Formal Methods: Lecture 7

José Meseguer

Computer Science Department
University of Illinois at Urbana-Champaign

The Quest for Critical Pairs

Given a Maude module $\texttt{fmod}(\Sigma, E)\texttt{endfm}$, we can summarize the main ideas of Lecture 7 by saying that, under the assumption that the equations E are terminating,

> We have reduced checking confluence to checking joinability of all possible nested simplifications with overlap.

The quest for the desired critical pairs, and the associated automated way of checking confluence, proceeds by means of several subsequent reductions of the original confluence problem to checking joinability of a finite number of pairs of terms.

Nested Simplifications with Overlap

Recall the joinablilty problem for nested simplifications with overlap to which we have reduced the confluence property (assuming termination). We have equations $u = v$ and $u' = v'$ in E (**Note**: $u' = v'$ could be the same equation $u = v$ considered twice!) and should consider terms t with positions π and $\pi.\nu$, and a substitution θ such that:

1. $t/\pi = \overline{\theta}(u)$;
2. ν is a nonvariable position in u, and $t/\pi.\nu = \overline{\theta}(u)/\nu = \overline{\theta}(u')$.

This formulation of the overlap situation assumes that the variables in u and u' are disjoint, or have been made so by renaming their variables, even in the case when the equation $u = v$ is considered twice (self-overlap).

Context-Free Nested Simplifications

The joinabiliy problem for nested simplifications with overlap then consists in showing,

$$(*) \quad t[\pi \leftarrow \overline{\theta}(v)] \downarrow_E t[\pi \leftarrow \overline{\theta}(u)[\nu \leftarrow \overline{\theta}(v')]].$$

Our next reduction of the problem comes from the observation that the context t in which all this happens is irrelevant. That is, we can reduce the problem to that of checking joinability for all context-free nested simplifications of the form,

$$(\flat) \quad \overline{\theta}(v) \downarrow_E \overline{\theta}(u)[\nu \leftarrow \overline{\theta}(v')].$$

Context-Free Nested Simplifications (II)

Of course, $(*) \Rightarrow (\flat)$, but we also have $(\flat) \Rightarrow (*)$, because of the following,

Context Lemma: Let $t = t[\pi \leftarrow u] \in T_\Sigma(X)$, and suppose that we have, $u \xrightarrow{*}_E v$. Then we have, $t[\pi \leftarrow u] \xrightarrow{*}_E t[\pi \leftarrow v]$.

Proof: It is obviously enough to check it for one step ($\longrightarrow_E$), that is, for $u \longrightarrow_E v$. But by sort-decreasingness of $\longrightarrow_E$ we then have a well-formed term $t[\pi \leftarrow v] \in T_\Sigma(X)$, where if the rewriting of u happened at, say, position ν, then the rewriting of $t = t[\pi \leftarrow u]$ now happens at position $\pi.\nu$ and yields, $t[\pi \leftarrow u] \longrightarrow_E t[\pi \leftarrow v]$. q.e.d.

Overlaps as Unification Problems

The next observation is 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 ν in u and a substitution θ such that,

$$(\dagger) \quad \overline{\theta}(u/\nu) = \overline{\theta}(u').$$

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 ν 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, $\overline{\theta}(t) = \overline{\theta}(t')$. The substitution θ is then called a unifier of t and t'.

Since a substitution is nothing but an assignment $\theta : X \longrightarrow T_\Sigma(Y)$ into a term algebra, we can see the unification problem as a special instance of a more general problem, namely, the problem of existential satisfaction of an equation $t = t'$ in an algebra A, which is often expressed as the problem of solving equations in an algebra A.

Solving Equations = Existential Satisfaction

Given a Σ-algebra A, and given a Σ-equation $t = t'$, say, with variables X, by a solution to the equation $t = t'$ in A we mean an assignment $a : X \longrightarrow A$ such that, $\overline{a}(t) = \overline{a}(t')$.

But this is just another way of speaking about a solution to the existential satisfaction problem, $A \models (\exists X)\, t = t'$, where, by definition,

$A \models (\exists X)\, t = t' \quad \Leftrightarrow \quad$ there is an assignment a, with $\overline{a}(t) = \overline{a}(t')$.

Unification = Existential Satisfaction

Therefore, choosing V as a family of variables with infinitely many variables of each sort, we can rephrase the problem of unifying t and t' as that of finding a solution to the existential satisfaction problem,

$$T_\Sigma(V) \models (\exists X)\, t = t'.$$

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 $\mu : X \longrightarrow T_\Sigma(V)$ such that $\overline{\mu}(t) = \overline{\mu}(t')$, we can then always find a θ_i and a substitution $\rho : V \longrightarrow T_\Sigma(V)$ such that, $\overline{\mu} = \overline{\theta_i}; \overline{\rho}$.

A-Unification

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

Therefore, it is natural, given a set of equational axioms A, to generalize the unification problem to the following A-unification problem: given an equation $t = t'$ find a solution to the existential satisfaction problem,

$$T_{\Sigma/A}(V) \models (\exists X)\, t = t'.$$

A-Unification (II)

That is to say, we need to find an assignment $a : X \longrightarrow T_{\Sigma/A}(V)$ such that $\overline{a}(t) = \overline{a}(t')$. But since the elements of $T_{\Sigma/A}(V)$ are A-equivalence classes of terms in $T_\Sigma(V)$, by choosing a representative in each equivalence class we can reduce the A-unification problem to the one of finding a substitution $\theta : X \longrightarrow T_\Sigma(V)$ such that,

$$\overline{\theta}(t) \equiv_A \overline{\theta}(t').$$

A-Unification (III)

For an arbitrary equational theory A, the A-unification problem is in general undecidable. But for some frequently used sets of axioms A there are known algorithms that can find a finite family of most general unifiers for each given unification problem $t = t'$.

