Program Verification: Lecture 8

José Meseguer

Computer Science Department
University of Illinois at Urbana-Champaign

Overlaps as Unification Problems

We reduced confluence (under the termination assumption) to joinability of context-free nested simplifications with overlap. But note that we can have a context-free overlap situation with equations $u = v$ and $u' = v'$ (again, with disjoint variables) if and only if there is a nonvariable position p in u and a substitution θ such that,

$$(\dagger) \quad u_p\theta = u'\theta.$$

Therefore, finding all possible context-free nested simplifications with overlap can be reduced to finding, for all pairs of equations $u = v$ and $u' = v'$ in E and all nonvariable positions p in u, all solutions to $(\dagger)$. Problems of the form $(\dagger)$ are called unification problems.

Unification

In general, the unification problem consists in, given terms t and t' whose sorts are in the same connected component, finding a substitution θ that makes them equal, so that we have identical terms, $t\theta = t'\theta$. The substitution θ is then called a unifier of t and t'.

Under very reasonable conditions on Σ, such as finiteness , this problem is decidable in a very strong sense. Namely, we can effectively find a finite set of unifiers $\{\theta_1, \ldots \theta_n\}$, that are the most general possible, in the sense that for any other substitution $\mu : \mathit{vars}(t = t') \longrightarrow T_\Sigma(V)$ such that $t\mu = t'\mu$, we can always find a θ_i, say, $\theta_i : \mathit{vars}(t = t') \longrightarrow T_\Sigma(X)$, and a substitution $\rho : X \longrightarrow T_\Sigma(V)$ such that for each $x \in \mathit{vars}(t = t')$ we have $x\mu = x\theta_i\,\rho$.

B-Unification

The standard unification problem is to try to unify two terms. But we have already encountered situations, such as the relation $\longrightarrow_{\vec{E}/B}$, in which it is very useful to deal not with terms, but with equivalence classes of terms modulo some equational axioms B.

Therefore, it is natural, given a set of equational axioms B, such as the associativity, commutativity, and identity of some operators, to generalize the unification problem to the following B-unification problem: given an equation $t = t'$ are there substitutions θ such that

$$t\theta =_B t'\theta.$$

B-Unification (II)

For B any combination of associativity, commutativity, and identity axioms, there are known algorithms that can find a family of most general unifiers for each given unification problem $t = t'$. However, for the case of associativity alone, or of associativity and identity alone, this family of most general unifiers may be infinite.

In particular, for Σ a finite signature, if we choose B to be any combination of associativity, commutativity and identity axioms for different (subsort-overloaded) binary operators in Σ, except associativity without commutativity, there is indeed an algorithm that, given an B-unification problem, either declares the problem unsolvable, or finds a finite set of most general unifiers solving it. Such a B-unification algorithm is used by the Church-Rosser Checker.

More on Unification

So far we have said nothing about unification algorithms, that can effectively find a set of most general unifiers or declare the corresponding problem unsolvable.

Unification is indeed a vast research area, and the more we can do in this course is to give a flavor for the key ideas. This can be done quite well by considering the simplest version of the unification problem, for which, if the given equation has a solution, then it has a unique most general unifier.

More on Unification (II)

This simplest version is the case of a sensible many-sorted signature Σ without ad-hoc overloading.

The key idea of a unification algorithm is to transform the original equation we want to solve into a set of equations equivalent to the original equation, in the sense that both sets have the same solutions.

We then stop either with failure, or with a set of equations in solved form, that is, equations having the shape, $\{x_1 = t_1, \ldots, x_n = t_n\}$, where the x_i do not appear in the t_j. But this is just another garb for a substitution $\theta = \{(x_1, t_1), \ldots, (x_n, t_n)\}$.

