To work effectively with constrained clauses in a calculus, we need methods to check the satisfiability of constraints:

Possible for LPO, KBO, but expensive.

If constraints become too large, we may delete some conjuncts of the constraint. (Note that the calculus remains sound, if constraints are replaced by implied constraints.)

Refutational Completeness

The refutational completeness proof for constraint superposition looks mostly like in Sect. 3.4.

Lifting works as before, so every ground infererence that is required in the proof is an instance of some inference from the corresponding constrained clauses. (Easy.)

There is one significant problem, though.

Case 2 in the proof of Thm. 3.9 does not work for constrained clauses:

If we have a ground instance $C\theta$ where $x\theta$ is reducible by $R_{C\theta}$, we can no longer conclude that $C\theta$ is true because it follows from some rule in $R_{C\theta}$ and some smaller ground instance $C\theta'$.

Example: Let $C\,[\![K]\!]$ be the clause $f(x) \approx a\ [\![x \succ a]\!]$, let $\theta = \{x \mapsto b\}$, and assume that $R_{C\theta}$ contains the rule $b \to a$.
Then θ satisfies K, but $\theta' = \{x \mapsto a\}$ does not, so $C\theta'$ is *not* a ground instance of $C\,[\![K]\!]$.

Solution:

Assumption: We start the saturation with a set N_0 of *unconstrained* clauses; the limit N_* contains constrained clauses, though.

During the model construction, we ignore ground instances $C\theta$ of clauses in N_* for which $x\theta$ is reducible by $R_{C\theta}$.

We obtain a model R_∞ of all *variable irreducible* ground instances of clauses in N_*.

R_∞ is also a model of all *variable irreducible* ground instances of clauses in N_0.

Since all clauses in N_0 are unconstrained, every ground instance of a clause in N_0 follows from some rule in R_∞ and some smaller ground instance; so it is true in R_∞.

Consequently, R_∞ is a model of *all* ground instances of clauses in N_0.

Other Constraints

The approach also works for other kinds of constraints.

In particular, we can replace unification by equality constraints ($\rightsquigarrow$ "basic superposition"):

Pos. Superposition:
$$\frac{D' \lor t \approx t' \; [\![K_2]\!] \qquad C' \lor s[u] \approx s' \; [\![K_1]\!]}{D' \lor C' \lor s[t'] \approx s' \; [\![K_2 \land K_1 \land K]\!]}$$
where u is not a variable and
$K = (t = u)$

Note: In contrast to ordering constraints, these constraints are essential for soundness.

The Drawback

Constraints reduce the number of required inferences; however, they are detrimental to redundancy:

Since we consider only *variable irreducible* ground instances during the model construction, we may use only such instances for redundancy:

A clause is redundant, if all its variable irreducible ground instances follow from smaller variable irreducible ground instances.

Even worse, since we don't know R_∞ in advance, we must consider variable irreducibility w. r. t. arbitrary rewrite systems.

Consequence: Not every subsumed clause is redundant!

Literature

Robert Nieuwenhuis, Albert Rubio: Paramodulation-Based Theorem Proving. Handbook of Automated Reasoning, Vol. 1, Ch. 7, pp. 371–443, Elsevier Science B.V., 2001.

3.8 Hierarchic Superposition

The superposition calculus is a powerful tool to deal with formulas in *uninterpreted* first-order logic.

What can we do if some symbols have a *fixed interpretation*?

Can we combine superposition with decision procedures, e. g., for linear rational arithmetic? Can we integrate the decision procedure as a “black box”?

Sorted Logic

It is useful to treat this problem in sorted logic (cf. Sect. 1.11, page 31).

A many-sorted signature $\Sigma = (\Xi, \Omega, \Pi)$ fixes an alphabet of non-logical symbols, where

- Ξ is a set of sort symbols,
- Ω is a sets of function symbols,
- Π is a set of predicate symbols.

Each function symbol $f \in \Omega$ has a unique declaration $f : \xi_1 \times \cdots \times \xi_n \to \xi_0$; each predicate symbol $P \in \Pi$ has a unique declaration $P : \xi_1 \times \cdots \times \xi_n$ with $\xi_i \in \Xi$.

In addition, each variable x has a unique declaration $x : \xi$.

