Uniform Description of Efficient Decision Procedures Using Extended Signatures

Ashish Tiwari

Tiwari@csl.sri.com

Computer Science Laboratory
SRI International
Menlo Park CA 94025

Part of the work was done jointly with Prof. Leo Bachmair, SUNY Stony Brook

Formal Logic

Formal logic provides languages for precise formulation of

- specification of hardware and software designs, protocols,
- correctness properties of programs,
- queries for database search, . . .

Inference mechanisms help to

- prove correctness of specifications,
- answer search queries, . . .

Formal Logic

Applications usually involve many different "theories".

For example, in a typical program correctness application, if,

$$\begin{array}{rcl} \textbf{select}(v, i) & : & \text{Select the } i\text{-th element of array } v \\ \textbf{store}(v, i, e) & : & \text{Store } e \text{ as the } i\text{-th element of array } v \end{array}$$

then, we may obtain:

$$\begin{array}{rcl} \phi & : & \textbf{select}(\textbf{store}(v, i, e+1), i) \approx [1 + \textbf{select}(v, i)] \wedge \\ & & (e < \textbf{select}(v, i)) \end{array}$$

Theorem Proving

General theorem provers are systems that automatically infer new facts from known ones.

How to handle applications involving different “theories”?
Add axioms corresponding to various useful theories into a general-purpose deduction system.

But...theorem provers are:

$+$: Powerful (expressiveness) $-$: Unpredictable and slow

and adding more axioms does *not* help!

Decision Procedures

Decision procedures are specialized procedures designed for subclass of formulas possibly from a particular domain.

$-$: Limited in use $\qquad$ $+$: Fast, predictable

Examples include

- algorithms in a computer algebra system
- Presburger arithmetic
- congruence closure
- model checking, ...

The Combination Problem

$T_1,\ T_2$	:	FO theories over signatures Σ_1, Σ_2.
$T = T_1 \cup T_2$	:	FO theory over $\Sigma_1 \cup \Sigma_2$.
$\Pi(T)$	:	Satisfiability problem of quantifier-free formulas in theory T.

Given decision procedures for $\Pi(T_1)$ and $\Pi(T_2)$, can we get one for $\Pi(T)$?

Variable Abstraction

Terms	$t[s]$	$\mapsto$	$t[x]$	$\wedge$	$x \approx s$
Formulae	ϕ	$\mapsto$	ϕ_1	$\wedge$	ϕ_2
	$(\Sigma_1 \cup \Sigma_2)$		$(\Sigma_1 \cup \mathcal{V})$		$(\Sigma_2 \cup \mathcal{V})$

Example:

$$\begin{array}{lcl}
\phi & : & \textbf{select}(\textbf{store}(v, i, e+1), i) \approx [1 + \textbf{select}(v, i)] \wedge \\
 & & (e < \textbf{select}(v, i)) \\
\phi_1 & : & \textbf{select}(\textbf{store}(v, i, z_0), i) \approx z_1 \ \wedge (\textbf{select}(v, i) \approx z_2) \\
\phi_2 & : & (z_0 \approx e + 1) \ \wedge \ (1 + z_2 \approx z_1) \ \wedge \ (e < z_2)
\end{array}$$

Congruence Closure: Problem

Σ : Signature containing constants and function symbols

$\Sigma = \{f_1, f_2, f_3, \ldots, f_n\}$

ϕ : $t_1 \approx s_1 \;\wedge\; t_2 \approx s_2 \;\wedge \cdots \wedge\; t_k \approx s_k \;\wedge$

$(t'_1 \not\approx s'_1 \;\wedge \cdots \wedge\; t'_l \neq s'_l)$

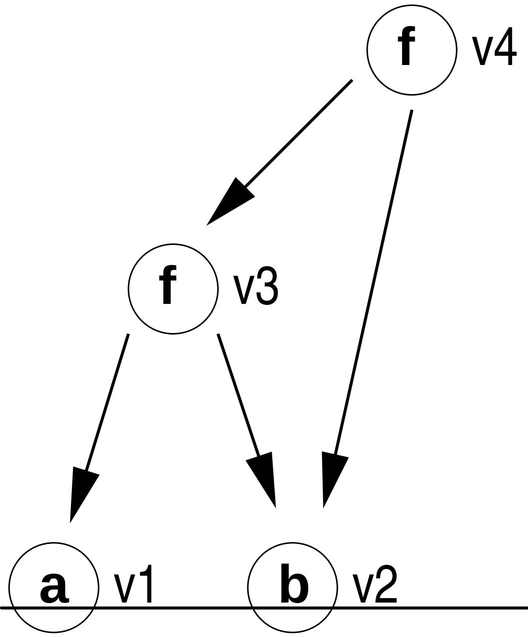
t_i, s_i, t'_i, s'_i are terms over Σ

Is ϕ satisfiable?