The Unification Algorithm

We can describe the unification algorithm, à la Martelli-Montanari, as a set of inference rules, that transform a set of equations E into another set of equations that is equivalent to it from the solvability point of view, or into the constant failure. The following inference rules assume a many-sorted signature, make the equality symbol commutative, and use a global set V of variables:

- Delete:

$$\frac{\{E,\ t = t\}}{\{E\}}$$

- Decompose:

$$\frac{\{E,\ f(t_1,\dots,t_n) = f(t'_1,\dots,t'_n)\}}{\{E,\ t_1 = t'_1,\dots,t_n = t'_n)\}}$$

The Unification Algorithm (II)

- Conflict:

$$\frac{\{E,\ f(t_1,\ldots,t_n) = g(t'_1,\ldots,t'_m)\}}{\text{failure}}$$

  if $f \neq g$

- Coalesce:

$$\frac{\{E,\ x = y\}}{\{E\{x \mapsto y\},\ x = y\}}$$

  if $x, y \in \text{vars}(E),\ x \neq y$

- Check:

$$\frac{\{E,\ x = t\}}{\text{failure}}$$

  if $x \in \text{vars}(t),\ x \neq t$

The Unification Algorithm (III)

- Eliminate:
$$\frac{\{E,\ x = t\}}{\{E\{x \mapsto t\},\ x = t\}}$$
if $x \notin \mathrm{vars}(t),\ t \notin V,\ x \in \mathrm{vars}(E)$.

We can illustrate the use of these inference rules by finding the most general unifier for a relatively simple, yet nontrivial, unification problem, namely, solving the equation,

$$f(g(x, h(y)), z) = f(z, g(k(u), v))$$

for which the above rules give us the following transformations:

$\{f(g(x, h(y)), z) = f(z, g(k(u), v))\} \longrightarrow$ (Decompose)

The Unification Algorithm (IV)

$\{g(x,h(y)) = z,\ z = g(k(u),v)\} \longrightarrow$ (Eliminate)

$\{g(x,h(y)) = g(k(u),v),\ z = g(k(u),v)\} \longrightarrow$ (Decompose)

$\{x = k(u),\ v = h(y),\ z = g(k(u),v)\} \longrightarrow$ (Eliminate)

$\{x = k(u),\ v = h(y),\ z = g(k(u),h(y))\}$,

which is the desired most general unifier, yielding the identity,

$f(g(k(u),h(y)),g(k(u),h(y))) = f(g(k(u),h(y)),g(k(u),h(y)))$.

Unification Modulo Commutativity

To illustrate the case of B-unification in a many-sorted signature Σ, let us assume that $B = Comm$ is a collection of commutativity axioms for some binary symbols $\Sigma_{comm} \subseteq \Sigma$. The inference rules for unification modulo commutativity are:

- Delete:

$$\frac{\{E,\ t = t\}}{\{E\}}$$

- Decompose: $(f \in (\Sigma - \Sigma_{comm}))$

$$\frac{\{E,\ f(t_1, \ldots, t_n) = f(t'_1, \ldots, t'_n)\}}{\{E,\ t_1 = t'_1, \ldots, t_n = t'_n)\}}$$

Unification Modulo Commutativity (II)

- Decompose-C: $(f \in \Sigma_{comm})$

$$\frac{\{E,\ f(t_1, t_2) = f(t'_1, t'_2)\}}{\{E,\ t_1 = t'_1, t_2 = t'_2)\}\ \vee\ \{E,\ t_1 = t'_2, t_2 = t'_1)\}}$$

- Conflict:

$$\frac{\{E,\ f(t_1, \ldots, t_n) = g(t'_1, \ldots, t'_m)\}}{\text{failure}}$$

  if $f \neq g$

- Coalesce:

$$\frac{\{E,\ x = y\}}{\{E\{x \mapsto y\},\ x = y\}}$$

  if $x, y \in \text{vars}(E),\ x \neq y$

Unification Modulo Commutativity (III)

- Check:
$$\frac{\{E,\ x = t\}}{\text{failure}}$$
if $x \in \text{vars}(t),\ x \neq t$

- Eliminate:
$$\frac{\{E,\ x = t\}}{\{E\{x \mapsto t\},\ x = t\}}$$
if $x \notin \text{vars}(t),\ t \notin V,\ x \in \text{vars}(E)$.