In particular, under very reasonable assumptions about the underlying signature Σ (such as finiteness) if we choose $A = C$, where C is a collection of commutativity axioms for different (subsort-overloaded) binary operators in Σ, there is indeed an algorithm that, given a C-unification problem, either declares the problem unsolvable, or finds a finite set of most general unifiers solving it. Such a C-unification algorithm is indeed 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 signature Σ without ad-hoc overloading, and such that its poset of sorts $(S, <)$ has $<= \emptyset$ (no subsorts). Such a signature is called a many-sorted signature, whereas the general signatures are usually called order-sorted.

The key idea of the 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, $\{x_1 = t_1, \ldots, x_n = t_n\}$. But this is just another garb for a substitution $\theta : x_i \mapsto t_i$.

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 inference rules assume that equality is commutative and are the following:

- **Delete:**

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

- **Decompose:**

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

The Unification Algorithm (II)

- **Conflict:**

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

if $f \neq g$

- **Coalesce:**

$$\frac{\{E,\ x = y\}}{\{E(x/y),\ x = y\}}$$

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

- **Check:**

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

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

The Unification Algorithm (III)

- **Eliminate:**

$$\frac{\{E,\ x = t\}}{\{E(x/t),\ x = t\}}$$

if $x \notin \mathsf{vars}(t),\ t \notin V,\ x \in \mathsf{vars}(E)$.

We can illustrate the use of these 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,

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

Where to Go from Here

We can only sketch how to go from here to more general unification algorithms, such as A-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)

Detailed inference rules for order-sorted C-unification, which is implemented in the Curch-Rosser Checker can be found in,

F. Durán and J. Meseguer, "A Church-Rosser Checker Tool for Maude Specifications," 2000, in `http://maude.csl.sri.com`.

Where to Go from Here (III)

A good survey of unification, including different unification algorithms for different theories can be found in,

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.

A general treatment of order-sorted A-unification can be found in,

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

What Are Critical Pairs?

We are now ready to define critical pairs, and prove our main result, justifying the Church-Rosser Checker:

Theorem: Let $(\Sigma, E \cup C)$ be an order-sorted equational theory, where C consists of commutativity axioms for some (subsort-overloaded) binary operators in Σ, and where the equations E are unconditional, with $\longrightarrow_{E/C}$ terminating. Then, E is confluent modulo C iff, for each pair of equations $u = v$ and $u' = v'$ in E (including equations $u = v$ considered twice) for each nonvariable position ν in u, and for each most general C-unifier θ such that $\overline{\theta}(u/\nu) \equiv_C \overline{\theta}(u')$, we have,

$$(\flat) \quad \overline{\theta}(v) \downarrow_{E/C} \overline{\theta}(u)[\nu \leftarrow \overline{\theta}(v')].$$

The corresponding equations $\overline{\theta}(v) = \overline{\theta}(u)[\nu \leftarrow \overline{\theta}(v')]$, are called the critical pairs of the equations E modulo C.

What Are Critical Pairs? (II)

Proof: We had already reduced checking confluence to cheking that, for each pair of equations $u = v$ and $u' = v'$ in E, for each nonvariable position ν in u, and for each C-unifier μ such that $\overline{\mu}(u/\nu) \equiv_C \overline{\mu}(u')$, we have,

$$(\flat) \quad \overline{\mu}(v) \downarrow_{E/C} \overline{\mu}(u)[\nu \leftarrow \overline{\mu}(v')].$$

But if $\{\theta_1, \ldots \theta_n\}$, are the most general C-unifiers for the equation $u/\nu = u'$, then we can find a θ_i and a substitution $\rho : V \longrightarrow T_\Sigma(V)$ such that, $\overline{\mu} = \overline{\theta_i}; \overline{\rho}$.

We are then done if we prove the following (**Note**: the generalization to $\longrightarrow_{E/C}$ is left as an exercise)

What Are Critical Pairs? (III)

Substitution Lemma: If $t \stackrel{*}{\longrightarrow}_E t'$ and ρ is a substitution, then $\overline{\rho}(t) \stackrel{*}{\longrightarrow}_E \overline{\rho}(t')$.

Proof: Note that, since $\overline{\rho}$ is by definition a Σ-homomorphism, we have the identity,

$$\overline{\rho}(t[\pi \leftarrow u]) = \overline{\rho}(t)[\pi \leftarrow \overline{\rho}(u)].$$

Therefore, if $t \longrightarrow_E t'$, then $\overline{\rho}(t) \longrightarrow_E \overline{\rho}(t')$. The reflexive and transitive extension is then trivial. q.e.d.

q.e.d.

Conditional Critical Pairs

If some equations in E are conditional, we should instead consider conditional critical pairs, where if we are checking overlaps of conditional equations $u = v \Leftarrow C_1$ and $u' = v' \Leftarrow C_2$, then the critical pairs take the form,

$$\overline{\theta}(v) = \overline{\theta}(u)[\nu \leftarrow \overline{\theta}(v')] \Leftarrow \overline{\theta}(C_1) \wedge \overline{\theta}(C_2).$$

In Summary

What the Church-Rosser Checker does is:

- it checks that the equations E are sort-decreasing;
- it forms all the critical pairs for the equations 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.