Appendix to Lecture 8: Local Confluence Checking Algorithm

José Meseguer, CS Department, UIUC

This appendix gives a detailed account of the local confluence checking method explained in Lecture 8. It assumes an order-sorted equational theory (Σ, E) whose left-to-right oriented equations define a term rewriting system $(\Sigma, \vec{E})$, where the rules $\vec{E}$ are assumed terminating, so that if $(\Sigma, \vec{E})$ is locally confluent, then it is also confluent. I refer to the paper [1] for a detailed account of how the case of confluence checking for an equational theory (Σ, E) generalizes to checking local confluence for an equational theory $(\Sigma, E \cup B)$ with associated rewrite theory $(\Sigma, B, \vec{E})$ with $\to_{\vec{E}}$ terminating, replacing syntacting unification by B-unification.

Definition. Under the above assumptions about $(\Sigma, \vec{E})$, assume without loss of generality that the rules $\vec{E} = \{u_1 \to v_1, \ldots, u_n \to v_n\}$ do not share any variables, i.e., $vars(u_i \to v_i) \cap vars(u_j \to v_j) = \emptyset$ for all i, j s.t. $1 \leq i < j \leq n$. Assume, further, that for each $u_i \to v_i$ in $\vec{E}$ we have chosen a *variable-renamed* version of it $u_i' \to v_i'$ such that $vars(u_i \to v_i) \cap vars(u_i' \to v_i') = \emptyset$. The the set $CP(\Sigma, \vec{E})$ of *critical pairs* of $(\Sigma, \vec{E})$ is then defined as the union:

$$CP(\Sigma, \vec{E}) = (\bigcup_{1 \leq i < j \leq n} CP(u_i \to v_i, u_j \to v_j) \cup CP(u_j \to v_j, u_i \to v_i)) \;\cup\; \bigcup_{1 \leq i \leq n} CP(u_i \to v_i, u_i' \to v_i')$$

where for any Σ-rules $u \to v$ and $w \to q$ not sharing any variables, the set of critical pairs $CP(u \to v, w \to q)$ is defined as the set of pairs of Σ-terms:

$$CP(u \to v, w \to q) = \{(v\theta, (u[q]_p)\theta) \mid p \in Pos_\Sigma(u) \;\wedge\; \theta \in Unif_\Sigma(u|_p = w)\}$$

where, $Unif_\Sigma(u|_p = w)$ denotes the set of order-sorted (according to the signature Σ) unifiers of the equation $u|_p = w$ and, by definition, the set $Pos_\Sigma(u)$ of *non-variable positions* of u is the set

$$Pos_\Sigma(u) = \{p \in Pos(u) \mid u|_p \notin vars(u)\}$$

The main theorem proved in Lecture 8 is then:

Theorem. Under the above assumptions on (Σ, E), the term rewriting system $(\Sigma, \vec{E})$ is locally confluent (and therefore confluent by the assumption that $\to_{\vec{E}}$ is terminating) iff

$$\forall (t, t') \in CP(\Sigma, \vec{E}),\; t \downarrow_{\vec{E}} t'$$

Remark 1. A necessary condition for a critical pair $(v\theta, (u[q]_p)\theta)$ to exist in $CP(u \to v, w \to q)$ is that $top(w) = top(u|_p)$, where for any non-variable term $t = f(t_1, \ldots, t_n)$, $top(t) = f$. This is clearly a necessary condition for $Unif_\Sigma(u|_p = w)$ to be non-empty, since if $top(w) \neq top(u|_p)$ there can be no unifiers of the equation $u|_p = w$. In practice this makes it easy to detect that some critical pairs do not exist. For example, $CP(f(h(x), y) \to k(x, y), g(z) \to l(z)) = \emptyset$, because $\{f, h\} \cap \{g\} = \emptyset$. That is, if $top(w)$ does not appear anywhere in u, then there are no critical pairs in $CP(u \to v, w \to q)$.

Remark 2. In the case of the set $CP(u_i \to v_i, u_i' \to v_i')$ of *self-overlaps* of a rule $u_i \to v_i$ with a renamed copy of itself $u_i' \to v_i'$, there is always a *trivial critical pair* at the top position $p = \varepsilon$, namely, (v_i', v_i'), for which $v_i' \downarrow_{\vec{E}} v_i'$ trivially holds. This is because $Unif_\Sigma(u_i = u_i') = \{\alpha\}$, with α the variable-renaming substitution such that $u_i' \to v_i' = u_i\alpha \to v_i\alpha$. Therefore, when generating $CP(u_i \to v_i, u_i' \to v_i')$ we can always *discard* the case of the empty position $p = \varepsilon$.

Remark 3. For a Maude functional module `fmod` (Σ, E) `endfm` one can automate the checking of the joinability $t \downarrow_{\vec{E}} t'$ of a critical pair $(t, t') \in CP(\Sigma, \vec{E})$ by giving the Maude command:

`red t == t'` so that if the result is `true`, then $t \downarrow_{\vec{E}} t'$. This is because `red t == t'` returns `true` iff $t!_{\vec{E}} = t'!_{\vec{E}}$. The only cavet is that if, say, we have a critical pair $(x + s(y), s(x + y))$ with, say, x of sort `NzNat` and y of sort `Nat` variables *not declared* in `fmod` (Σ, E) `endfm` then the reduce command should declare the variables x and y *on the fly* as:

```
red x:Nat + s(y:NzNat) == s(x:Nat + y:NzNat).
```

References

[1] F. Durán and J. Meseguer. On the Church-Rosser and coherence properties of conditional order-sorted rewrite theories. *J. Algebraic and Logic Programming*, 81:816–850, 2012.