Note that now, because of Rule Decompose-C, there can be several solutions to a unification problem. Also, we define failure as an identity element for $\_ \vee \_$.

We can illustrate the use of these rules by finding the most general unifiers modulo commutativity when $\Sigma_{comm} = \{g\}$.

Unification Modulo Commutativity (IV)

Let us apply these rules to solve the equation,

$$f(g(h(y),x),z) = f(z,g(k(u),v))$$

$\{f(g(h(y),x),z) = f(z,g(k(u),v))\} \longrightarrow$ (Decompose)

$\{g(h(y),x) = z,\ z = g(k(u),v)\} \longrightarrow$ (Eliminate)

$\{g(h(y),x) = g(k(u),v),\ z = g(k(u),v)\} \longrightarrow$ (Decompose-C)

$\{x = v,\ k(u) = h(y),\ z = g(k(u),v)\} \vee \{x = k(u),\ v = h(y),\ z = g(k(u),v)\} \longrightarrow$ (**Conflict** $\vee$ **Eliminate**)

failure $\vee$ $\{x = k(u),\ v = h(y),\ z = g(k(u),h(y))\} =$
$\{x = k(u),\ v = h(y),\ z = g(k(u),h(y))\}$

applying the resulting unifier we obtain the identity,

$f(g(h(y),k(u)),g(h(y),k(u))) =_{comm} f(g(k(u),h(y)),g(k(u),h(y)))$.

Where to Go from Here

We can only sketch how to go from here to more general unification algorithms, such as B-unification in an order-sorted signature Σ.

First of all, note that the presence of overloading and subsorts will typically move us from a single most general unifier to a finite set of them. This is because of the presence of subsort overloaded operators, which may lead to several different solutions. Note also that, in the presence of subsorts, even apparently innocent equations such as $x : s = y : s'$ may lead to failure, because the sorts s and s' may not have any common subsort. For example, in an `INT` specification, an equation `X = Y`, with `X` of sort `NzNat`, and `Y` of sort `NzNeg` will fail.

Where to Go from Here (II)

Chapter 15.1 of the Maude book gives a detail presentation of the inference rules for order-sorted C-unification and gives an implementation that you can use in Maude for experimentation.

The latest version of Maude provides general order-sorted unification algorithm modulo any combination of C and/or A and/or U axioms. Since the set of A-unifiers of an equation can be infinite, Maude provides a finite set and a warning if more solutions may exist.

The Church-Rosser Checker uses unification modulo any combinations of associativity, commutativity, and identity axioms; but may not generate all critical pairs [giving a warning] for associativity without commutativity axioms.

Where to Go from Here (III)

For a survey of unification algorithms modulo axioms see:

J.-P. Jouannaud and C. Kirchner, "Solving Equations in Abstract Algebras," in J.-L. Lassez and G.Plotkin, eds., Computational Logic: Essays in Honor of Alan Robinson.

For order-sorted B-unification see:

J. Meseguer, J.A. Goguen, and G. Smolka, "Order-Sorted Unification," J. Symbolic Computation, Volume 8, 1989, pages 383–413.

J. Hendrix and J. Meseguer, "Equational Order-Sorted Unification Revisited," Electr. Notes Theor. Comput. Sci., Vol. 290, 2012, pages 37–50.

What Are Critical Pairs?

