On Ordering Constraints for Deduction with Built-In Abelian Semigroups, Monoids and Groups

Guillem Godoy and Robert Nieuwenhuis
Technical University of Catalonia
Barcelona

LICS 2001

Domain knowledge in deduction. Why building it in?

Example: FOL with equality. One can:

1. use resolution

$$\frac{A \vee C \qquad A' \vee D}{C\sigma \vee D\sigma} \qquad \text{if } \sigma = mgu(A, A')$$

+ equality axioms:

$$\begin{array}{rll}
 & \rightarrow x \simeq y & \textit{(reflexivity)} \\
x \simeq y & \rightarrow y \simeq x & \textit{(symmetry)} \\
x \simeq y \wedge y \simeq z & \rightarrow x \simeq z & \textit{(transitivity)} \\
x \simeq y & \rightarrow f(\ldots x \ldots) \simeq f(\ldots y \ldots) & \\
x \simeq y \ \wedge \ P(\ldots x \ldots) & \rightarrow P(\ldots y \ldots) & \\
 & \textit{(monotonicity)} &
\end{array}$$

search space too large

2. built-in equality: paramodulation [RW'69]:

$$\frac{C \vee s \simeq t \qquad D}{C\sigma \vee D\sigma[t\sigma]_p} \qquad \text{if } \sigma = mgu(s, D|_p)$$

can impose restrictions like $s\sigma > C\sigma, t\sigma$ where $>$ is a reduction ordering on ground terms.

method of choice in practice (see [NR'01])

Building in other domain knowledge

Build in assoc. and commut. AC of $+$.
Can flatten $+(+(a,b),+(c,a))$ into $2a+b+c$

Can prove unsatisfiability of the clause set:

$$\{\ P(x+2a+b), \quad \neg P(a+c+y)\ \}$$

in one sigle step of resolution with
AC-unification: $x\sigma = c+z, \quad y\sigma = a+b+z$.

Other solution: $x\sigma = c, \quad y\sigma = a+b$ (not needed in case AC0: abelian monoids).

Ordered AC-Resolution:

$$\frac{A \vee C \qquad A' \vee D}{C\sigma \vee D\sigma} \qquad \begin{array}{l}\text{if } \sigma \in \textit{AC-mgu}(A, A') \\ \text{and } A\sigma > C\sigma, D\sigma\end{array}$$

where $>$ is an AC-compatible ordering on ground atoms.

Drawback: each inference produces as many conclusions as unifiers; for AC, doubly exponentially many!

Solution: inherited equality constraints

Then (E-)Resolution becomes:

$$\frac{A \vee C \mid T \qquad A' \vee D \mid T'}{C \vee D \mid T \wedge T' \wedge A = A'}$$

Semantics of $C \mid T$:

All ground instances $C\sigma$ such that $\sigma \models T$

where $=$ is interpreted as $=_E$ on ground terms (if $E = \emptyset$, syntactic equality).

- If T is unsatisfiable, $C \mid T$ is redundant
- Unification problems stored as constraints
- Generate one conclusion, also if $E \neq \emptyset$
- Do not need to compute unifiers; only check (E-)unifiability
- Do not need inferences on subterms coming from unifiers: Basic Strategy

Can also inherit ordering constraints

Then (E-)resolution becomes:

$$\frac{A \vee C \mid T \qquad A' \vee D \mid T'}{C \vee D \mid T \wedge T' \wedge\ A = A'\ \wedge\ A > C\ \wedge\ A > D}$$

where $=$ and $>$ are interpreted as given E-congruence and ordering on ground terms.

- ordering constraints are checked to be compatible with the ones of previous inferences: smaller search space
- deciding constraint satisfiability is hard
- Refutation completeness with constraints:
  - easy to prove for resolution: CLP(X)
  - hard to prove for paramodulation:
    - incomplete w/ initial constraints
    - special inference rules for each E
    - see [NR'01]

Paramodulation with Built-in AG
[GN, LICS 2000]

What did we prove?

- new technique for ordered paramodulation with built-in abelian groups (AG).
- uses AG-unification and not AC-unification
- no inferences at all with the AG axioms
- compatible with constraint inheritance
- yields decision procedure for ground clauses modulo AG
- can use orderings $>$ based on recursive path ordering (RPO) on flattened terms

Problem: could only approximate constraint satisfiability, i.e., detect some unsatisfiable constraints.

Here we solve this problem (among others).

Definition of the orderings

Let R_{AG} be:

$$\begin{array}{rcll} x+0 & \to & x & \quad (R_0) \\ -x+x & \to & 0 & \\ -(-x) & \to & x & \\ -0 & \to & 0 & \\ -(x+y) & \to & (-x)+(-y) & \end{array}$$

applied modulo AC of (flattened) $+$

Define:

$$\begin{array}{llllll} s & \succ_{frpo} & t & \text{if} & \mathit{flat}(s) & \succ_{rpo} & \mathit{flat}(t) \\ s & \succ_{\mathit{ac0\text{-}rpo}} & t & \text{if} & \mathit{nf}_{R_0}(s) & \succ_{frpo} & \mathit{nf}_{R_0}(t) \\ s & \succ_{\mathit{ag\text{-}rpo}} & t & \text{if} & \mathit{nf}_{R_{AG}}(s) & \succ_{frpo} & \mathit{nf}_{R_{AG}}(t) \end{array}$$

where:

- *nf* means normal form
- $\succ_{rpo}$ has precedence $f_1 \ \ldots \ f_n \ - \ + \ 0$ (with 0 smallest)
- in $\succ_{rpo}$ we have $mul = \{+\}$

Thm: The orderings have the right properties for AC- AC0- and AG-paramodulation.