Compute congruence closure of the equations and check for each inequality.

Example of Congruence Closure

If $\mathcal{E} = \{f(f(a,b),b) \approx b,\ f(a,b) \approx a\}$, then the terms are represented by the DAG below, and:



$$G_{\mathcal{E}} = \{(v2, v3), (v1, v4)\}.$$

The final congruence closure is

$$\{\{v1, v2, v3, v4\}\}.$$

Congruence Closure: A Different Look

T_i : Theory of equality over $\Sigma_i = \{f_i\}$ and constants U

$$\phi \quad : \quad t_1 \approx s_1 \ \wedge \ t_2 \approx s_2 \ \wedge \cdots \wedge \ t_k \approx s_k \ \wedge$$
$$(t'_1 \not\approx s'_1 \ \wedge \cdots \wedge \ t'_l \neq s'_l)$$

s_i, t_i, s'_i, t'_i terms over $\cup_i \Sigma_i$

Is ϕ satisfiable?

Use variable abstraction and deal with T_i separately!

Signatures and Extensions

How do equations look like?

D-rules : $f(c_1, \ldots, c_k) \rightarrow c_0$, where $f \in \Sigma$ and $c_0, \ldots, c_k \in U$.

C-rules : $c \rightarrow d$, where $c, d \in U$.

Example. Let $\Sigma_0 = \{a,\ b,\ f\}$, and let

$$\mathcal{E}_0 = \{f(f(a,b),b) \approx b,\quad f(a,b) \approx a\}.$$

Then,

$$\begin{aligned} D_0 &= \{a \rightarrow c_1,\ b \rightarrow c_2,\ f(c_1,c_2) \rightarrow c_3,\ f(c_3,c_2) \rightarrow c_4\}, \\ C_0 &= \{c_4 \rightarrow c_2,\ c_3 \rightarrow c_1\}. \end{aligned}$$

Here, $U = \{c_1, c_2, \ldots\}$.

Individual Theories

How to reason in T_i?

Use standard critical-pair completion for ground equations over $\Sigma_i \cup U$.

Superposition: $$\frac{f(\ldots) \to c, f(\ldots) \to d}{f(\ldots) \to c, c \approx d}$$

Collapse: $$\frac{f(\ldots, c, \ldots) \to c', c \to d}{f(\ldots, d, \ldots) \to c', c \to d}$$

Composition: $$\frac{f(\ldots) \to c, c \to d}{f(\ldots) \to d, c \to d}$$

Other Transition Rules

Extension:
$$\frac{C[g(\ldots)] \approx t}{C[c'] \approx t, g(\ldots) \to c'}$$

Simplification:
$$\frac{C[g(\ldots)] \approx t, g(\ldots) \to c'}{C[c'] \approx t, g(\ldots) \to c'}$$

Deletion:
$$\frac{s \approx s}{}$$

Orientation:
$$\frac{f(\ldots) \approx c}{f(\ldots) \to c}$$

Abstract Congruence Closure

A ground rewrite system $R = D \cup C$ is an *(abstract) congruence closure* (over Σ and $K \subset U$) *for* E if

1. Every normal form $c \in K$ represents a term $t \in \mathcal{T}(\Sigma)$ via R, and
2. R is a fully reduced, ground convergent rewrite system over terms in $\mathcal{T}(\Sigma \cup K)$.
3. For all terms $s, t \in \mathcal{T}(\Sigma)$, we have:

$$s \leftrightarrow_E^* t \text{ if and only if } s \downarrow_R t.$$

Example: Abstract Congruence Closure

Let

$$E_0 = \{f(a,b) \approx a, \;\; f(f(a,b),b) \approx b\}.$$

Then,

$$\begin{aligned}
D_0 &= \{a \to c_1,\ b \to c_2,\ f(c_1,c_2) \to c_1,\ f(c_1,c_2) \to c_2\}, \\
D_1 &= \{a \to c_1,\ b \to c_2,\ f(c_1,c_2) \to c_2\}, \;\; C_1 = \{c_1 \to c_2\}, \\
D_2 &= \{a \to c_2,\ b \to c_2,\ f(c_2,c_2) \to c_2\}.
\end{aligned}$$

The set D_2 is an abstract congruence closure for E_0.

