Automated Reasoning II*

Uwe Waldmann

Summer Term 2024

Topics of the Course

Decision procedures:

equality (congruence closure),
algebraic theories,
combinations.

Satisfiability modulo theories (SMT):

CDCL(T),
dealing with universal quantification.

Superposition:

combining ordered resolution and completion,
optimizations,
integrating theories.

*This document contains the text of the lecture slides (almost verbatim) plus some additional information, mostly proofs of theorems that are presented on the blackboard during the course. It is not a full script and does not contain the examples and additional explanations given during the lecture. Moreover it should not be taken as an example how to write a research paper – neither stylistically nor typographically.

1 Decision Procedures

In general, validity (or unsatisfiability) of first-order formulas is undecidable.

To get decidability results, we have to impose restrictions on

- signatures,
- formulas,
- and/or algebras.

1.1 Theories and Fragments

So far, we have considered the validity or satisfiability of "unstructured" sets of formulas.

We will now split these sets of formulas into two parts: a theory (which we keep fixed) and a set of formulas that we consider relative to the theory.

A *first-order theory* $\mathcal{T}$ is defined by

its signature $\Sigma = (\Omega, \Pi)$

its axioms, that is, a set of closed Σ-formulas.

(We often use the same symbol $\mathcal{T}$ for a theory and its set of axioms.)

Note: This is the *syntactic view* of theories. There is also a *semantic view*, where one specifies a class of Σ-algebras $\mathcal{M}$ and considers $Th(\mathcal{M})$, that is, all closed Σ-formulas that hold in the algebras of $\mathcal{M}$.

A Σ-algebra that satisfies all axioms of $\mathcal{T}$ is called a *$\mathcal{T}$-algebra* (or *$\mathcal{T}$-interpretation*).

$\mathcal{T}$ is called *consistent* if there is at least one $\mathcal{T}$-algebra. (We will only consider consistent theories.)

We can define models, validity, satisfiability, entailment, equivalence, etc., relative to a theory $\mathcal{T}$:

A $\mathcal{T}$-algebra that is a model of a Σ-formula F is also called a $\mathcal{T}$-model of F.

A Σ-formula F is called $\mathcal{T}$-valid, if $\mathcal{A}, \beta \models F$ for all $\mathcal{T}$-algebras $\mathcal{A}$ and assignments β.

A Σ-formula F is called $\mathcal{T}$-satisfiable, if $\mathcal{A}, \beta \models F$ for some $\mathcal{T}$-algebra and assignment β (and otherwise $\mathcal{T}$-unsatisfiable).

($\mathcal{T}$-satisfiability of sets of formulas, $\mathcal{T}$-entailment, $\mathcal{T}$-equivalence: analogously.)

A *fragment* is some syntactically restricted class of Σ-formulas.

Typical restriction: only certain quantifier prefixes are permitted.

1.2 Equality

Theory of equality:

Signature: arbitrary

Axioms: none
(but the equality predicate $\approx$ has a fixed interpretation)

Alternatively:

Signature contains a binary predicate symbol $\sim$ instead of the built-in $\approx$

Axioms: reflexivity, symmetry, transitivity, congruence for $\sim$

In general, satisfiability of first-order formulas w. r. t. equality is undecidable.

However, we will show that it is decidable for *ground* first-order formulas.

Note: It suffices to consider conjunctions of literals. Arbitrary ground formulas can be converted into DNF; a formula in DNF is satisfiable if and only if one of its conjunctions is satisfiable.

Note that our problem can be written in several ways:

An equational clause
$\forall\vec{x}\,(A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_k)$ is $\mathcal{T}$-valid

iff

$\exists\vec{x}\,(\neg A_1 \wedge \ldots \wedge \neg A_n \wedge B_1 \wedge \ldots \wedge B_k)$ is $\mathcal{T}$-unsatisfiable

iff

the Skolemized (ground!) formula
$(\neg A_1 \wedge \ldots \wedge \neg A_n \wedge B_1 \wedge \ldots \wedge B_k)\{\vec{x} \mapsto \vec{c}\}$ is $\mathcal{T}$-unsatisfiable

iff

$(A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_k)\{\vec{x} \mapsto \vec{c}\}$ is $\mathcal{T}$-valid

Other names:

The theory is also known as *EUF* (equality with uninterpreted function symbols).

The decision procedures for the ground fragment are called *congruence closure* algorithms.

Congruence Closure

Goal: check (un-)satisfiability of a ground conjunction

$$u_1 \approx v_1 \land \ldots \land u_n \approx v_n \land \neg\, s_1 \approx t_1 \land \ldots \land \neg\, s_k \approx t_k$$

Idea:

transform $E = \{u_1 \approx v_1, \ldots, u_n \approx v_n\}$ into an equivalent convergent TRS R and check whether $s_i{\downarrow_R} = t_i{\downarrow_R}$.

if $s_i{\downarrow_R} = t_i{\downarrow_R}$ for some i:
$s_i{\downarrow_R} = t_i{\downarrow_R} \;\Leftrightarrow\; s_i \leftrightarrow_E^* t_i \;\Leftrightarrow\; E \models s_i \approx t_i \;\Rightarrow\;$ unsat.

if $s_i{\downarrow_R} = t_i{\downarrow_R}$ for no i:
$\mathrm{T}_\Sigma(X)/R = \mathrm{T}_\Sigma(X)/E$ is a model of the conjunction $\;\Rightarrow\;$ sat.

In principle, one could use Knuth-Bendix completion to convert E into an equivalent convergent TRS R.

If done properly (see exercises), Knuth-Bendix completion terminates for ground inputs.

However, for the ground case, one can optimize the general procedure.

First step:

Flatten terms: Introduce new constant symbols $c_1, c_2, \ldots$ for all subterms:

$$g(a, h(h(b))) \approx h(a)$$

is replaced by

$$a \approx c_1 \land b \approx c_2 \land h(c_2) \approx c_3 \land h(c_3) \approx c_4 \land g(c_1, c_4) \approx c_5 \land h(c_1) \approx c_6 \land c_5 \approx c_6$$

Result: only two kinds of equations left.

D-equations: $f(c_{i_1}, \ldots, c_{i_n}) \approx c_{i_0}$ for $f/n \in \Omega$, $n \geq 0$.
C-equations: $c_i \approx c_j$.

$\Rightarrow$ efficient indexing (e. g., using hash tables),
obvious termination for D-equations.

Inference Rules

The congruence closure algorithm is presented as a set of inference rules working on a set of equations E and a set of rules R: $E_0, R_0 \vdash E_1, R_1 \vdash E_2, R_2 \vdash \ldots$

At the beginning, $E = E_0$ is the set of C-equations and $R = R_0$ is the set of D-equations oriented left-to-right. At the end, E should be empty; then R is the result.

Notation: The formula $s \mathrel{\dot{\approx}} t$ denotes either $s \approx t$ or $t \approx s$.

Simplify:

$$\frac{E \cup \{c \mathrel{\dot{\approx}} c'\}, \quad R \cup \{c \to c''\}}{E \cup \{c'' \mathrel{\dot{\approx}} c'\}, \quad R \cup \{c \to c''\}}$$

Delete:

$$\frac{E \cup \{c \approx c\}, \quad R}{E, \quad R}$$

Orient:

$$\frac{E \cup \{c \mathrel{\dot{\approx}} c'\}, \quad R}{E, \quad R \cup \{c \to c'\}} \qquad \text{if } c \succ c'$$

Collapse:

$$\frac{E, \quad R \cup \{t[c]_p \to c',\, c \to c''\}}{E, \quad R \cup \{t[c'']_p \to c',\, c \to c''\}} \qquad \text{if } p \neq \varepsilon$$

Deduce:

$$\frac{E, \quad R \cup \{t \to c,\, t \to c'\}}{E \cup \{c \approx c'\}, \quad R \cup \{t \to c\}}$$

Note: for ground rewrite rules, critical pair computation does not involve substitution. Therefore, every critical pair computation can be replaced by a simplification, either using Deduce or Collapse.

Theorem 1.1 *Let E_0 be a finite set of C-equations, let R_0 be a finite set of D-equations oriented left-to-right w.r.t. $\succ$, and let $\succ$ be a total ordering on constants. Then the inference system terminates with a final state (E_n, R_n) where $E_n = \emptyset$, R_n is terminating and confluent, and $\approx_{E_0 \cup R_0}$ equals $\approx_{R_n}$.*

Strategy

The inference rules are applied according to the following strategy:

(1) If there is an equation in E, use Simplify as long as possible for this equation, then use either Delete or Orient. Repeat until E is empty.