Results

- NP-algorithms for each one of these orderings
- General hardness result:
  For any well-founded E-compatible ordering for these E, constraint sat is NP-hard already for conjunctions of inequations.

Related work: in [CNR95] (LICS'95) we solved AC-RPO. Good/Bad properties:

- complex, no AG or AC0, very exponential
- less restrictions on precedence
- decides satisfiability w.r.t. solutions that may have new symbols: less restrictive but needed for other applications
- technique cannot give NP algorithms because large solutions are built and inserted

Outline of the algorithms

AC case (relatively easy). Example: take

$$f(x+z) > y \ \wedge \ z > f(x)$$

1. Turn into disjunction of certain linear constraints. One of them is

$$f(w_1) > w_1 = x+y > y > w_2 = f(x) > x = 0$$

2. Define segments and apply splitting transf. Aim: sols. w/ local segment summands only.

1st segment: guess to split w_1, not y. Get:

$$f(W_1+w_1') > W_1+w_1' = x+y > y > w_2 = f(x)$$

For $W_1+w_1' > y$ guess $W_1 = y$. Insert w_1' (0):

get $f(y+x) > y > w_2 = f(x) > x = 0$

2nd segment: no splitting needed.

Outline of the algorithms (cont.)

3. Define system of diophantine (in)eqs.

Each segment produces several (in)equations.

Remember: one summand per segment.

Example: The system for

$$S = \quad w_1 = f(y+x) > y > w_3 = f(x) > x = 0$$

is

$$D_S = \begin{cases} w_1 & = & 1 \\ y & > & w_3 \\ w_3 & = & 1 \\ x & = & 1 \end{cases}$$

Lemma: S sat iff D_S sat over $\mathbb{N}^+$.

Thm: $\succ_{frpo}$ constraint satisfaction is in NP.

Very similar techniques for $\succ_{\mathit{ac0\text{-}rpo}}$.

Outline of the algorithms: the AG case

Examples over signature $f > a > - > + > 0$:

1. Terms (ordinals) in increasing order:

$0\ {\scriptstyle 0}$ $\quad a\ {\scriptstyle 1}$ $\quad a+a\ \ (=2a)\ {\scriptstyle 2}$ $\quad \ldots$ $\quad -a\ {\scriptstyle \omega}$
$-2a\ {\scriptstyle \omega+1}$ $\quad \ldots$ $\quad f(0)\ {\scriptstyle 2\omega}$ $\quad f(0)+a\ {\scriptstyle 2\omega+1}$ $\quad \ldots$
$f(0)-a\ {\scriptstyle 3\omega}$ $\quad \ldots$ $\quad 2f(0)\ {\scriptstyle 4\omega}$ $\quad 2f(0)+a\ {\scriptstyle 4\omega+1}$
$\ldots$ $\quad -f(0)\ {\scriptstyle \omega^2}$ $\quad \ldots$ $\quad f(a)\ {\scriptstyle 2\omega^2}$ $\quad \ldots$

2. $f(f(a)) \succ_{ag\text{-}rpo} f(a-f(0)+f(a-a))$ since

$$\begin{array}{ll} nf_{R_{AG}}(\ \ f(f(a))\ \ ) & = f(f(a)) \succ_{frpo} \\ nf_{R_{AG}}(\ \ f(a-f(0)+f(a-a))\ \ ) & = f(a) \end{array}$$

3. Terms can be smaller than their subterms:

$\sigma \models x > f(x-f(a))$ if $x\sigma = f(a)$, since

$$\begin{array}{ll} nf_{R_{AG}}(\ \ f(a)\ \ ) & = f(a) \succ_{frpo} \\ nf_{R_{AG}}(\ \ f(f(a)-f(a))\ \ ) & = f(0) \end{array}$$

Algorithms: the AG case (cont.)

4. another difference with AC: after splitting solutions may still need several summands:

If $f > - > + > 0$ then

$$\sigma \models f(f(f(0))) > -z > z > y > -y > f(0)$$

if $y\sigma = -2f(0)$ and $z\sigma = 2f(f(0))$.

But no solution where $y\sigma$ and $z\sigma$ are built from one single summand.

Key idea: can bound number of decisive summands and extend the splitting transformation and D_S accordingly (technically complex).

Again S sat iff D_S sat over $\mathbb{N}^+$ and

Thm: $\succ_{\textit{ag-rpo}}$ constraint satisfaction is in NP

Further work

- (easy) Extend to additional constraints like:
  - "x has only one summand"
  - "x and y are sums of the same summand"
  - etc.

  Very useful for applications of [GN'00]
- Implementation
- (hard) Can we solve D_S over the integers in case of $\succ_{\textit{ag-rpo}}$?
- (hard) Other E-compatible orderings