Theorem: Let $(\Sigma, B \cup E)$ be an order-sorted equational theory, with Σ B-preregular, $\vec{E}$ sort-decreasing, and $\longrightarrow_{\vec{E}/B}$ terminating. Let $V = vars(E)$, and $\gamma : V \to V'$ a bijective, sort-preserving renaming of variables with $V \cap V' = \emptyset$. Then, $\vec{E}$ is confluent modulo B iff, for each pair of equations[a] $u = v$ in E and $u' = v'$ in $E\gamma$ (including $(u' = v') \equiv (u\gamma = v\gamma)$) for each nonvariable position p in u, and for each most general order-sorted B-unifier θ such that $u_p\theta =_B u'\theta$, we have,

$$(\flat) \quad v\theta \downarrow_{\vec{E}/B} u[v']_p\theta.$$

The corresponding equations $v\theta = u[v']_p\theta$, are called the critical pairs of the equations E modulo B.

[a]If B contains associativity axioms, equations E should first be generalized to B-match "with extension," Cf. §4.8 in All About Maude.

What Are Critical Pairs? (II)

Proof: (For $B = \emptyset$). We had already reduced checking confluence to checking that, for each pair of equations $(u = v) \in E$ and $(u' = v') \in E\gamma$, for each nonvariable position p in u, and for each order-sorted unifier μ such that $u_p\mu = u'\mu$, we have,

$$(\flat) \quad v\mu \downarrow_{\vec{E}} u[v']_p\mu.$$

But if $\mathit{Unif}(u_p = u') = \{\theta_1, \ldots, \theta_n\}$ is the set of most general order-sorted B-unifiers for the equation $u_p = u'$, then we can find a $\theta_i \in \mathit{Unif}_B(u_p = u')$ and a substitution ρ such that $\mu = \theta_i\,\rho$.

Since we know that there is a w such that $v\theta_i \to^*_{\vec{E}} w$ and $u[v']_p\theta_i \to^*_{\vec{E}} w$, we will be done if we prove the following:

What Are Critical Pairs? (III)

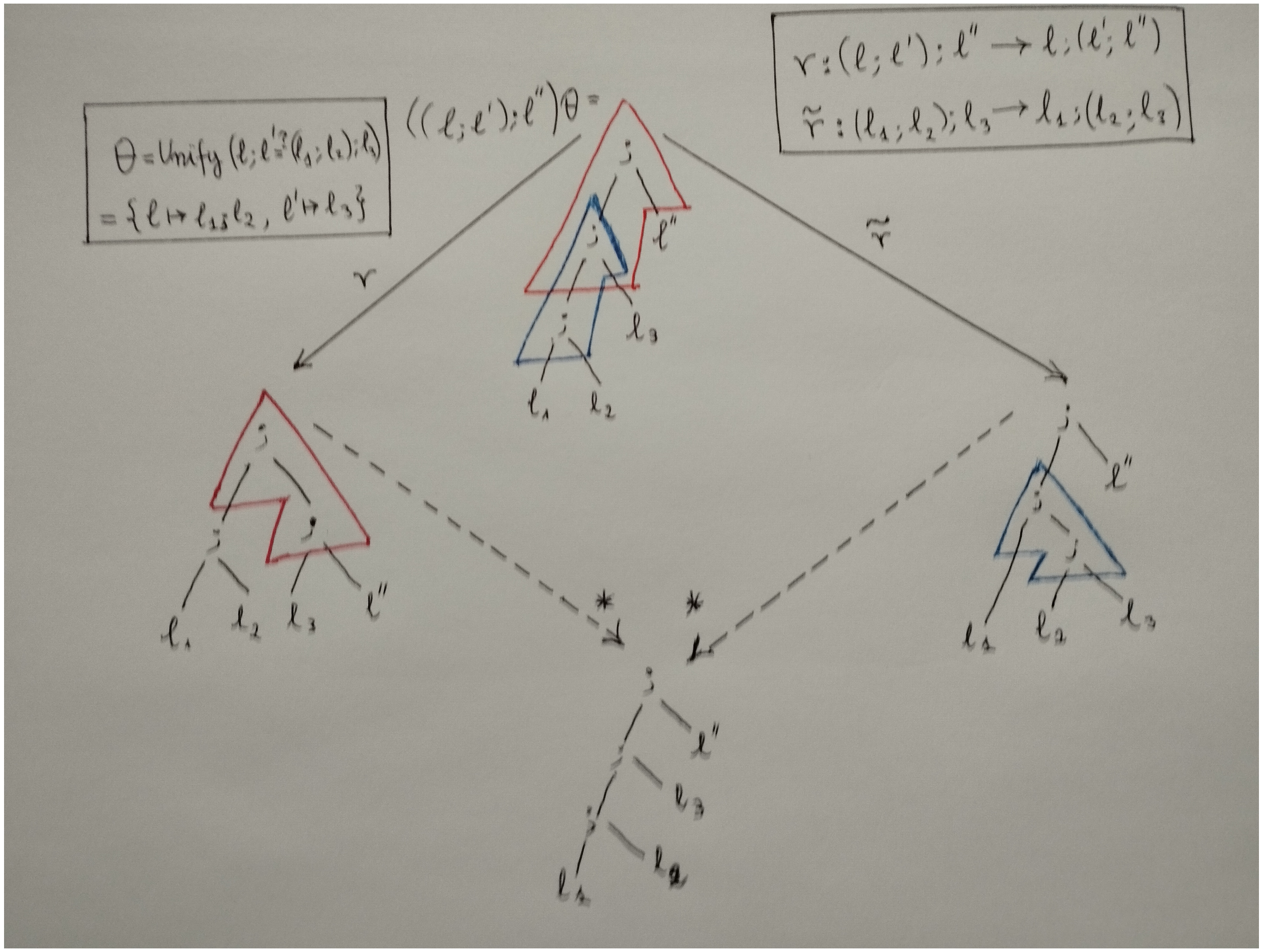
Substitution Lemma: If $t \xrightarrow{*}_{\vec{E}} t'$ and ρ is a substitution, then $t\rho \xrightarrow{*}_{\vec{E}} t'\rho$.

