Combining Unification Procedures

Reference:

F. Baader and K. U. Schulz.
Unification in the union of disjoint equational theories: Combining decision procedures.
J. of Symb. Comp., 21(2):211–243, 1996.

Combining Unification Procedures

Framework:

Given: Unification procedures (or unifiability checkers) modulo sets of equations E_i over disjoint signatures Σ_i.

Wanted: Unification procedure (or unifiability checker) for unification problems over $\bigcup_i \Sigma_i$ modulo $\bigcup_i E_i$.

Combining Unification Procedures

Idea:

Purify the unification problem by introducing new variables.

Result: an equivalent set of pure equations: each equation contains only terms over variables and one Σ_i.

Equivalently: a set of sets of E_i-unification problems.

Solve the E_i-unification problems.

Combine the solutions.

Combining Unification Procedures

Purification:

As usual: introduce new variables if Σ_i-operators occur below Σ_j-operators or if an there is an equation between a term with a Σ_i-top symbol and a Σ_j-top symbol (with $i \neq j$).

Example:
$\Sigma_1 = \{f/1, a/0\}$, $\Sigma_2 = \{g/2\}$.
$g(x, a) \approx f(w)$ becomes $a \approx y$, $f(w) \approx z$, $g(x, y) \approx z$.

Every solution of the old system can be extended to a solution of the new system; every solution of the new system is an extension of some solution of the old system.

Combining Unification Procedures

Problem:

The solutions of the E_i-unification procedures may be contradictory:

Example:
Proc. 1 yields $[a/y,\ f(w)/z]$, proc. 2 yields $[g(x,y)/z]$.

Solution:

Decide non-deterministically in advance, which variable "belongs to" which unification procedure.

All other unification procedures have to treat this variable like a constant.

Combining Unification Procedures

Problem:

Combining the solutions of the E_i-unification procedures may result in cyclic structures:

Example:
Proc. 1 yields $[f(z)/w]$, proc. 2 yields $[g(w,y)/z]$.

Combining Unification Procedures

Solution:

Fix a linear ordering $<$ on variables non-deterministically in advance.

Consider only unifiers that satisfy the linear constant restriction:
If a variable x belonging to Proc. i is mapped to a term t, then t may not contain variables that are larger than x and belong to other procedures.

Note: This means that we need unification procedures that can deal with linear constant restrictions.
(This is a non-trivial modification!)

Combining Unification Procedures

Problem:

We may lose solutions of the original unification problem that map several variables to the same term, if unification procedures have to treat these variables as distinct constants:

Example:
$\Sigma_1 = \{f/1\}$, $E_1 = \emptyset$, $\Sigma_2 = \{*/2\}$, $E_2 = \mathrm{AC}$,
$f(x) \approx f(y)$, $x * z = y * z$.
If x and y belong to the same procedure, then the other equation does not have a solution;
if x and y belong to different procedures, then one of the solutions violates the linear constant restriction.

Combining Unification Procedures

Solution:

Decide non-deterministically in advance, which variables are mapped to the same term.

That is: Guess an equivalence relation on variables;
replace all variables of the same equivalence class by the same representative.

The Baader-Schulz Combination Procedure

Given: equational theories E_i over disjoint signatures Σ_i such that the solvability of E_i-unification problems with linear constant restrictions is decidable for $i \in \{1, \ldots, n\}$.
(For simplicity: $n = 2$.)

Alternatively given: procedures that compute (or enumerate) complete sets of E_i-unifiers with linear constant restrictions.

The Baader-Schulz Combination Procedure

Input: a unification problem Γ over $\bigcup_i \Sigma_i$.

Step 1: Variable abstraction

Step 2: Splitting of non-pure equations

Step 3: Variable identification:

Guess an equivalence relation on variables; replace all variables of the same equivalence class by the same representative.

The Baader-Schulz Combination Procedure

Step 4: Choose ordering and theory indices:

Guess a strict linear ordering $<$ on the variables of the system. Guess a mapping *ind* from the set of variables into the index set $\{1, 2\}$.

Step 5: Split systems:

Split the system into two systems Γ_1 and Γ_2, such that Γ_i contains only pure Σ_i-equations.

Step 6: Unify:

Check the unifiability (or: compute a CSU) of Γ_i with the linear constant restriction $<$ where all variables x with $ind(x) \neq i$ are treated as constants.

The Baader-Schulz Combination Procedure

Theorem 8.9:

Γ is solvable if and only if, for some choice in steps 3 and 4, both Γ_1 and Γ_2 are solvable.

The Baader-Schulz Combination Procedure

Solutions for Γ can be computed from solutions for Γ_1 and Γ_2:

Let σ_i be a solution of Γ_i. W.o.l.o.g. σ_i maps all variables with index i to terms containing only variables with index $j \neq i$ and "new" variables (not occurring in Γ, Γ_1, Γ_2).

Define σ inductively over $<$: Let x be a variable with $\mathit{ind}(x) = i$.
Assume that $y\sigma$ has already been defined for all $y < x$.
By assumption, all "old" variables y in $x\sigma_i$ have index $\mathit{ind}(y) \neq i$.
Since σ_i satisfies the linear constant restriction, we have $y < x$, so $y\sigma$ is already defined. Let $x\sigma$ be the term obtained from $x\sigma_i$ by replacing every y by $y\sigma$.