(2) If Collapse is applicable, apply it, if now Deduce is applicable, apply it as well. Repeat until Collapse is no longer applicable.

(3) If E is non-empty, go to (1), otherwise return R.

Implementation

Instead of fixing the ordering $\succ$ in advance, it is preferable to define it on the fly during the algorithm:

If we orient an equation $c \approx c'$ between two constant symbols, we try to make that constant symbol larger that occurs less often in $R \;\Rightarrow\;$ fewer Collapse steps.

Additionally:

Use various index data structures so that all the required operations can be performed efficiently.

Use a union-find data structure to represent the equivalence classes encoded by the C-rules.

Average runtime for an implementation using hash tables: $O(m \log m)$, where m is the number of edges in the graph representation of the initial C and D-equations.

One Small Problem

The inference rules are sound in the usual sense: The conclusions are entailed by the premises, so every $\mathcal{T}$-model of the premises is a $\mathcal{T}$-model of the conclusions.

For the initial flattening, however, we get a weaker result: We have to *extend* the $\mathcal{T}$-models of the original equations to obtain models of the flattened equations. That is, we get a new algebra with the same universe as the old one, with the same interpretations for old functions and predicate symbols, but with appropriately chosen interpretations for the new constants.

Consequently, the relations $\approx_E$ and $\approx_R$ for the original E and the final R are not the same. For instance, $c_3 \approx_E c_7$ does not hold, but $c_3 \approx_R c_7$ may hold.

On the other hand, the model extension preserves the universe and the interpretations for old symbols. Therefore, if s and t are terms over the old symbols, we have $s \approx_E t$ iff $s \approx_R t$.

This is sufficient for our purposes: The terms s_i and t_i that we want to normalize using R do not contain new symbols.

Other Predicate Symbols

If the initial ground conjunction contains also non-equational literals $[\neg]\ P(t_1, \ldots, t_n)$, treat these like equational literals $[\neg]\ P(t_1, \ldots, t_n) \approx \mathit{true}$. Then use the same algorithm as before.

History

Congruence closure algorithms have been published, among others, by Shostak (1978). by Nelson and Oppen (1980), and by Downey, Sethi and Tarjan (1980).

Kapur (1997) showed that Shostak's algorithm can be described as a completion procedure.

Bachmair and Tiwari (2000) did this also for the Nelson/Oppen and the Downey/Sethi/Tarjan algorithm.

The algorithm presented here is the Downey/Sethi/Tarjan algorithm in the presentation of Bachmair and Tiwari.

Literature

Leo Bachmair, Ashish Tiwari: Abstract Congruence Closure and Specializations. Proc. CADE-17, 2000, pp 64–78, LNCS 1831, Springer.

Peter J. Downey, Ravi Sethi, Robert E. Tarjan: Variations on the Common Subexpression Problem. Journal of the ACM, 27(4):758–771, 1980.

Deepak Kapur: Shostak's congruence closure as completion. Proc. 8th RTA, 1997, pp. 23–37, LNCS 1232, Springer.

Greg Nelson, Derek C. Oppen: Fast Decision Procedures Based on Congruence Closure. Journal of the ACM, 27(2):356–364, 1980.

Robert E. Shostak: An algorithm for reasoning about equality. Communications of the ACM, 21(7):583–585, 1978.

1.3 Linear Rational Arithmetic

There are several ways to define *linear rational arithmetic.*

We need at least the following signature: $\Sigma = (\{0/0, 1/0, +/2\}, \{</2\})$ and the predefined binary predicate $\approx$.

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

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

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

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

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

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

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

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

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

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: $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{Q}^n$, $\mathbb{R}^n$, ...

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 mult_n, 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.

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 literal $x \sim t[\vec{y}]$ or $0 \sim t[\vec{y}]$, where $\sim\ \in \{<, >, \leq, \geq, \approx, \not\approx\}$ and $t[\vec{y}]$ has the form $\sum_i q_i \cdot y_i + q_0$.

In other words, we can either eliminate x completely or isolate in on one side of the literal, and we can replace every negative ordering literal by a positive one.

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

We first consider existentially quantified conjunctions of atoms.

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

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

is equivalent to

$$F\{x \mapsto t[\vec{y}]\}$$

If x occurs only in inequations, then

$$\begin{aligned} \exists x \, \Big(& \textstyle\bigwedge_i x < s_i(\vec{y}) \ \wedge \ \bigwedge_j x \leq t_j(\vec{y}) \\ & \wedge \ \textstyle\bigwedge_k x > u_k(\vec{y}) \ \wedge \ \bigwedge_l x \geq v_l(\vec{y}) \ \wedge \ \bigwedge_m 0 \sim_m w_m(\vec{y}) \Big) \end{aligned}$$

is equivalent to

$$\begin{aligned} & \textstyle\bigwedge_i \bigwedge_k s_i(\vec{y}) > u_k(\vec{y}) \ \wedge \ \bigwedge_j \bigwedge_k t_j(\vec{y}) > u_k(\vec{y}) \\ & \quad \wedge \ \textstyle\bigwedge_i \bigwedge_l s_i(\vec{y}) > v_l(\vec{y}) \ \wedge \ \bigwedge_j \bigwedge_l t_j(\vec{y}) \geq v_l(\vec{y}) \\ & \quad \wedge \ \textstyle\bigwedge_m 0 \sim_m w_m(\vec{y}) \end{aligned}$$

Proof: ($\Rightarrow$) by transitivity;
($\Leftarrow$) take $\frac{1}{2}(\min\{s_i, t_j\} + \max\{u_k, v_l\})$ as a witness.

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 $\exists$.

Consequence: every closed formula over the signature of ODAGs is ODAG-equivalent to either $\top$ or $\bot$.

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

Consequence: 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 $\exists \dots \exists$:

naive implementation produces a doubly exponential number of inequations, therefore needs $O(n^{2^m})$ runtime (the number of *necessary* inequations grows only exponentially, though).

m quantifiers $\exists\forall\exists\forall \dots \exists$:

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

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).

Let $F(x, \vec{y})$ be a *positive* boolean combination of linear (in-)equations $x \sim_i s_i(\vec{y})$ and $0 \sim_j s'_j(\vec{y})$ with $\sim_i, \sim_j \in \{\approx, \not\approx, <, \leq, >, \geq\}$, that is, a formula built from linear (in-) equations, $\wedge$ and $\vee$ (but without $\neg$).

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

$$\exists x\, F(x, \vec{y}) \quad \models\!\!\mid \quad \bigvee_{t \in T} F(x, \vec{y})\, \{x \mapsto t\}$$

In other words: We want to replace the infinite disjunction $\exists x$ by a finite disjunction.

If we keep the values of the variables $\vec{y}$ fixed, then we can consider F as a function $F : x \mapsto F(x, \vec{y})$ from $\mathbb{Q}$ to $\{0, 1\}$.

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

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

F is a piecewise constant function; more precisely, the set of all x with $F(x, \vec{y}) = 1$ is a finite union of intervals. (The union may be empty, the individual intervals may be finite or infinite and open or closed.)

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

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

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

(ii) or there is some point where the value of $F(x, \vec{y})$ 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.

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

For case (ii), we observe that $F(x, \vec{y})$ 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.)

$x \leq s_i(\vec{y})$ and $x < s_i(\vec{y})$ do not switch from 0 to 1 when x grows.

$x \geq s_i(\vec{y})$ and $x \approx s_i(\vec{y})$ switch from 0 to 1 at $s_i(\vec{y})$
$\Rightarrow s_i(\vec{y})$ is a test point.

$x > s_i(\vec{y})$ and $x \not\approx s_i(\vec{y})$ switch from 0 to 1 "right after" $s_i(\vec{y})$
$\Rightarrow s_i(\vec{y}) + \varepsilon$ (for some $0 < \varepsilon < \delta(\vec{y})$) is a test point.

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

$$\begin{aligned} T := \{r(\vec{y})\} &\cup \{\, s_i(\vec{y}) \quad\quad\ \mid \sim_i \in \{\geq, =\}\,\} \\ &\cup \{\, s_i(\vec{y}) + \varepsilon \mid \sim_i \in \{>, \neq\}\,\}. \end{aligned}$$

is a set of test points.

Problem:
We don't know how small $r(\vec{y})$ has to be for case (i), and we don't know $\delta(\vec{y})$ for case (ii).

Idea:
We consider the limits for $r \to -\infty$ and for $\varepsilon \searrow 0$, that is, we redefine

$$\begin{aligned} T := \{-\infty\} &\cup \{\, s_i(\vec{y}) \quad\quad\ \mid \sim_i \in \{\geq, =\}\,\} \\ &\cup \{\, s_i(\vec{y}) + \varepsilon \mid \sim_i \in \{>, \neq\}\,\}. \end{aligned}$$

How can we eliminate the infinitesimals ∞ and ε when we substitute elements of T for x?