We assume that all terms, atoms, substitutions are well-sorted.

A many-sorted algebra $\mathcal{A}$ consists of

- a non-empty set $\xi_\mathcal{A}$ for each $\xi \in \Xi$,
- a function $f_\mathcal{A} : \xi_{1,\mathcal{A}} \times \cdots \times \xi_{n,\mathcal{A}} \to \xi_{0,\mathcal{A}}$ for each $f : \xi_1 \times \cdots \times \xi_n \to \xi_0 \in \Omega$,
- a subset $P_\mathcal{A} \subseteq \xi_{1,\mathcal{A}} \times \cdots \times \xi_{n,\mathcal{A}}$ for each $P : \xi_1 \times \cdots \times \xi_n \in \Pi$.

Hierarchic Specifications

A *specification* $SP = (\Sigma, \mathcal{C})$ consists of

- a signature $\Sigma = (\Xi, \Omega, \Pi)$,
- a class of term-generated Σ-algebras $\mathcal{C}$ closed under isomorphisms.

If $\mathcal{C}$ consists of *all* term-generated Σ-algebras satisfying the set of Σ-formulas N, we write $SP = (\Sigma, N)$.

A *hierarchic specification* $HSP = (SP, SP')$ consists of

- a base specification $SP = (\Sigma, \mathcal{C})$,
- an extension $SP' = (\Sigma', N')$,

where $\Sigma = (\Xi, \Omega, \Pi)$, $\Sigma' = (\Xi', \Omega', \Pi')$, $\Xi \subseteq \Xi'$, $\Omega \subseteq \Omega'$, and $\Pi \subseteq \Pi'$.

A Σ'-algebra $\mathcal{A}$ is called a model of $HSP = (SP, SP')$, if $\mathcal{A}$ is a model of N' and $\mathcal{A}|_\Sigma \in \mathcal{C}$, where the reduct $\mathcal{A}|_\Sigma$ is defined as $((\xi_\mathcal{A})_{\xi\in\Xi}, (f_\mathcal{A})_{f\in\Omega}, (P_\mathcal{A})_{P\in\Pi})$.

Note:

- no confusion: models of HSP may not identify elements that are different in the base models.
- no junk: models of HSP may not add new elements to the interpretations of base sorts.

Example:

Base specification: $((\Xi, \Omega, \Pi), \mathcal{C})$, where

$$\Xi = \{\mathit{int}\}$$

$$\begin{aligned}\Omega = \{\, & 0, 1, -1, 2, -2, \ldots : \to \mathit{int},\\ & - : \mathit{int} \to \mathit{int},\\ & + : \mathit{int} \times \mathit{int} \to \mathit{int}\,\}\end{aligned}$$

$$\begin{aligned}\Pi = \{\, & \geq : \mathit{int} \times \mathit{int},\\ & > : \mathit{int} \times \mathit{int}\,\}\end{aligned}$$

$$\mathcal{C} = \text{isomorphy class of } \mathbb{Z}$$

Extension: $((\Xi', \Omega', \Pi'), N')$, where

$$\Xi' = \Xi \cup \{\mathit{list}\}$$

$$\begin{aligned}\Omega' = \Omega \cup \{\, & \mathit{cons} : \mathit{int} \times \mathit{list} \to \mathit{list},\\ & \mathit{length} : \mathit{list} \to \mathit{int},\\ & \mathit{empty} : \to \mathit{list},\\ & a : \to \mathit{list}\,\}\end{aligned}$$

$$\Pi' = \Pi$$

$$\begin{aligned}N' = \{\, & \mathit{length}(a) \geq 1,\\ & \mathit{length}(\mathit{cons}(x, y)) \approx \mathit{length}(y) + 1\,\}\end{aligned}$$

Goal:

Check whether N' has a model in which the sort int is interpreted by $\mathbb{Z}$ and the symbols from Ω and Π accordingly.

Hierarchic Superposition

In order to use a prover for the base theory, we must preprocess the clauses:

A term that consists only of base symbols and variables of base sort is called a base term (analogously for atoms, literals, clauses).

A clause C is called *weakly abstracted*, if every base term that occurs in C as a subterm of a non-base term (or non-base non-equational literal) is a variable.