For variables z that have been replaced by x during step 3, define $z\sigma = x\sigma$.

The Baader-Schulz Combination Procedure

Theorem 8.10:

Let $(\Gamma_1^1, \Gamma_2^1), \dots, (\Gamma_1^k, \Gamma_2^k)$ be the pairs of unification problems that can be obtained from Γ for some choice in steps 3 and 4.

Let M_i^j be a complete set of solutions of the E_i-unification problem with linear constant restriction Γ_i^j.

Then the set of all substitutions σ that can be obtained as described from some $\sigma_1 \in M_1^j$ and $\sigma_2 \in M_2^j$ is a complete set of $(E_1 \cup E_2)$-unifiers for Γ.

The Baader-Schulz Combination Procedure

Proof of Theorem 8.9 and 8.10:

(a) If σ is computed as above then it is a solution of Γ.

(b) If Γ has a solution, then some (Γ_1^j, Γ_2^j) has a solution.

(c) If τ is a solution of Γ, then it is an instance of some σ computed as above from some (Γ_1^j, Γ_2^j).

The Baader-Schulz Combination Procedure

Proof (a):

It suffices to show: σ is a solution of the system Γ' that was split in step 5.

Let $s \approx t$ be some equation in Γ'.

W.o.l.o.g.: $s \approx t$ is contained in Γ_1.

σ_1 is a solution of Γ_1, i.e., $s\sigma_1 \approx_{E_1} t\sigma_1$.

Therefore $s\sigma_1\sigma \approx_{E_1} t\sigma_1\sigma$.

By the construction of σ, we have $s\sigma_1\sigma = s\sigma$ and $t\sigma_1\sigma = t\sigma$; therefore $s\sigma \approx_{E_1} t\sigma$.

Since $E_1 \subseteq E_1 \cup E_2$: $s\sigma \approx_{E_1 \cup E_2} t\sigma$. $\square$

The Baader-Schulz Combination Procedure

Proof (b), (c):

Rather technical (see Baader and Schulz, Sect. 4).

Unification with Linear Constant Restrictions

The Baader-Schulz combination procedure requires not only a procedure that solves a unification problem, but a procedure that solves a unification problem with a linear constant restriction.

Suppose we have an E_i-CSU of some unification problem.

Some of the substitutions may violate the linear constant restriction.

Can we simply drop them?

Unification with Linear Constant Restrictions

By dropping substitutions that violate the linear constant restriction, we may lose completeness:

Example:

Let E_1 be the axioms of Abelian groups.

Unification problem: $x \approx y + c$, LCR: $x < c < y$, $ind(x) = ind(y) = 1$, $ind(c) \neq 1$.

CSU: $[y + c/x]$.

This substitution violates the linear constant restriction, but it has an E_1-instance that satisfies the linear constant restriction: LCR-CSU: $[w/x, w - c/y]$.

Unification with Linear Constant Restrictions

But: no problem, for regular equational theories:

An equation $u \approx v$ is regular, if $Var(u) = Var(v)$.
A set of equations is regular, if all equations are regular.

Examples for regular equations: A, C, 1, I
(but not the inverse axiom).

Unification with Linear Constant Restrictions

Lemma 8.11:

Let E be regular. If σ violates a linear constant restriction, and $\sigma \lesssim_E \tau$, then τ violates the linear constant restriction.

Proof:

Assume that $x\sigma = t[c]$ with $x < c$.
Then $x\tau = t' \approx_E x\sigma\rho = (t[c])\rho$.
Obviously c occurs in $(t[c])\rho$.
Since t' can be obtained from $(t[c])\rho$ by applications of equations in E and no such application can delete c, c occurs also in t'.

Complexity Results

Corollary 8.12:

If E_i-unification with linear constant restrictions is finitary, then $\bigcup_i E_i$-unification is finitary.

Corollary 8.13:

If the unifiability procedures modulo E_i are in NP, then their combination is in NP.

Some Optimizations

Collapse-free theories:

An equation $x \approx t$ with $x \neq t$ is called a collapse equation. A set of equations is collapse-free, if it does not contain a collapse equation.

Let E_i be a collapse-free set of equations.
If the unification problem in step 4 contains an equation $x \approx t$ and t is a non-variable Σ_i-term, then all mappings *ind* with $ind(x) \neq i$ can be excluded.

Some Optimizations

Collapse-free theories:

Let E_1 and E_2 be collapse-free sets of equations.
If the unification problem in step 4 contains equations $x \approx t_1$ and $y \approx t_2$, and each t_i is a non-variable Σ_i-term, then x and y need not be identified.

Some Optimizations

Choosing a linear constant restriction:

A linear constant restriction restricts only variables with different indices.

Reordering variables with the same index does not influence the solutions, as long as the relationship to variables with a different index is not changed.

Hence: Fix some standard ordering $\prec$ on variables and require that consecutive variables with the same index must be ordered in $<$ in the same way as in $\prec$.

Some Optimizations

Free theories:

Suppose that one of the theories is free (that is, $E_i = \emptyset$).

Applying the Martelli-Montanari algorithm to Σ_i-equations after step 2 does not influence the set of solutions.

May detect $x \approx y$: x and y must be identified in step 3.

May detect $f(\dots) \approx g(\dots)$: fail immediately.

May detect $x \approx f(\dots)$: $ind(x)$ must be i (see above).