Virtual substitution:

$$
\begin{aligned}
&(x < s(\vec{y}))\,\{x \mapsto -\infty\} := \lim_{r\to-\infty}(r < s(\vec{y})) = \top\\
&(x \leq s(\vec{y}))\,\{x \mapsto -\infty\} := \lim_{r\to-\infty}(r \leq s(\vec{y})) = \top\\
&(x > s(\vec{y}))\,\{x \mapsto -\infty\} := \lim_{r\to-\infty}(r > s(\vec{y})) = \bot\\
&(x \geq s(\vec{y}))\,\{x \mapsto -\infty\} := \lim_{r\to-\infty}(r \geq s(\vec{y})) = \bot\\
&(x \approx s(\vec{y}))\,\{x \mapsto -\infty\} := \lim_{r\to-\infty}(r \approx s(\vec{y})) = \bot\\
&(x \not\approx s(\vec{y}))\,\{x \mapsto -\infty\} := \lim_{r\to-\infty}(r \not\approx s(\vec{y})) = \top
\end{aligned}
$$

$$
\begin{aligned}
&(x < s(\vec{y}))\,\{x \mapsto u+\varepsilon\} := \lim_{\varepsilon\searrow 0}(u+\varepsilon < s(\vec{y})) = (u < s(\vec{y}))\\
&(x \leq s(\vec{y}))\,\{x \mapsto u+\varepsilon\} := \lim_{\varepsilon\searrow 0}(u+\varepsilon \leq s(\vec{y})) = (u < s(\vec{y}))\\
&(x > s(\vec{y}))\,\{x \mapsto u+\varepsilon\} := \lim_{\varepsilon\searrow 0}(u+\varepsilon > s(\vec{y})) = (u \geq s(\vec{y}))\\
&(x \geq s(\vec{y}))\,\{x \mapsto u+\varepsilon\} := \lim_{\varepsilon\searrow 0}(u+\varepsilon \geq s(\vec{y})) = (u \geq s(\vec{y}))\\
&(x \approx s(\vec{y}))\,\{x \mapsto u+\varepsilon\} := \lim_{\varepsilon\searrow 0}(u+\varepsilon \approx s(\vec{y})) = \bot\\
&(x \not\approx s(\vec{y}))\,\{x \mapsto u+\varepsilon\} := \lim_{\varepsilon\searrow 0}(u+\varepsilon \not\approx s(\vec{y})) = \top
\end{aligned}
$$

We have traversed the real axis from $-\infty$ to $+\infty$. Alternatively, we can traverse it from $+\infty$ to $-\infty$. In this case, the test points are

$$
\begin{aligned}
T' := \{+\infty\} &\cup \{\, s_i(\vec{y}) \qquad\;\; \mid \sim_i \in \{\leq, =\}\,\}\\
&\cup \{\, s_i(\vec{y}) - \varepsilon \mid \sim_i \in \{<, \neq\}\,\}.
\end{aligned}
$$

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 formulas $\forall x\, F$, 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 one plus the number of atoms (one plus half of the number of atoms, if there are only ordering literals), the formula size grows quadratically; therefore $O(n^2)$ runtime.

Multiple quantifiers of the same kind:

$$\begin{aligned}
& \exists x_2\, \exists x_1.\ F(x_1, x_2, \vec{y}) \\
\rightsquigarrow\quad & \exists x_2.\ \big(\textstyle\bigvee_{t_1 \in T_1} F(x_1, x_2, \vec{y})\,\{x_1 \mapsto t_1\}\big) \\
\rightsquigarrow\quad & \textstyle\bigvee_{t_1 \in T_1} (\exists x_2.\ F(x_1, x_2, \vec{y})\,\{x_1 \mapsto t_1\}) \\
\rightsquigarrow\quad & \textstyle\bigvee_{t_1 \in T_1} \bigvee_{t_2 \in T_2} (F(x_1, x_2, \vec{y})\,\{x_1 \mapsto t_1\}\,\{x_2 \mapsto t_2\})
\end{aligned}$$

m quantifiers $\exists \ldots \exists$ or $\forall \ldots \forall$:

formula size is multiplied by n in each step, therefore $O(n^{m+1})$ runtime.

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

doubly exponential runtime.

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

Literature

Andreas Dolzmann: Algorithmic Strategies for Applicable Real Qunantifier Elimination. PhD thesis, Universität Passau, 2000.

Jean-Baptiste Joseph Fourier: Solution d'une question particulière du calcul des inégalités. Nouveau Bulletin des Sciences par la Société philomahique de Paris, 1826.

F. Levi: Arithmetische Gesetze im Gebiete discreter Gruppen. Rendiconti del Circolo Matematico di Palermo, 35:225–236, 1913.

Rüdiger Loos, Volker Weispfenning: Applying Linear Quantifier Elimination. The Computer Journal, 36(5):450–462, 1993.

1.4 Existentially-quantified LRA

So far, we have considered formulas that may contain free, existentially quantified, and universally quantified variables.

For the special case of conjunction of linear inequations in which *all* variables are existentially quantified, there are more efficient methods available.

Main idea: reduce satisfiability problem to optimization problem.

Linear Optimization

Goal:

Solve a linear optimization (also called: linear programming) problem for given numbers $a_{ij}, b_i, c_j \in \mathbb{R}$:

$$\begin{array}{ll} \text{maximize} & \sum_{1 \le j \le n} c_j x_j \\ \text{for} & \bigwedge_{1 \le i \le m} \sum_{1 \le j \le n} a_{ij} x_j \le b_i \end{array}$$

or in vectorial notation:

$$\begin{array}{ll} \text{maximize} & \vec{c}^{\top} \vec{x} \\ \text{for} & A\vec{x} \le \vec{b} \end{array}$$

Simplex algorithm:

Developed independently by Kantorovich (1939), Dantzig (1948).

Polynomial-time average-case complexity; worst-case time complexity is exponential, though.

Interior point methods:

First algorithm by Karmarkar (1984).

Polynomial-time worst-case complexity (but large constants).

In practice: no clear winner.

Implementations:

GLPK (GNU Linear Programming Kit),

Gurobi.

Main idea of Simplex:

$A\vec{x} \leq \vec{b}$ describes a convex polyhedron.

Pick one vertex of the polyhedron,
then follow the edges of the polyhedron towards an optimal solution.

By convexity, the local optimum found in this way is also a global optimum.

Details: see special lecture on optimization.

Using an optimization procedure for checking satisfiability:

Goal: Check whether $A\vec{x} \leq \vec{b}$ is satisfiable.

To use the Simplex method, we have to transform the original (possibly empty) polyhedron into another polyhedron that is non-empty and for which we know one initial vertex.

Every real number can be written as the difference of two non-negative real numbers. Use this idea to convert $A\vec{x} \leq \vec{b}$ into an equisatisfiable inequation system $\vec{y} \geq \vec{0}$, $B\vec{y} \leq \vec{b}$ for new variables $\vec{y}$.

Multiply those inequations of the inequation system $B\vec{y} \leq \vec{b}$ in which the number on the right-hand side is negative by -1. We obtain two inequation systems $D_1\vec{y} \leq \vec{g}_1$, $D_2\vec{y} \geq \vec{g}_2$, such that $\vec{g}_1 \geq \vec{0}$, $\vec{g}_2 > 0$.

Now solve

$$\begin{aligned}
&\text{maximize} \;\; \vec{1}^\top(D_2\vec{y} - \vec{z}) \\
&\text{for} \;\; \vec{y}, \vec{z} \geq \vec{0} \\
&\qquad D_1\vec{y} \leq \vec{g}_1 \\
&\qquad D_2\vec{y} - \vec{z} \leq \vec{g}_2
\end{aligned}$$

where $\vec{z}$ is a vector of new variables with the same size as $\vec{g}_2$.

Observation 1: $\vec{0}$ is a vertex of the polyhedron of this optimization problem.

Observation 2: The maximum is $\vec{1}^\top\vec{g}_2$ if and only if $\vec{y} \geq \vec{0}$, $D_1\vec{y} \leq \vec{g}_1$, $D_2\vec{y} \geq \vec{g}_2$ has a solution.

($\Rightarrow$): If $\vec{1}^\top(D_2\vec{y} - \vec{z}) = \vec{1}^\top\vec{g}_2$ for some $\vec{y}, \vec{z}$ satisfying $D_2\vec{y} - \vec{z} \leq \vec{g}_2$, then $D_2\vec{y} - \vec{z} = \vec{g}_2$, hence $D_2\vec{y} = \vec{g}_2 + \vec{z} \geq \vec{g}_2$.