Every clause can be transformed into an equivalent weakly abstracted clause. We assume that all input clauses are weakly abstracted.

A substitution is called simple, if it maps every variable of a base sort to a base term.

The inference rules of the hierarchic superposition calculus correspond to the rules of of the standard superposition calculus with the following modifications:

- The term ordering $\succ$ must have the property that every base ground term (or non-equational literal) is smaller than every non-base ground term (or non-equational literal).
- We consider only simple substitutions as unifiers.
- We perform only inferences on non-base terms (or non-base non-equational literals).
- If the conclusion of an inference is not weakly abstracted, we transform it into an equivalent weakly abstracted clause.

While clauses that contain non-base literals are manipulated using superposition rules, base clauses have to be passed to the base prover.

This yields one more inference rule:

Constraint Refutation: $$\frac{M}{\bot}$$

where M is a set of base clauses that is inconsistent w.r.t. $\mathcal{C}$.

Problems

There are two potential problems that are harmful to refutational completeness:

- We can only apply the constraint refutation rule to finite sets M. If $\mathcal{C}$ is not compact, this is not sufficient.
- Since we only consider simple substitutions, we will only obtain a model of all *simple ground instances*.

  To show that we have a model of *all* instances, we need an additional condition called *sufficient completeness w. r. t. simple instances*.

A set N of clauses is called *sufficiently complete with respect to simple instances*, if for every model $\mathcal{A}'$ of the set of simple ground instances of N and every ground non-base term t of a base sort there exists a ground base term t such that $t' \approx t$ is true in $\mathcal{A}'$.

Note: Sufficient completeness w. r. t. simple instances ensures the absence of junk.

If the base signature contains Skolem constants, we can sometimes enforce sufficient completeness by equating ground extension terms with a base sort to Skolem constants.

Skolem constants may harmful to compactness, though.

Completeness of Hierarchic Superposition

If the base theory is compact, the hierarchic superposition calculus is refutationally complete for sets of clauses that are sufficiently complete with respect to simple instances (Bachmair, Ganzinger, Waldmann, 1994; Baumgartner, Waldmann 2013).

Main proof idea:

If the set of base clauses in N has some base model, represent this model by a set E of convergent ground equations and a set D of ground disequations.

Then show: If N is saturated w. r. t. hierarchic superposition, then $E \cup D \cup \tilde{N}$ is saturated w. r. t. standard superposition, where $\tilde{N}$ is the set of simple ground instances of clauses in N that are reduced w. r. t. E.

A Refinement

In practice, a base signature often contains *domain elements*, that is, constant symbols that are

- guaranteed to be different from each other in every base model, and
- minimal w. r. t. $\succ$ in their equivalent class.

Typical example for domain elements: number constants $0, 1, -1, 2, -2, \dots$

If the base signature contains *domain elements*, then weak abstraction can be redefined as follows:

A clause C is called *weakly abstracted*, if every base term that occurs in C as a subterm of a non-base term (or non-base non-equational literal) is a variable *or a domain element*.

Why does that work?

Literature

Leo Bachmair, Harald Ganzinger. Uwe Waldmann: Refutational Theorem Proving for Hierarchic First-Order Theories. Applicable Algebra in Engineering, Communication and Computing, 5(3/4):193–212, 1994.

Peter Baumgartner, Uwe Waldmann: Hierarchic Superposition With Weak Abstraction. Automated Deduction, CADE-24, LNAI 7898, pp. 39–57, Springer, 2013.

3.9 Integrating Theories I: E-Unification

Dealing with mathematical theories naively in a superposition prover is difficult:

Some axioms (e. g., commutativity) cannot be oriented w. r. t. a reduction ordering.
⇒ Provers compute many equivalent copies of a formula.

Some axiom sets (e. g., torsion-freeness, divisibility) are infinite.
⇒ Can we tell which axioms are really needed?

Hierarchic ("black-box") superposition is easy to implement, but conditions like compactness and sufficient completeness are rather restrictive.

Can we integrate theories directly into theorem proving calculi ("white-box" integration)?

Idea:

In order to avoid enumerating entire congruence classes w. r. t. an equational theory E, treat formulas as *representatives* of their congruence classes.

