 1: Purification

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

Task: Find ϕ_1, ϕ_2 s.t. ϕ_i is a pure Σ_i-formula and $\phi_1 \wedge \phi_2$ equivalent with ϕ

$$f(s_1, \ldots, s_n) \approx g(t_1, \ldots, t_m) \quad \mapsto \quad u{\approx}f(s_1, \ldots, s_n) \wedge u{\approx}g(t_1, \ldots, t_m)$$

$$f(s_1, \ldots, s_n) \not\approx g(t_1, \ldots, t_m) \quad \mapsto \quad u{\approx}f(s_1, \ldots, s_n) \wedge v{\approx}g(t_1, \ldots, t_m) \wedge u \not\approx v$$

$$(\neg)P(\ldots, s_i, \ldots) \quad \mapsto \quad (\neg)P(\ldots, u, \ldots) \wedge u{\approx}s_i$$

$$(\neg)P(\ldots, s_i[t], \ldots) \quad \mapsto \quad (\neg)P(\ldots, s_i[t \mapsto u], \ldots) \wedge u{\approx}t$$

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

Termination: Obvious

Correctness: $\phi_1 \wedge \phi_2$ and ϕ equisatisfiable.

Step 1: Purification

$$c \leq d \;\wedge\; d \leq c + \mathsf{car}(\mathsf{cons}(0, c)) \;\wedge\; P(h(c) - h(d)) \;\wedge\; \neg P(0)$$

Step 1: Purification

$$c \leq d \;\wedge\; d \leq c + \underbrace{\mathrm{car}(\mathrm{cons}(0, c))}_{c_1} \;\wedge\; P(h(c) - h(d)) \;\wedge\; \neg P(0)$$

Step 1: Purification

$$c \leq d \ \wedge\ d \leq c + \underbrace{\mathsf{car}(\mathsf{cons}(0, c))}_{c_1} \ \wedge\ P(\underbrace{h(c) - h(d)}_{c_2}) \ \wedge\ \neg P(0)$$

Step 1: Purification

$$c \leq d \;\wedge\; d \leq c + \underbrace{\mathrm{car}(\mathrm{cons}(0, c))}_{c_1} \;\wedge\; P(\underbrace{\underbrace{h(c)}_{c_3} - \underbrace{h(d)}_{c_4}}_{c_2}) \;\wedge\; \neg P(\underbrace{0}_{c_5})$$

Step 1: Purification

$$c \leq d \;\wedge\; d \leq c + \underbrace{\mathrm{car}(\mathrm{cons}(0, c))}_{c_1} \;\wedge\; P(\underbrace{\underbrace{h(c)}_{c_3} - \underbrace{h(d)}_{c_4}}_{c_2}) \;\wedge\; \neg P(\underbrace{0}_{c_5})$$

$\mathcal{R}$	$\mathcal{L}$	$\mathcal{E}$
$c \leq d$	$c_1 \approx \mathrm{car}(\mathrm{cons}(c_5, c))$	$P(c_2)$
$d \leq c + c_1$		$\neg P(c_5)$
$c_2 \approx c_3 - c_4$		$c_3 \approx h(c)$
$c_5 \approx 0$		$c_4 \approx h(d)$

Step 1: Purification

$$c \leq d \ \wedge \ d \leq c + \underbrace{\text{car}(\text{cons}(0, c))}_{c_1} \ \wedge \ P(\underbrace{\underbrace{h(c)}_{c_3} - \underbrace{h(d)}_{c_4}}_{c_2}) \ \wedge \ \neg P(\underbrace{0}_{c_5})$$

$\mathcal{R}$	$\mathcal{L}$	$\mathcal{E}$
$c \leq d$	$c_1 \approx \text{car}(\text{cons}(c_5, c))$	$P(c_2)$
$d \leq c + c_1$		$\neg P(c_5)$
$c_2 \approx c_3 - c_4$		$c_3 \approx h(c)$
$c_5 \approx 0$		$c_4 \approx h(d)$
satisfiable	satisfiable	satisfiable

Step 2: Propagation

$$c \leq d \;\wedge\; d \leq c + \underbrace{\text{car}(\text{cons}(0, c))}_{c_1} \;\wedge\; P(\underbrace{\underbrace{h(c)}_{c_3} - \underbrace{h(d)}_{c_4}}_{c_2}) \;\wedge\; \neg P(\underbrace{\;0\;}_{c_5})$$