($\Leftarrow$): $\vec{1}^\top(D_2\vec{y} - \vec{z})$ can never be larger than $\vec{1}^\top\vec{g}_2$. If $\vec{y} \geq \vec{0}$, $D_1\vec{y} \leq \vec{g}_1$, $D_2\vec{y} \geq \vec{g}_2$ has a solution, choose $\vec{z} = D_2\vec{y} - \vec{g}_2$; then $\vec{1}^\top(D_2\vec{y} - \vec{z}) = \vec{1}^\top\vec{g}_2$.

A Simplex variant:

Transform the satisfiability problem into the form

$$\begin{aligned}&A\vec{x} = \vec{0}\\&\vec{l} \leq \vec{x} \leq \vec{u}\end{aligned}$$

(where l_i may be $-\infty$ and u_i may be $+\infty$).

Relation to optimization problem is obscured.

But: More efficient if one needs an incremental decision procedure, where inequations may be added and retracted (Dutertre and de Moura 2006).

1.5 Non-linear Real Arithmetic

Tarski (1951): Quantifier elimination is possible for *non-linear* real arithmetic (or more generally, for real-closed fields). His algorithm had non-elementary complexity, however.

An improved algorithm by Collins (1975) (with further improvements by Hong) has doubly exponential complexity: Cylindrical algebraic decomposition (CAD).

Implementation: QEPCAD.

Cylindrical Algebraic Decomposition

Given: First-order formula over atoms of the form $f_i(\vec{x}) \sim 0$, where the f_i are polynomials over variables $\vec{x}$.

Goal: Decompose $\mathbb{R}^n$ into a finite number of regions such that all polynomials have invariant sign on every region X:

$$\begin{aligned}\forall i\ (\ &\forall \vec{x} \in X.\ f_i(\vec{x}) < 0\\ &\lor \forall \vec{x} \in X.\ f_i(\vec{x}) = 0\\ &\lor \forall \vec{x} \in X.\ f_i(\vec{x}) > 0\ )\end{aligned}$$

Note: Implementation needs exact arithmetic using algebraic numbers (i. e., zeroes of univariate polynomials with integer coefficients).

1.6 Real Arithmetic incl. Transcendental Functions

Real arithmetic with exp/log: decidability unknown.

Real arithmetic with trigonometric functions: undecidable

The following formula holds exactly if $x \in \mathbb{Z}$:

$$\exists y\ (\sin(y) = 0 \wedge 3 < y \wedge y < 4 \wedge \sin(x \cdot y) = 0)$$

(note that necessarily $y = \pi$).

Consequence: Peano arithmetic (which is undecidable) can be encoded in real arithmetic with trigonometric functions.

However, real arithmetic with transcendental functions is decidable for formulas that are *stable under perturbations*, i. e., whose truth value does not change if numeric constants are modified by some sufficiently small ε.

Example:

Stable under perturbations: $\exists x\ x^2 \leq 5$

Not stable under perturbations: $\exists x\ x^2 \leq 0$
(Formula is true, but if we subtract an arbitrarily small $\varepsilon > 0$ from the right-hand side, it becomes false.)

Unsatisfactory from a mathematical point of view, but sufficient for engineering applications (where stability under perturbations is necessary anyhow).

Approach:

Interval arithmetic + interval bisection if necessary (Ratschan).

Sound for general formulas; complete for formulas that are stable under perturbations; may loop forever if the formula is not stable under perturbations.

1.7 Linear Integer Arithmetic

Linear integer arithmetic = Presburger arithmetic.

Decidable (Presburger, 1929), but quantifier elimination is only possible if additional divisibility operators are present:

$\exists x\ (y = 2x)$ is equivalent to $\text{divides}(2, y)$ but not to any quantifier-free formula over the base signature.

Cooper (1972): Quantifier elimination procedure, triple exponential for arbitrarily quantified formulas.

The Omega Test

Omega test (Pugh, 1991): variant of Fourier–Motzkin for conjunctions of (in-)equations in linear integer arithmetic.

Idea:

- Perform easy transformations, e. g.:
  $3x + 6y \leq 8 \;\mapsto\; 3x + 6y \leq 6 \;\mapsto\; x + 2y \leq 2$
  $3x + 6y = 8 \;\mapsto\; \bot$
  (since $3x + 6y$ must be divisible by 3).

- Eliminate equations
  (easy, if one coefficient is 1; tricky otherwise).

- If only inequations are left:
  no real solutions $\rightarrow$ unsatisfiable for $\mathbb{Z}$
  "sufficiently many" real solutions $\rightarrow$ satisfiable for $\mathbb{Z}$
  otherwise: branch

What does "sufficiently many" mean?

Consider inequations $ax \leq s$ and $bx \geq t$ with $a, b \in \mathbb{N}^{>0}$ and polynomials s, t.

If these inequations have real solutions, the interval of solutions ranges from $\frac{1}{b}t$ to $\frac{1}{a}s$.

The longest possible interval of this kind that does not contain any integer number ranges from $i + \frac{1}{b}$ to $i + 1 - \frac{1}{a}$ for some $i \in \mathbb{Z}$; it has the length $1 - \frac{1}{a} - \frac{1}{b}$.

Consequence:

If $\frac{1}{a}s > \frac{1}{b}t + (1 - \frac{1}{a} - \frac{1}{b})$, or equivalently, $bs \geq at + ab - a - b + 1$ is satisfiable, then the original problem must have integer solutions.

It remains to consider the case that $bs \geq at$ is satisfiable (hence there are real solutions) but $bs \geq at + ab - a - b + 1$ is not (hence the interval of real solutions need not contain an integer).

In the latter case, $bs \leq at + ab - a - b$ holds, hence for every solution of the original problem:

$t \leq bx \leq \frac{b}{a}s \leq t + (b - 1 - \frac{b}{a})$

and if x is an integer, $t \leq bx \leq t + \left\lfloor b - 1 - \frac{b}{a} \right\rfloor$

$\Rightarrow$ Branch non-deterministically:
Add one of the equations $bx = t + i$ for $i \in \{0, \dots, \lfloor b - 1 - \frac{b}{a} \rfloor\}$.

Alternatively, if $b > a$:
Add one of the equations $ax = s - i$ for $i \in \{0, \dots, \lfloor a - 1 - \frac{a}{b} \rfloor\}$.

Note: Efficiency depends highly on the size of coefficients. In applications from program verification, there is almost always some variable with a very small coefficient. If all coefficients are large, the branching step gets expensive.

Branch-and-Cut

Alternative approach: Reduce satisfiability problem to optimization problem (like Simplex). ILP, MILP: (mixed) integer linear programming.

Two basic approaches:

Branching: If the simplex algorithm finds a solution with $x = 2.7$, add the inequation $x \leq 2$ or the inequation $x \geq 3$.

Cutting planes: Derive an inequation that holds for all real solutions, then round it to obtain an inequation that holds for all integer solutions, but not for the real solution found previously.

Example:

Given:
$$\begin{aligned} 2x - 3y &\leq 1 \\ 2x + 3y &\leq 5 \\ -5x - 4y &\leq -7 \end{aligned}$$

Simplex finds an extremal solution $x = \frac{3}{2}$, $y = \frac{2}{3}$.

From the first two inequations, we see that $4x \leq 6$, hence $x \leq \frac{3}{2}$. If $x \in \mathbb{Z}$, we conclude $x = \lfloor x \rfloor \leq \lfloor \frac{3}{2} \rfloor = 1$.

$\Rightarrow$ Add the inequation $x \leq 1$, which holds for all integer solutions, but cuts off the solution $(\frac{3}{2}, \frac{2}{3})$.

In practice:

Use both: Alternate between branching and cutting steps.
Better performance than the individual approaches.

1.8 Difference Logic

Difference Logic (DL):

Fragment of linear rational or integer arithmetic.

Formulas: conjunctions of atoms $x - y < c$ or $x - y \leq c$,
$x, y \in X$, $c \in \mathbb{Q}$ (or $c \in \mathbb{Z}$).

One special variable x_0 whose value is fixed to 0 is permitted;
this allows to express atoms like $x < 3$ in the form $x - x_0 < 3$.

Solving difference logic:

Let F be a conjunction in DL.
For simplicity: only non-strict inequalities.

Define a weighted graph G:

Vertices V: Variables in F.

Edges E: $x - y \leq c \ \rightsquigarrow$ edge (x, y) with weight c.

Theorem: F is unsatisfiable iff G has a negative cycle.

Can be checked in $O(|V| \cdot |E|)$ using the Bellman-Ford algorithm.