Compute an inference between formula C and D if an inference between some clause represented by C and some clause represented by D would be possible.

Consequence: We have to check whether there are substitutions that make terms s and t equal w. r. t. E.
⇒ Unification is replaced by E-unification.

E-Unification

E-unification (unification modulo an equational theory E):

For a set of equality problems $\{s_1 \approx t_1, \dots, s_n \approx t_n\}$, an E-unifier is a substitution σ such that for all $i \in \{1, \dots, n\}$: $s_i\sigma \approx_E t_i\sigma$.

Recall: $s_i\sigma \approx_E t_i\sigma$ means $E \models s_i\sigma \approx t_i\sigma$.

In general, there are infinitely many (E-)unifiers.
What about most general unifiers?

Frequent cases: $E = \emptyset$, $E = \text{AC}$, $E = \text{ACU}$:

$$\begin{aligned} x + (y + z) &\approx (x + y) + z && \text{(associativity = A)} \\ x + y &\approx y + x && \text{(commutativity = C)} \\ x + 0 &\approx x && \text{(identity (unit) = U)} \end{aligned}$$

The identity axiom is also abbreviated by "1", in particular, if the binary operation is denoted by $*$. (ACU = AC1).

Example:

$x + y$ and c are ACU-unifiable with $\{x \mapsto c,\ y \mapsto 0\}$ and $\{x \mapsto 0,\ y \mapsto c\}$.

$x + y$ and $x' + y'$ are ACU-unifiable with $\{x \mapsto z_1 + z_2,\ y \mapsto z_3 + z_4,\ x' \mapsto z_1 + z_3,\ y' \mapsto z_2 + z_4\}$ (among others).

More general substitutions:

Let X be a set of variables.
A substitution σ is more general modulo E than a substitution σ' on X, if there exists a substitution ρ such that $x\sigma\rho \approx_E x\sigma'$ for all $x \in X$.

Notation: $\sigma \lesssim_E^X \sigma'$.

(Why X? Because we cannot restrict to idempotent substitutions.)

Complete sets of unifiers:

Let S be an E-unification problem, let $X = Var(S)$.
A set C of E-unifiers of S is called complete (CSU),
if for every E-unifier σ' of S there exists a $\sigma \in C$
with $\sigma \lesssim_E^X \sigma'$.

A complete set of E-unifiers C is called minimal (μCSU),
if for all $\sigma, \sigma' \in C$, $\sigma \lesssim_E^X \sigma'$ implies $\sigma = \sigma'$.

Note: every E-unification problem has a CSU. (Why?)

The set of equations E is of unification type

unitary, if every E-unification problem has a μCSU with cardinality ≤ 1 (e. g.: $E = \emptyset$);

finitary, if every E-unification problem has a finite μCSU (e. g.: $E =$ ACU, $E =$ AC, $E =$ C);

infinitary, if every E-unification problem has a μCSU and some E-unification problem has an infinite μCSU (e. g.: $E =$ A);

zero (or nullary), if some E-unification problem does not have a μCSU (e. g.: $E = \mathrm{A} \cup \{x + x \approx x\}$).

Unification modulo ACU

Let us first consider elementary ACU-unification:
the terms to be unified contain only variables and the function symbols from $\Sigma = (\{+/2,\ 0/0\}, \emptyset)$.

Since parentheses and the order of summands don't matter, every term over $X_n = \{x_1, \dots, x_n\}$ can be written as a sum $\sum_{i=1}^{n} a_i\, x_i$.

The ACU-equivalence class of a term $t = \sum_{i=1}^{n} a_i\, x_i \in T_\Sigma(X_n)$ is uniquely determined by the vector $\vec{v}_n(t) = (a_1, \dots, a_n)$.