Proof: It is enough to prove the case for $t \longrightarrow_{\vec{E}} t'$, since then the case $t \xrightarrow{*}_{\vec{E}} t'$ follows easily by induction on the number of steps. But $t \longrightarrow_{\vec{E}} t'$ means that there is an equation $(u = v) \in E$, position q and a substitution θ such that $t = t[u\theta]_q$ and $t' = t[v\theta]_q$.

But note that, by the definition of the function $\_\rho$, we can easily prove that we have, $t\rho = t\rho[u\theta\rho]_q$, and $t'\rho = t\rho[v\theta\rho]_q$. Therefore, $t\rho \longrightarrow_{\vec{E}} t'\rho$ holds by applying $u = v$ at position q with substitution $\theta; \_\rho$, as desired. q.e.d.

This finishes the proof of the Theorem (for $B = \emptyset$). q.e.d.

Critical Pair for Self-Overlap of Associativity Rule



$\theta = \mathrm{Unify}(\ell;\ell' \overset{?}{=} (\ell_1;\ell_2);\ell_3)$
$= \{\ell \mapsto \ell_1;\ell_2,\ \ell' \mapsto \ell_3\}$
$((\ell;\ell');\ell'')\theta =$
$r : (\ell;\ell');\ell'' \to \ell;(\ell';\ell'')$
$\tilde{r} : (\ell_1;\ell_2);\ell_3 \to \ell_1;(\ell_2;\ell_3)$
r
$\tilde{r}$
ℓ_1
ℓ_2
ℓ_3
ℓ''
$*$
$*$

In Summary

What the Church-Rosser Checker does is:

- it checks that the oriented equations $\vec{E}$ are sort-decreasing;
- it forms all the critical pairs for the oriented equations $\vec{E}$ and tries to join them;
- it returns as proof obligations those equation specializations that it could not prove sort-decreasing, and those simplified critical pairs that it could not join.

The arguments in Lecture 7 and in this lecture have shown that this method is correct for checking confluence, under the termination assumption.