1.9 C-Arithmetic

In languages like C: Bounded integer arithmetic (modulo 2^n), in device drivers also combined with bitwise operations.

Bit-Blasting (encode everything as boolean circuits, use CDCL):

Naive encoding: possible, but often too inefficient.

If combined with over-/underapproximation techniques (Bryant, Kroening, et al.): successful.

1.10 Decision Procedures for Data Structures

There are decision procedures for, e. g.,

Arrays (read, write)

Lists (car, cdr, cons)

Sets or multisets with cardinalities

Bitvectors

Note: There are usually restrictions on quantifications. Unrestricted universal quantification can lead to undecidability.

Literature: Further Decision Procedures

Aaron R. Bradley, Zohar Manna: The Calculus of Computation. Springer, 2007.

Aaron R. Bradley, Zohar Manna, Henny B. Sipma: What's decidable about arrays? Verification, Model Checking, and Abstract Interpretation (VMCAI), LNCS 3855, pp. 427-442, Springer, 2006.

Randal E. Bryant, Daniel Kroening, Joël Ouaknine, Sanjit A. Seshia, Ofer Strichman, Bryan Brady: Deciding bit-vector arithmetic with abstraction. 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07), LNCS 4424, pp. 358–372, Springer, 2007.

George E. Collins: Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. 2nd. GI Conf. Automata Theory and Formal Languages, LNCS 33, pp. 134–183, Springer, 1975.

D. C. Cooper: Theorem Proving in Arithmetic Without Multiplication. Machine Intelligence, vol. 7, pp. 91–99. American Elsevier, New York, 1972.

George B. Dantzig: Linear Programming and Extensions. Princeton Univ. Press, 1963.

L. V. Kantorovich: Mathematical Methods in the Organization and Planning of Production. Publication House of the Leningrad State University, 1939. Translated in Management Science, 6:366–422, 1960.

Narendra Karmarkar: A New Polynomial Time Algorithm for Linear Programming. Combinatorica, 4(4):373–395, 1984.

Daniel Kroening, Ofer Strichman: Decision Procedures – An Algorithmic Point of View. Springer, 2008.

Mojżesz Presburger: Über der Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. Comptes Rendus Premier Congrès des Mathématiciens des Pays Slaves, Warsaw, pp. 92-101, 1929.

William Pugh: The Omega Test: a fast and practical integer programming algorithm for dependence analysis. Comm. of the ACM, 35(8):102-114, 1992.

Stefan Ratschan: Approximate Quantified Constraint Solving by Cylindrical Box Decomposition. Reliable Computing, 8(1):21–42, 2002.

Alfred Tarski: A Decision Method for Elementary Algebra and Geometry. Univ. of California Press, Berkeley, 1951.

1.11 Combining Decision Procedures

Problem:

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be first-order theories over the signatures Σ_1 and Σ_2.

Assume that we have decision procedures for the satisfiability of existentially quantified formulas (or the validity of universally quantified formulas) w. r. t. $\mathcal{T}_1$ and $\mathcal{T}_2$.

Can we combine them to get a decision procedure for the satisfiability of existentially quantified formulas w. r. t. $\mathcal{T}_1 \cup \mathcal{T}_2$?

General assumption:

Σ_1 and Σ_2 are disjoint.

The only symbol shared by $\mathcal{T}_1$ and $\mathcal{T}_2$ is built-in equality.

We consider only conjunctions of literals.

For general formulas, convert to DNF first and consider each conjunction individually.

Abstraction

To be able to use the individual decision procedures, we have to transform the original formula in such a way that each atom contains only symbols of one of the signatures (plus variables).

This process is known as *variable abstraction* or *purification*.

We apply the following rule as long as possible:

$$\frac{\exists \vec{x}\,(F[t])}{\exists \vec{x}, y\,(F[y] \land t \approx y)}$$

if the top symbol of t belongs to Σ_i and t occurs in F directly below a Σ_j-symbol or in a (positive or negative) equation $s \approx t$ where the top symbol of s belongs to Σ_j $(i \neq j)$, and if y is a new variable.

It is easy to see that the original and the purified formula are equivalent.

Stable Infiniteness

Problem:

Even if the Σ_1-formula F_1 and the Σ_2-formula F_2 do not share any symbols (not even variables), and if F_1 is $\mathcal{T}_1$-satisfiable and F_2 is $\mathcal{T}_2$-satisfiable, we cannot conclude that $F_1 \wedge F_2$ is $(\mathcal{T}_1 \cup \mathcal{T}_2)$-satisfiable.

Example:

Consider

$\mathcal{T}_1 = \{\forall x, y, z\, (x \approx y \;\vee\; x \approx z \;\vee\; y \approx z)\}$

and

$\mathcal{T}_2 = \{\exists x, y, z\, (x \not\approx y \;\wedge\; x \not\approx z \;\wedge\; y \not\approx z)\}$.

All $\mathcal{T}_1$-models have at most two elements, and all $\mathcal{T}_2$-models have at least three elements.

Since $\mathcal{T}_1 \cup \mathcal{T}_2$ is contradictory, there are no $(\mathcal{T}_1 \cup \mathcal{T}_2)$-satisfiable formulas.

To ensure that $\mathcal{T}_1$-models and $\mathcal{T}_2$-models can be combined to $(\mathcal{T}_1 \cup \mathcal{T}_2)$-models, we require that both $\mathcal{T}_1$ and $\mathcal{T}_2$ are stably infinite.

A first-order theory $\mathcal{T}$ is called *stably infinite*, if every existentially quantified formula that has a $\mathcal{T}$-model has also a $\mathcal{T}$-model with a (countably) infinite universe.

Note: By the Löwenheim–Skolem theorem, “countable” is redundant here.

Shared Variables

Even if $\exists\vec{x}\, F_1$ is $\mathcal{T}_1$-satisfiable and $\exists\vec{x}\, F_2$ is $\mathcal{T}_2$-satisfiable, it can happen that $\exists\vec{x}\,(F_1 \wedge F_2)$ is not $(\mathcal{T}_1 \cup \mathcal{T}_2)$-satisfiable, for instance because the shared variables x and y must be equal in all $\mathcal{T}_1$-models of $\exists\vec{x}\, F_1$ and different in all $\mathcal{T}_2$-models of $\exists\vec{x}\, F_2$.

Example:

Consider

$F_1 = (x + (-y) \approx 0)$,

and

$F_2 = (f(x) \not\approx f(y))$

where $\mathcal{T}_1$ is linear rational arithmetic and $\mathcal{T}_2$ is EUF.

We must exchange information about shared variables to detect the contradiction.

The Nelson–Oppen Algorithm (Non-deterministic Version)

Suppose that $\exists \vec{x}\, F$ is a purified conjunction of Σ_1 and Σ_2-literals.

Let F_1 be the conjunction of all literals of F that do not contain Σ_2-symbols; let F_2 be the conjunction of all literals of F that do not contain Σ_1-symbols. (Equations between variables are in both F_1 and F_2.)

The Nelson–Oppen algorithm starts with the pair F_1, F_2 and applies the following inference rules.

Unsat:

$$\frac{F_1, F_2}{\bot}$$

if $\exists \vec{x}\, F_i$ is unsatisfiable w. r. t. $\mathcal{T}_i$ for some i.

Branch:

$$\frac{F_1, F_2}{F_1 \wedge (x \approx y), F_2 \wedge (x \approx y) \quad | \quad F_1 \wedge (x \not\approx y), F_2 \wedge (x \not\approx y)}$$

if x and y are two different variables appearing in both F_1 and F_2 such that neither $x \approx y$ nor $x \not\approx y$ occurs in both F_1 and F_2

"|" means non-deterministic (backtracking!) branching of the derivation into two subderivations. Derivations are therefore trees. All branches need to be reduced until termination.

Clearly, all derivation paths are finite since there are only finitely many *shared variables* in F_1 and F_2, therefore the procedure represented by the rules is terminating.

We call a constraint configuration to which no rule applies *irreducible*.

Theorem 1.2 (Soundness) *If "Branch" can be applied to F_1, F_2, then $\exists \vec{x}\,(F_1 \wedge F_2)$ is satisfiable in $\mathcal{T}_1 \cup \mathcal{T}_2$ if and only if one of the successor configurations of F_1, F_2 is satisfiable in $\mathcal{T}_1 \cup \mathcal{T}_2$.*

Corollary 1.3 *If all paths in a derivation tree from F_1, F_2 end in $\perp$, then $\exists \vec{x}\,(F_1 \wedge F_2)$ is unsatisfiable in $\mathcal{T}_1 \cup \mathcal{T}_2$.*

For completeness we need to show that if one branch in a derivation terminates with an irreducible configuration F_1, F_2 (different from $\perp$), then $\exists \vec{x}\,(F_1 \wedge F_2)$ (and, thus, the initial formula of the derivation) is satisfiable in the combined theory.