Analogously, a substitution $\sigma = \{\, x_i \to \sum_{j=1}^{m} b_{ij}\, x_j \mid 1 \le i \le n \,\}$ is uniquely determined by the matrix

$$M_{n,m}(\sigma) = \begin{pmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & & \vdots \\ b_{n1} & \cdots & b_{nm} \end{pmatrix}$$

Let $t = \sum_{i=1}^{n} a_i\, x_i$ and $\sigma = \{\, x_i \to \sum_{j=1}^{m} b_{ij}\, x_j \mid 1 \le i \le n \,\}$.

$$\begin{aligned} \text{Then } t\sigma &= \textstyle\sum_{i=1}^{n} a_i\, (\sum_{j=1}^{m} b_{ij}\, x_j) \\ &= \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} a_i\, b_{ij}\, x_j \\ &= \textstyle\sum_{j=1}^{m} \sum_{i=1}^{n} a_i\, b_{ij}\, x_j \\ &= \textstyle\sum_{j=1}^{m} (\sum_{i=1}^{n} a_i\, b_{ij})\, x_j. \end{aligned}$$

Consequence:

$$\vec{v}_m(t\sigma) = \vec{v}_n(t) \cdot M_{n,m}(\sigma).$$

Let $S = \{s_1 \approx t_1, \dots, s_k \approx t_k\}$ be a set of equality problems over $T_\Sigma(X_n)$.

Then the following properties are equivalent:

(a) σ is an ACU-unifier of S from $X_n \to T_\Sigma(X_m)$.

(b) $\vec{v}_m(s_i\sigma) = \vec{v}_m(t_i\sigma)$ for all $i \in \{1, \dots, k\}$.

(c) $\vec{v}_n(s_i) \cdot M_{n,m}(\sigma) = \vec{v}_n(t_i) \cdot M_{n,m}(\sigma)$ for all $i \in \{1, \dots, k\}$.

(d) $(\vec{v}_n(s_i) - \vec{v}_n(t_i)) \cdot M_{n,m}(\sigma) = \vec{0}_m$ for all $i \in \{1, \dots, k\}$.

(e) $M_{k,n}(S) \cdot M_{n,m}(\sigma) = \vec{0}_{k,m}$.
where $M_{k,n}(S)$ is the $k \times n$ matrix whose rows are the vectors $\vec{v}_n(s_i) - \vec{v}_n(t_i)$.

(f) The columns of $M_{n,m}(\sigma)$ are non-negative integer solutions of the system of homogeneous linear diophantine equations $DE(S)$:

$$M_{k,n}(S) \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

Computing unifiers:

Obviously: if $\vec{y}_1, \dots, \vec{y}_r$ are solutions of $DE(S)$ and $a_1, \dots, a_r$ are natural numbers, then $a_1\vec{y}_1 + \cdots + a_r\vec{y}_r$ is also a solution. (In particular, the zero vector is a solution!)

In fact, one can compute a finite set of solutions $\vec{y}_1, \dots, \vec{y}_r$, such that every solution of $DE(S)$ can be represented as such a linear combination.

Moreover, if we combine these column vectors $\vec{y}_1, \dots, \vec{y}_r$ to an $n \times r$ matrix, this matrix represents a most general unifier of S. (Proof: see Baader/Nipkow.)

From ACU to AC

A complete set of AC-unifiers for elementary AC-unification problems can be computed from a most general ACU-unifier by some postprocessing.

Elementary AC-unification is finitary and the elementary unifiability problem is solvable in polynomial time.

But that does not mean that minimal complete sets of AC-unifiers can be computed efficiently.

E. Domenjoud has computed the exact size of AC-μCSUs for unification problems of the following kind:

$$m\,x_1 + \cdots + m\,x_p \approx n\,y_1 + \cdots + n\,y_q$$

where $gcd(m, n) = 1$.

The number of unifiers is

$$(-1)^{p+q} \sum_{i=0}^{p} \sum_{j=0}^{q} (-1)^{i+j} \binom{p}{i} \binom{q}{j} 2^{\binom{m+j-1}{m}\binom{n+i-1}{n}}$$

For $p = m = 1$ and $q = n = 4$, that is, for the equation

$$4\,x \approx y_1 + y_2 + y_3 + y_4$$

this is

$$34\,359\,607\,481.$$

Consequence:

If possible, avoid the enumeration of AC-μCSUs
(which may have doubly exponential size).

Rather: only check AC-unifiability.

Or: use ACU instead.

Unification with Constants

So far:

Elementary unification:
terms over variables and $\{+, 0\}$ or $\{+\}$.

Step 2:

Additional free constants.