$\mathcal{R}$	$\mathcal{L}$	$\mathcal{E}$
$c \leq d$	$c_1 \approx \text{car}(\text{cons}(c_5, c))$	$P(c_2)$
$d \leq c + c_1$		$\neg P(c_5)$
$c_2 \approx c_3 - c_4$		$c_3 \approx h(c)$
$c_5 \approx 0$		$c_4 \approx h(d)$

deduce and propagate equalities between constants entailed by components

Step 2: Propagation

$$c \leq d \;\wedge\; d \leq c + \underbrace{\mathrm{car}(\mathrm{cons}(0, c))}_{c_1} \;\wedge\; P(\underbrace{\underbrace{h(c)}_{c_3} - \underbrace{h(d)}_{c_4}}_{c_2}) \;\wedge\; \neg P(\underbrace{0}_{c_5})$$

$\mathcal{R}$	$\mathcal{L}$	$\mathcal{E}$
$c \leq d$	$c_1 \approx \mathrm{car}(\mathrm{cons}(c_5, c))$	$P(c_2)$
$d \leq c + c_1$		$\neg P(c_5)$
$c_2 \approx c_3 - c_4$		$c_3 \approx h(c)$
$c_5 \approx 0$		$c_4 \approx h(d)$

$$c_1 \approx c_5$$

Step 2: Propagation

$$c \leq d \;\wedge\; d \leq c + \underbrace{\mathrm{car}(\mathrm{cons}(0, c))}_{c_1} \;\wedge\; P(\underbrace{\underbrace{h(c)}_{c_3} - \underbrace{h(d)}_{c_4}}_{c_2}) \;\wedge\; \neg P(\underbrace{0}_{c_5})$$

$\mathcal{R}$	$\mathcal{L}$	$\mathcal{E}$
$c \leq d$	$c_1 \approx \mathrm{car}(\mathrm{cons}(c_5, c))$	$P(c_2)$
$d \leq c + c_1$		$\neg P(c_5)$
$c_2 \approx c_3 - c_4$		$c_3 \approx h(c)$
$c_5 \approx 0$		$c_4 \approx h(d)$
$c_1 \approx c_5$	$c_1 \approx c_5$	
$c \approx d$		

Step 2: Propagation

$$c \leq d \;\wedge\; d \leq c + \underbrace{\mathrm{car}(\mathrm{cons}(0, c))}_{c_1} \;\wedge\; P(\underbrace{\underbrace{h(c)}_{c_3} - \underbrace{h(d)}_{c_4}}_{c_2}) \;\wedge\; \neg P(\underbrace{0}_{c_5})$$

$\mathcal{R}$	$\mathcal{L}$	$\mathcal{E}$
$c \leq d$	$c_1 \approx \mathrm{car}(\mathrm{cons}(c_5, c))$	$P(c_2)$
$d \leq c + c_1$		$\neg P(c_5)$
$c_2 \approx c_3 - c_4$		$c_3 \approx h(c)$
$c_5 \approx 0$		$c_4 \approx h(d)$
$c_1 \approx c_5$	$c_1 \approx c_5$	$c \approx d$
$c \approx d$		$c_3 \approx c_4$

Step 2: Propagation

$$c \leq d \;\wedge\; d \leq c + \underbrace{\text{car}(\text{cons}(0, c))}_{c_1} \;\wedge\; P(\underbrace{\underbrace{h(c)}_{c_3} - \underbrace{h(d)}_{c_4}}_{c_2}) \;\wedge\; \neg P(\underbrace{0}_{c_5})$$

$\mathcal{R}$	$\mathcal{L}$	$\mathcal{E}$
$c \leq d$	$c_1 \approx \text{car}(\text{cons}(c_5, c))$	$P(c_2)$
$d \leq c + c_1$		$\neg P(c_5)$
$c_2 \approx c_3 - c_4$		$c_3 \approx h(c)$
$c_5 \approx 0$		$c_4 \approx h(d)$
$c_1 \approx c_5$	$c_1 \approx c_5$	$c \approx d$
$c \approx d$		$c_3 \approx c_4$
$c_2 \approx c_5$		$\bot$

The Nelson-Oppen algorithm

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 ϕ_i is a pure Σ_i-formula and $\phi_1 \wedge \phi_2$ is equisatisfiable with ϕ.

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.

The Nelson-Oppen algorithm

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 ϕ_i is a pure Σ_i-formula and $\phi_1 \wedge \phi_2$ is equisatisfiable with ϕ.

not problematic; requires linear time

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; termination guaranteed
Sound: if inconsistency detected input unsatisfiable
Complete: under additional assumptions