As $\exists \vec{x}\,(F_1 \wedge F_2)$ is irreducible by "Unsat", the two formulas are satisfiable in their respective component theories, that is, we have $\mathcal{T}_i$-models $\mathcal{A}_i$ of $\exists \vec{x}\, F_i$ for $i \in \{1, 2\}$. We are left with combining the models into a single one that is both a model of the combined theory and of the combined formula. These constructions are called *amalgamations*.

Let F be a Σ_i-formula and let S be a set of variables of F. F is called *compatible* with an equivalence $\sim$ on S if the formula

$$\exists \vec{z} \left(F \wedge \bigwedge_{x,y \in S,\, x \sim y} x \approx y \wedge \bigwedge_{x,y \in S,\, x \not\sim y} x \not\approx y\right) \tag{1}$$

is $\mathcal{T}_i$-satisfiable whenever F is $\mathcal{T}_i$-satisfiable. This expresses that F does not contradict equalities between the variables in S as given by $\sim$.

The formula $\displaystyle\bigwedge_{x,y \in S,\, x \sim y} x \approx y \wedge \bigwedge_{x,y \in S,\, x \not\sim y} x \not\approx y$ is called an *arrangement* of S.

Proposition 1.4 *If F_1, F_2 is a pair of conjunctions over $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, that is irreducible by "Branch", then both F_1 and F_2 are compatible with some equivalence $\sim$ on the shared variables S of F_1 and F_2.*

Proof. If F_1, F_2 is irreducible by the branching rule, then for each pair of shared variables x and y, both F_1 and F_2 contain either $x \approx y$ or $x \not\approx y$. Choose $\sim$ to be the equivalence given by all (positive) variable equations between shared variables that are contained in F_1.

Let $\Sigma = (\Omega, \Pi)$; let $\Sigma' = (\Omega', \Pi')$ with $\Omega' \subseteq \Omega$ and $\Pi' \subseteq \Pi$ be a subsignature of Σ.

Let $\mathcal{A}$ be a Σ-algebra. Then the *reduct* $\mathcal{A}|_{\Sigma'}$ is the Σ'-algebra $\mathcal{A}'$ with

$U_{\mathcal{A}'} = U_{\mathcal{A}}$,
$f_{\mathcal{A}'} = f_{\mathcal{A}}$ for all $f \in \Omega'$, and
$P_{\mathcal{A}'} = P_{\mathcal{A}}$ for all $P \in \Pi'$.

Lemma 1.5 (Amalgamation Lemma) *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two stably infinite theories over disjoint signatures Σ_1 and Σ_2. Furthermore let F_1, F_2 be a pair of conjunctions of literals over $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, both compatible with some equivalence $\sim$ on the shared variables of F_1 and F_2. Then $F_1 \wedge F_2$ is $(\mathcal{T}_1 \cup \mathcal{T}_2)$-satisfiable if and only if each F_i is $\mathcal{T}_i$-satisfiable.*

Proof. The "only if" part is obvious.

For the "if" part, assume that each of the F_i is $\mathcal{T}_i$-satisfiable. That is, there exist models $\mathcal{A}_i$ of $\mathcal{T}_i$ and variable assignments β_i such that $\mathcal{A}_i, \beta_i \models F_i$. As the F_i are compatible with an equivalence $\sim$ on their shared variables, we may assume that the β_i also satisfy the extended conjunctions in (1) with S the set of shared variables. In particular, whenever we have two shared variables x and y, $\beta_1(x) = \beta_1(y)$ if and only if $\beta_2(x) = \beta_2(y)$. Since the theories are stably infinite we may additionally assume that the $\mathcal{A}_i$ have countably infinite universes, hence there are bijections ρ_i from the domain of $\mathcal{A}_i$ to $\mathbb{N}$ such that $\rho_1(\beta_1(x))) = \rho_2(\beta_2(x))$ for each shared variable x. Now define $\mathcal{A}$ to be the algebra having $\mathbb{N}$ as its domain; for f or P in Σ_i define $f_{\mathcal{A}}(n_1, \ldots, n_k) = \rho_i(f_{\mathcal{A}_i}(\rho_i^{-1}(n_1), \ldots, \rho_i^{-1}(n_k)))$ and $P_{\mathcal{A}}(n_1, \ldots, n_k) \Leftrightarrow P_{\mathcal{A}_i}(\rho_i^{-1}(n_1), \ldots, \rho_i^{-1}(n_k))$. Define $\beta(x) = \rho_i(\beta_i(x))$ if x is a variable occurring in F_i. By construction of the ρ_i this definition is independent of the choice of i. Clearly $\mathcal{A}|_{\Sigma_i}, \beta \models F_i$, for $i = 1, 2$, hence $\mathcal{A}, \beta \models F_1 \wedge F_2$. Moreover, the reducts $\mathcal{A}|_{\Sigma_i}$ are isomorphic (via ρ_i) to $\mathcal{A}_i$ and thus are models of $\mathcal{T}_i$, so that $\mathcal{A}$ is a model of $\mathcal{T}_1 \cup \mathcal{T}_2$ as required.

Theorem 1.6 *The non-deterministic Nelson–Oppen algorithm is terminating and complete for deciding satisfiability of pure conjunctions of literals F_1 and F_2 over $\mathcal{T}_1 \cup \mathcal{T}_2$ for signature-disjoint, stably infinite theories $\mathcal{T}_1$ and $\mathcal{T}_2$.*

Proof. Suppose that F_1, F_2 is irreducible by the inference rules of the Nelson–Oppen algorithm. Applying the amalgamation lemma in combination with Prop. 1.4 we infer that F_1, F_2 is satisfiable w. r. t. $\mathcal{T}_1 \cup \mathcal{T}_2$.

Convexity

The number of possible equivalences of shared variables grows superexponentially with the number of shared variables, so enumerating all possible equivalences non-deterministically is going to be inefficient.

A much faster variant of the Nelson–Oppen algorithm exists for convex theories.

A first-order theory $\mathcal{T}$ is called *convex w. r. t. equations*, if for every conjunction Γ of Σ-equations and non-equational Σ-literals and for all Σ-equations A_i ($1 \leq i \leq n$), whenever $\mathcal{T} \models \forall\vec{x}\,(\Gamma \rightarrow A_1 \vee \ldots \vee A_n)$, then there exists some index j such that $\mathcal{T} \models \forall\vec{x}\,(\Gamma \rightarrow A_j)$.

Theorem 1.7 *If a first-order theory $\mathcal{T}$ is convex w. r. t. equations and has no trivial models (i. e., models with only one element), then $\mathcal{T}$ is stably infinite.*

Proof. We shall prove the contrapositive of the statement. Suppose $\mathcal{T}$ is not stably infinite. Then there exists a $\mathcal{T}$-satisfiable conjunction of literals $\exists\vec{x}\,F$ that has only finite $\mathcal{T}$-models. As $\mathcal{T}$ is a first-order theory and first-order logic is compact, all $\mathcal{T}$-models of $\exists\vec{x}\,F$ are bounded in cardinality by some number m.

Let $y_1, \ldots, y_{m+1}$ be fresh variables not occurring in F. Then the formula $F_0 = \exists\vec{x}\,F \wedge \exists y_1 \ldots y_{m+1} \bigwedge_{1\leq i<j\leq m+1} y_i \not\approx y_j$ is $\mathcal{T}$-unsatisfiable since it expresses the fact that $\exists\vec{x}\,F$ has a model with more than m elements. Therefore, $\mathcal{T} \models \neg F_0$.

We can write F in the form $F^+ \wedge F^-$, where F^- contains the negative equational literals in F and F^+ contains the rest. Then $\mathcal{T} \models \neg F_0$ can be written as $\mathcal{T} \models \forall\vec{x}\,\vec{y}\,(\neg F^+ \vee \neg F^- \vee \bigvee_{1\leq i<j\leq m+1} y_i \approx y_j)$, or equivalently, $\mathcal{T} \models \forall\vec{x},\vec{y}\,(F^+ \rightarrow (\neg F^- \vee \bigvee_{1\leq i<j\leq m+1} y_i \approx y_j))$. Note that $\neg F^-$ is a disjunction of positive equational literals.