Step 3:

Additional arbitrary free function symbols.
$\rightsquigarrow$ Unification in the union of disjoint equational theories.

Unification with constants:

We can treat constants a_i like variables x_i that *must* be mapped to themselves.

Consequence: The algorithm is similar to the one we have seen before, but we have to deal with homogeneous and inhomogeneous linear diophantine equations.

Some complexity bounds change, however:

Unification type:

elementary ACU-unification: unitary;
ACU-unification with constants: finitary.

Checking unifiability:

elementary ACU-unification: trivial;
ACU-unification with constants: NP-complete.

Combining Unification Procedures

The Baader–Schulz combination procedure allows to combine unification procedures for disjoint theories (e. g., ACU and the free theory).

Basic idea (as usual): Use abstraction to convert the combined unification problem into a union of two pure unification problems; solve them individually; combine the results.

Problem 1:

The individual unification procedures might map the same variable to different terms, e. g., $\{x \mapsto y + z\}$ and $\{x \mapsto f(w)\}$.

Solution: Guess for each variable non-deterministically which procedure treats it like a constant.

Problem 2:

Combining the results might produce cycles, e. g., $\{x \mapsto y + z\}$ and $\{y \mapsto f(x)\}$.

Solution: Guess an ordering of the variables non-deterministically; each individual unifier that is computed must respect the ordering.

Note: This is a non-trivial extension that may be impossible for some unification procedures (but it is possible for *regular* equational theories, i. e., theories where for each equation $u \approx v$ the terms u and v contain the same variables).

Literature

Franz Baader, Tobias Nipkow: Term Rewriting and All That. Cambridge University Press, 1998.

Franz Baader, Klaus Schulz: Unification in the union of disjoint equational theories: Combining decision procedures. Automated Deduction, CADE-11, LNCS 607, pp. 50–65, Springer, 1992.

Eric Domenjoud: A technical note on AC-unification. The number of minimal unifiers of the equation $\alpha x_1 + \dots + \alpha x_p \doteq_{\mathrm{AC}} \beta y_1 + \dots + \beta y_q$. Journal of Automated Reasoning, 8(1):39–44, 1992.

François Fages: Associative-commutative unification. Automated Deduction, CADE-7, LNCS 170, pp. 194–208, Springer, 1984.

Mike Livesey, Jörg H. Siekmann: Unification of AC-terms (bags) and ACI-terms (sets). Internal report, University of Essex, 1975.

Gordon Plotkin: Building-in equational theories. Machine Intelligence, 7:73–90, American Elsevier, 1972.

Manfred Schmidt-Schauß: Unification under associativity and idempotence is of type nullary. Journal of Automated Reasoning, 2:277–282, 1986.

3.10 Integrating Theories II: Calculi

We can replace syntactic unification by E-unification in the superposition calculus.

Moreover, it is usually necessary to choose a term ordering in such a way that all terms in an E-congruence class behave in the same way in comparisons (E-compatible ordering).

However, this is usually not sufficient.

AC and ACU

Example: Let $E = \text{AC}$. The clauses

$$\begin{aligned} a + b &\approx d \\ b + c &\approx e \\ c + d &\not\approx a + e \end{aligned}$$

are contradictory w. r. t. AC, but if $a \succ b \succ c \succ d \succ e$, then the maximal sides of these clauses are not AC-unifiable.

We have to compute inferences if some part of a maximal sum overlaps with a part of another maximal sum (the constant b in the example above).

Technically, we can do this in such a way that we first replace positive literals $s \approx t$ by $s + x \approx t + x$, and then unify maximal sides w. r. t. AC or ACU (Peterson and Stickel 1981, Wertz 1992, Bachmair and Ganzinger 1994).

However, it turns out that even if we integrate AC or ACU in such a way into superposition, the resulting calculus is not particularly efficient – not even for ground formulas.

This is not surprising: The uniform word problem for AC or ACU is EXPSPACE-complete (Cardoza, Lipton, and Meyer 1976, Mayr and Meyer 1982).

Abelian Groups

Working in Abelian groups is easier:

If we integrate also the inverse axiom, it is sufficient to compute inferences if the maximal part of a maximal sum overlaps with the maximal part of another maximal sum (like in Gaussian elimination).