What did we gain?

- Time complexity: Exponential to Polynomial jump
- Very simple ordering used
- Generalize to handle AC symbols: Suppose $\Sigma_{AC} \subset \Sigma$ defined to be AC. We additionally need completion procedure for commutative monoids. Almost straight-forward extension. Cf. regular ground AC-completion.
- Ground convergent systems:

$$\begin{array}{ccccc} E & \mapsto & R = D \cup C & \mapsto^? & E' \\ \Sigma & & \Sigma \cup K & & \Sigma(\text{ convergent}) \end{array}$$

- Non-symmetric relations: Non-termination to Polynomial jump

Complexity Analysis

1. Extension, Simplification, Orientation, Deletion Steps: $O(n)$ steps.
2. Superposition, Collapse, and Composition: $O(n\delta)$ steps, where δ is the depth of the ordering on K.

Consider a rule obtained after the first stage above:

$$\underline{f}\ (\underline{c_1},\ \underline{c_2}, \ldots,\ \underline{c_k})\ \rightarrow\ \underline{c}$$

Each marked position changes at most δ times and there are $O(n)$ such positions.

Time complexity: $O(n\delta)$.

Abstract Rewrite Closure

Reln	Ordered Inference Rule	Ground Case?	Our Method
$\approx$	superposition	EXP time	Poly time
$\rightarrow$	ordered partial para-modulation	non-terminating	Poly time
$\approx \cup \rightarrow$	combination	non-terminating	Poly time

Why?

Substructure sharing using new constants and extra flexibility in choosing ordering.

Combining Concepts from Different Areas

Interpretations for Extended Signatures:

The DAG interpretation for Congruence Closure Algorithms The constants $c_1, c_2, \ldots \in U$ are pointers (to vertices in a DAG) and a D-rule $f(c_1, c_2) \to c$ says that the c points to the DAG vertex with symbol f and pointers c_1 and c_2.

The states interpretation for Tree Automata The constants $c_1, c_2, \ldots \in U$ are states of an automaton and a D-rule $f(c_1, c_2) \to c$ represents a transition of a bottom-up tree automata.

In other words, we have combined techniques from tree automata, standard rewriting, and DAG-based implementations.

Deciding Confluence of Ground TRS

Consider

$$E_0 = \{a \to fab,\ fab \to fba\}$$

and

$$E_1 = \{gfa \to fgfa,\ gfa \to ffa,\ ffa \to fa\}$$

E_1 is confluent, i.e., any two congruent terms can be rewritten to a common term, e.g., $fga \leftrightarrow^*_{E_1} ffgfa$ and

$$fga \ \to^* \ fa \ \leftarrow^* \ ffgfa$$

whereas E_0 is not, e.g., $f(fba, b) \leftrightarrow^*_{E_0} fba$, but

$$f(fba, b) \ \to^* ?? \ \leftarrow^* \ f(b, a)$$

What was known?

- Reachability for GTRS is decidable in polynomial time: Using tree automata techniques
- Congruence for GTRS is decidable in polynomial time: Using dag based congruence closure algorithms
- Confluence for GTRS is decidable in exponential time: Using tree automata techniques (Ground Tree Transducers)
- Open: Polynomial time algorithm for deciding confluence of ground term rewrite systems?

Polynomial Time Algorithms

Let E be the input set of ground rewrite rules.

1. Construct an abstract rewrite closure $D \cup F \cup B$ for E
2. Construct an abstract congruence closure R for E (over the same extended signature used above)
3. Check that every pair of constants c and d in the same congruence class (in R) rewrite to some common term using $D \cup F \cup B$
4. Check that every constant c and signature $f(\ldots)$ in the same congruence class (in R) rewrite to some common term using $D \cup F \cup B$

References

1. With L. Bachmair, "Abstract Congruence Closure", *To appear in JAR, Prelim version in CADE 2000*.

2. "Rewrite Closures for Ground and Cancellative AC Theories", In *FST&TCS 2001*.

3. "Polynomial time algorithms for deciding confluence of certain term rewrite systems", Submitted to *LICS 2002*.

http://www.csl.sri.com/users/tiwari/