Assume that $\mathcal{T} \models \forall\vec{x},\vec{y}\,(F^+ \rightarrow A)$ for some literal A of $\neg F^- \vee \bigvee_{1\leq i<j\leq m+1} y_i \approx y_j$. If A is a literal of $\neg F^-$, then $\mathcal{T} \models \forall\vec{x},\vec{y}\,(F^+ \rightarrow A) \models \forall\vec{x},\vec{y}\,(F^+ \rightarrow \neg F^-) \models \forall\vec{x},\vec{y}\neg F$, which cannot hold since F is $\mathcal{T}$-satisfiable. Otherwise A is a literal $y_i \approx y_j$, then $\mathcal{T} \models \forall\vec{x},\vec{y}\,(F^+ \rightarrow y_i \approx y_j)$. This cannot hold either: Note that $\exists\vec{x}\,F$ and thus $\exists\vec{x}\,F^+$ is $\mathcal{T}$-satisfiable. So let $\mathcal{A}$ be some $\mathcal{T}$-model of $\exists\vec{x}\,F^+$. By assumption, $\mathcal{A}$ is not a trivial model, therefore there is an $\mathcal{A}$-assignment β to $\vec{x},\vec{y}$ that satisfies F^+ and maps y_i and y_j to two arbitrary different elements. Consequently, $\mathcal{A},\beta \not\models (F^+ \rightarrow y_i \approx y_j)$.

Lemma 1.8 *Suppose $\mathcal{T}$ is convex, F a conjunction of literals, and S a subset of its variables. Let, for any pair of variables x_i and x_j in S, $x_i \sim x_j$ if and only if $\mathcal{T} \models \forall\vec{x}\,(F \to x_i \approx x_j)$. Then F is compatible with $\sim$.*

Proof. We show that with this choice of $\sim$ the constraint (1), that is,

$$\exists\vec{z}\Big(F \wedge \bigwedge_{x,y\in S,\ x\sim y} x \approx y \wedge \bigwedge_{x,y\in S,\ x\not\sim y} x \not\approx y\Big)$$

is $\mathcal{T}$-satisfiable whenever F is. Suppose, to the contrary, that F is $\mathcal{T}$-satisfiable but (1) is not, that is,

$$\mathcal{T} \models \forall\vec{z}\Big(F \to \bigvee_{x,y\in S,\ x\sim y} x \not\approx y \ \vee \bigvee_{x,y\in S,\ x\not\sim y} x \approx y\Big)$$

or, equivalently,

$$\mathcal{T} \models \forall\vec{z}\Big(F^+ \wedge \bigwedge_{x,y\in S,\ x\sim y} x \approx y \ \to\ \neg F^- \vee \bigvee_{x,y\in S,\ x\not\sim y} x \approx y\Big).$$

where F^- contains the negative equational literals in F and F^+ contains the rest. By convexity of $\mathcal{T}$, the antecedent implies one of the equations of the succedent.

Suppose that this equation A comes from $\neg F^-$. Then

$$\mathcal{T} \models \forall\vec{z}\Big(F^+ \wedge \bigwedge_{x,y\in S,\ x\sim y} x \approx y \ \to\ A)$$

and therefore

$$\mathcal{T} \models \forall\vec{z}\Big(F^+ \wedge \bigwedge_{x,y\in S,\ x\sim y} x \approx y \ \to\ \neg F^-)$$

which means

$$\mathcal{T} \models \forall\vec{z}\Big((F^+ \wedge F^-) \wedge \bigwedge_{x,y\in S,\ x\sim y} x \approx y \ \to\ \bot)$$

which cannot hold since $F = (F^+ \wedge F^-)$ is $\mathcal{T}$-satisfiable and entails the equations $x \approx y$ with $x \sim y$.

So the equation A must come from the last part of the disjunction. In other words, there exists a pair of different variables x' and y' in S such that $x' \not\sim y'$ and

$$\mathcal{T} \models \forall\vec{z}\Big(F^+ \wedge \bigwedge_{x,y\in S,\ x\sim y} x \approx y \ \to\ x' \approx y'\Big).$$

Since

$$\mathcal{T} \models \forall\vec{z}\Big(F \to \bigwedge_{x,y\in S,\ x\sim y} x \approx y\Big),$$

we derive $\mathcal{T} \models \forall\vec{z}\Big(F \ \to\ x' \approx y'\Big)$, which is impossible.

The Nelson–Oppen Algorithm (Deterministic Version for Convex Theories)

Unsat:

$$\frac{F_1, F_2}{\bot}$$

if $\exists\vec{x}\, F_i$ is unsatisfiable w. r. t. $\mathcal{T}_i$ for some i.

Propagate:

$$\frac{F_1, F_2}{F_1 \wedge (x \approx y), F_2 \wedge (x \approx y)}$$

if x and y are two different variables appearing in both F_1 and F_2 such that
$\mathcal{T}_1 \models \forall\vec{x}\,(F_1 \to x \approx y)$ and $\mathcal{T}_2 \not\models \forall\vec{x}\,(F_2 \to x \approx y)$
or $\mathcal{T}_2 \models \forall\vec{x}\,(F_2 \to x \approx y)$ and $\mathcal{T}_1 \not\models \forall\vec{x}\,(F_1 \to x \approx y)$.

Theorem 1.9 *If $\mathcal{T}_1$ and $\mathcal{T}_2$ are signature-disjoint theories that are convex w. r. t. equations and have no trivial models, then the deterministic Nelson–Oppen algorithm is terminating, sound and complete for deciding satisfiability of pure conjunctions of literals F_1 and F_2 over $\mathcal{T}_1 \cup \mathcal{T}_2$.*

Proof. Termination and soundness are obvious: there are only finitely many different equations that can be added, and each of them is entailed by given formulas.

For completeness, we have to show that every configuration that is irreducible by "Unsat" and "Propagate" is satisfiable w. r. t.. $\mathcal{T}_1 \cup \mathcal{T}_2$: Let F_1, F_2 be such a configuration. As it is irreducible by "Propagate", we have, for every equation $x \approx y$ between shared variables, $\mathcal{T}_1 \models \forall\vec{x}\,(F_1 \to x \approx y)$ if and only if $\mathcal{T}_2 \models \forall\vec{x}\,(F_2 \to x \approx y)$. Consequently, F_1 and F_2 are compatible with the same equivalence on the shared variables of F_1 and F_2. Moreover, each of the formulas F_i is $\mathcal{T}_i$-satisfiable, and since convexity implies stable infiniteness, F_i has a $\mathcal{T}_i$-model with a countably infinite universe. Hence, by the amalgamation lemma, $F_1 \wedge F_2$ is $(\mathcal{T}_1 \cup \mathcal{T}_2)$-satisfiable.

Corollary 1.10 *The deterministic Nelson–Oppen algorithm for convex theories requires at most $O(n^3)$ calls to the individual decision procedures for the component theories, where n is the number of shared variables.*

Iterating Nelson–Oppen

The Nelson–Oppen combination procedures can be iterated to work with more than two component theories by virtue of the following observations where signature disjointness is assumed:

Theorem 1.11 *If $\mathcal{T}_1$ and $\mathcal{T}_2$ are stably infinite, then so is $\mathcal{T}_1 \cup \mathcal{T}_2$.*

Proof. The non-deterministic Nelson–Oppen algorithm is sound and complete for $\mathcal{T}_1 \cup \mathcal{T}_2$, that is, an existentially quantified conjunction F over $\Sigma_1 \cup \Sigma_2$ is satisfiable if and only if in every derivation from the purified form of F there exists a branch leading to some irreducible constraint F_1, F_2 entailing F. The amalgamation lemma 1.5 constructs a model with a countably infinite universe for F from the models of F_1 and F_2.

Lemma 1.12 *A first-order theory $\mathcal{T}$ is convex w. r. t. equations if and only if for every conjunction Γ of Σ-equations and non-equational Σ-literals and for all equations $x_i \approx x_i'$ $(1 \leq i \leq n)$, whenever $\mathcal{T} \models \forall\vec{x}\,(\Gamma \to x_1 \approx x_1' \vee \ldots \vee x_n \approx x_n')$, then there exists some index j such that $\mathcal{T} \models \forall\vec{x}\,(\Gamma \to x_j \approx x_j')$.*

Lemma 1.13 *Let $\mathcal{T}$ be a first-order theory that is convex w. r. t. equations. Let F is a conjunction of literals; let F^- be the conjunction of all negative equational literals in F and let F^+ be the conjunction of all remaining literals in F. If $\mathcal{T} \models \forall\vec{x}\,(F \to x \approx y)$, then $\exists\vec{x}\,F$ is $\mathcal{T}$-unsatisfiable or $\mathcal{T} \models \forall\vec{x}\,(F^+ \to x \approx y)$.*

Proof. $\mathcal{T} \models \forall\vec{x}\,(F \to x \approx y)$ is equivalent to $\mathcal{T} \models \forall\vec{x}\,(F^+ \to (\neg F^- \vee x \approx y))$. By convexity of $\mathcal{T}$ we know that $\mathcal{T} \models \forall\vec{x}\,(F^+ \to x \approx y)$ or $\mathcal{T} \models \forall\vec{x}\,(F^+ \to A)$ for some literal $\neg A$ in F^-. In the latter case, $\exists\vec{x}\,(F^+ \wedge \neg A)$ is $\mathcal{T}$-unsatisfiable; hence $\exists\vec{x}\,F$, that is, $\exists\vec{x}\,(F^+ \wedge F^-)$ is $\mathcal{T}$-unsatisfiable as well.