Intuitively, in Abelian groups we can always isolate the maximal part of a sum on one side of an equation.

What does that mean for the non-ground case?

Example:

$$g(y) + x \not\approx 2z \quad \vee \quad f(x) + z \approx 2y$$

Shielded variables (x, y):

occur below a free function symbol,
$\leadsto$ cannot be mapped to a maximal term,
$\leadsto$ are not involved in inferences.

Unshielded variables (z):

can be instantiated with $m \cdot u + s$, where u is maximal,
$\leadsto$ must be considered in inferences,
$\leadsto$ variable overlaps (similar to ACU).

Variable overlaps are ugly:

If we want to derive a contradiction from

$$\begin{aligned} 2a &\approx c \\ 2b &\approx d \\ 2x &\not\approx c + d \end{aligned}$$

and $a \succ b \succ c \succ d$, we have to map x to a sum of two variables $x' + x''$, unify x' with a and x'' with b.

Divisible Torsion-free Abelian Groups

Working in divisible torsion-free Abelian groups is still easier:

DTAGs permit variable elimination.

Every clause can be converted into a DTAG-equivalent clause without *unshielded* variables.

Since only overlaps of maximal parts of maximal sums have to be computed, variable overlaps become unnecessary.

Moreover, if abstraction is performed eagerly, terms to be unified do not contain $+$, so ACU-unification can be replaced by standard unification.

Other Theories

A similar case: Chaining calculus for orderings.

$$\frac{D' \;\vee\; t' < t \qquad C' \;\vee\; s < s'}{(D' \;\vee\; C' \;\vee\; t' < s')\sigma}$$

where σ is a most general unifier of t and s.

Avoids explicit inferences with transitivity.
Only maximal sides of ordering literals have to be overlapped.
But unshielded variables can be maximal.

In dense linear orderings without endpoints, all unshielded variables can be eliminated.

DTAG-superposition and chaining can be combined to get a calculus for ordered divisible Abelian groups. Again, all unshielded variables can be eliminated.

Conclusion

Integrating theory axioms into superposition can become easier by integrating more axioms:

Easier unification problem (AC $\to$ ACU).

More restrictive inference rules (ACU $\to$ AG).

Fewer (or no) variable overlaps (AG $\to$ DTAG).

Main drawback of all theory integration methods:

For each theory, we have to start from scratch, both for the completeness proof and the implementation.

Literature

Leo Bachmair, Harald Ganzinger: Rewrite techniques for transitive relations. IEEE Symposium on Logic in Computer Science, LICS-9, pp. 384–393, 1994.

Leo Bachmair, Harald Ganzinger: Ordered chaining for total orderings. Automated Deduction, CADE-12, LNAI 814, pp. 435–450, Springer, 1994.

Leo Bachmair, Harald Ganzinger: Associative-commutative superposition. Conditional and Typed Rewriting Systems, CTRS-94, LNCS 968, pp. 1–14, Springer, 1994.

E. Cardoza, R. Lipton, A. R. Meyer: Exponential space complete problems for Petri nets and commutative semigroups: preliminary report. Eighth Annual ACM Symposium on Theory of Computing, STOC, pp. 50–54, 1976.

Guillem Godoy, Robert Nieuwenhuis: Paramodulation with built-in Abelian groups. IEEE Symposium on Logic in Computer Science, LICS-15, pp. 413–424, 2000.

Ernst W. Mayr, Albert R. Meyer: The complexity of the word problems for commutative semigroups and polynomial ideals. Advances in Mathematics, 46(3):305–329, 1982.

Gerald E. Peterson, Mark E. Stickel: Complete sets of reductions for some equational theories. Journal of the ACM, 28(2):233–264, 1981.

Uwe Waldmann: Cancellative abelian monoids and related structures in refutational theorem proving (Part I & II). Journal of Symbolic Computation, 33(6):777–829/831–861, 2002.

Uwe Waldmann: Superposition and chaining for totally ordered divisible abelian groups. Technical report MPI-I-2001-2-001, Max-Planck-Institut für Informatik, Saarbrücken, 2001.

Ulrich Wertz: First-order theorem proving modulo equations. Technical report MPI-I-92-216, Max-Planck-Institut für Informatik, Saarbrücken, 1992.