Theorem 1.14 *If $\mathcal{T}_1$ and $\mathcal{T}_2$ are convex w. r. t. equations and do not have trivial models, then so is $\mathcal{T}_1 \cup \mathcal{T}_2$.*

Proof. Suppose that $\mathcal{T}_1$ and $\mathcal{T}_2$ are convex w. r. t. equations and do not have trivial models. Then clearly $\mathcal{T}_1 \cup \mathcal{T}_2$ cannot have trivial models either, since any such model would also be a trivial model of $\mathcal{T}_1$ and $\mathcal{T}_2$.

Assume furthermore that $\mathcal{T} \models \forall\vec{x}\,(\Gamma \to x_1 \approx x_1' \vee \ldots \vee x_n \approx x_n')$ for some conjunction Γ of $(\Sigma_1 \cup \Sigma_2)$-equations and non-equational $(\Sigma_1 \cup \Sigma_2)$-literals. Then $\exists\vec{x}\,(\Gamma \wedge x_1 \not\approx x_1' \wedge \ldots \wedge x_n \not\approx x_n')$ is $\mathcal{T}$-unsatisfiable, and we can detect this by some run of the deterministic Nelson–Oppen algorithm starting with $\exists\vec{x},\vec{y}\,(\Gamma_1 \wedge \Gamma_2 \wedge x_1 \not\approx x_1' \wedge \ldots \wedge x_n \not\approx x_n')$, where $\Gamma_1 \wedge \Gamma_2$ is the result of purifying Γ. This run consists of a sequence of "Propagate" steps followed by a final "Unsat" step, and without loss of generality, we use the "Propagate" rule only if "Unsat" cannot be applied. Consequently, whenever we add an equation $x \approx y$ that is entailed by F_1 w. r. t. $\mathcal{T}_1$ or by F_2 w. r. t. $\mathcal{T}_2$, then it is by Lemma 1.13 already entailed by the positive and the non-equational literals in F_1 or F_2. Furthermore, due to the convexity of $\mathcal{T}_1$ and $\mathcal{T}_2$, the final "Unsat" step depends on at most one negative equational literal in F_1 or F_2. We can therefore construct a similar Nelson–Oppen derivation that starts with only the positive and the non-equational literals in

Γ_1 and Γ_2, plus at most one negative equational literal that may be needed for the "Unsat" step. If a negative equational literal is needed, it is one of the $x_j \not\approx x'_j$; then $\exists\vec{x}\,(\Gamma \wedge x_j \not\approx x'_j)$ is $\mathcal{T}$-unsatisfiable and $\forall\vec{x}\,(\Gamma \rightarrow x_j \approx x'_j)$ is $\mathcal{T}$-valid; if no negative equational literal is needed at all, then $\exists\vec{x}\,\Gamma$ is $\mathcal{T}$-unsatisfiable, so $\forall\vec{x}\,(\Gamma \rightarrow x_j \approx x'_j)$ is $\mathcal{T}$-valid for every j.

Extensions

Many-sorted logics:

$read/2$ becomes $read : array \times int \rightarrow data.$
$write/3$ becomes $write : array \times int \times data \rightarrow array.$
Variables: $x : data$

Only one declaration per function/predicate/variable symbol.
All terms, atoms, substitutions must be well-sorted.

Algebras:

Instead of universe $U_{\mathcal{A}}$, one set per sort: $array_{\mathcal{A}}$, $int_{\mathcal{A}}$.

Interpretations of function and predicate symbols correspond to their declarations: $read_{\mathcal{A}} : array_{\mathcal{A}} \times int_{\mathcal{A}} \rightarrow data_{\mathcal{A}}$

If we consider combinations of theories with shared sorts but disjoint function and predicate symbols, then we get essentially the same combination results as before.

However, stable infiniteness and/or convexity are only required for the shared sorts.

Non-stably infinite theories:

If we impose stronger conditions on one theory, we can relax the conditions on the other one.

For instance, EUF can be combined with any other theory; stable infiniteness is not required.

E.g.: Strongly polite theories, shiny theories, flexible theories.

Strong Politeness

A theory $\mathcal{T}$ is called *smooth*, if every quantifier-free formula that has a $\mathcal{T}$-model with some (finite or infinite) cardinality κ_0 has also $\mathcal{T}$-models with cardinality κ for every $\kappa \geq \kappa_0$.

A theory $\mathcal{T}$ is called *finitely witnessable*, if there is a computable function wit that maps every quantifier-free formula F to a quantifier-free formula G such that

1. F and $\exists \vec{w}\, G$ are $\mathcal{T}$-equivalent, where $\vec{w} = \text{var}(G) \setminus \text{var}(F)$,
2. if $G \wedge \Delta$ is $\mathcal{T}$-satisfiable for some arrangement Δ, then there is a $\mathcal{T}$-model $\mathcal{A}$ and an assignment β such that $\mathcal{A}, \beta \models (G \wedge \Delta)$ and $U_{\mathcal{A}} = \{\, \beta(x) \mid x \in \text{var}(G \wedge \Delta) \,\}$

A theory $\mathcal{T}$ is called *strongly polite*, if it is smooth and finitely witnessable.

Theorem 1.15 (Barrett & Jovanović) *We can combine two theories $\mathcal{T}_1$ and $\mathcal{T}_2$ if one of them if strongly polite.*

Again, in the many-sorted case, smoothness and finite witnessability must hold for all the shared sorts.

Non-disjoint combinations:

Have to ensure that both decision procedures interpret shared symbols in a compatible way.

Some results, e. g. by Ghilardi, using strong model theoretical conditions on the theories.

Another Combination Method

Shostak's method:

Applicable to combinations of EUF and *solvable* theories.

A Σ-theory $\mathcal{T}$ is called *solvable*, if there exists an effectively computable function *solve* such that, for any $\mathcal{T}$-equation $s \approx t$:

(A) $\mathit{solve}(s \approx t) = \bot$ if and only if $\mathcal{T} \models \forall \vec{x}\, (s \not\approx t)$;

(B) $\mathit{solve}(s \approx t) = \emptyset$ if and only if $\mathcal{T} \models \forall \vec{x}\, (s \approx t)$; and otherwise

(C) $\mathit{solve}(s \approx t) = \{x_1 \approx u_1, \ldots, x_n \approx u_n\}$, where

- the x_i are pairwise different variables occurring in $s \approx t$;

– the x_i do not occur in the u_j; and

– $\mathcal{T} \models \forall\vec{x}\,((s \approx t) \leftrightarrow \exists\vec{y}\,(x_1 \approx u_1 \wedge \ldots \wedge x_n \approx u_n))$, where $\vec{y}$ are the variables occurring in one of the u_j but not in $s \approx t$, and $\vec{x} \cap \vec{y} = \emptyset$.

Additionally useful (but not required):

A canonizer, that is, a function that simplifies terms by computing some unique normal form

Main idea of the procedure:

If $s \approx t$ is a positive equation and $\mathit{solve}(s \approx t) = \{x_1 \approx u_1, \ldots, x_n \approx u_n\}$, replace $s \approx t$ by $x_1 \approx u_1 \wedge \ldots \wedge x_n \approx u_n$ and use these equations to eliminate the x_i elsewhere.

Practical problem:

Solvability is a rather restrictive condition.

Literature

Harald Ganzinger: Shostak Light. Automated Deduction, CADE-18, LNCS 2392, pp 332–346, Springer, 2002.

Silvio Ghilardi: Model Theoretic Methods in Combined Constraint Satisfiability. Journal of Automated Reasoning, 33(3–4):221–249, 2005.

Dejan Jovanović and Clark Barrett: Polite theories revisited. Logic for Programming, Artificial Intelligence, and Reasoning, LNCS 6397, pp. 402–416, Springer, 2010.

Greg Nelson, Derek C. Oppen: Simplification by Cooperating Decision Procedures. ACM Transactions on Programming Languages and Systems, 1(2):245–257, 1979.

Robert E. Shostak: Deciding Combinations of Theories. Journal of the ACM, 31(1):1–12, 1984.