6.2 An Improved Completion Algorithm

Donnerstag, 17. Dezember 2015 13:14

Drawback of BASIC_COMPLETION: rules are only added, but never removed or simplified.

$\Rightarrow$ • TRSs get very large

• One has to construct critical pairs with huge amounts of rules.

• This increases the probability of failing (by obtaining a critical pair that cannot be oriented).

Idea: When obtaining a new rule by completion, use this rule to simplify (and maybe even remove) previous rules.

E.g.: if a new rule $\ell \to r$ is obtained, apply this rule to simplify the left- and right-hand sides of old rules $s \to t$. If this simplification reduces s and t to the same term, then remove $s \to t$.

Ex. 6.2.1 Group Example

$\mathcal{R}_0 = \{ (G1), (G2), (G3) \}$

$\mathcal{R}_1 = \{ (G1), (G2), (G3), (G4) \}$

(G4) and (G2) have non-joinable critical pair:

$$\begin{tikzcd}
 & f(f(x,e),i(e)) \arrow[dl, "(G2)"'] \arrow[dr, "(G4)"] & \\
f(x,i(e)) & & x
\end{tikzcd}$$

$f(x,i(e)) \to x \quad (G5)$ This holds because $i(e) \equiv e$.

(G4) and (G5) have a non-joinable critical pair:

$$\begin{tikzcd}
 & f(f(x,i(e)),i(i(e))) \arrow[dl, "(G4)"'] \arrow[dr, "(G5)"] & \\
x & & f(x,i(i(e)))
\end{tikzcd}$$

$f(x,i(i(e))) \to x \quad (G6)$ This holds because $i(e) = e$.

(G3) and (G6) have a non-joinable critical pair:

$$\begin{tikzcd}
 & f(i(e),i(i(e))) \arrow[dl, "(G3)"'] \arrow[dr, "(G6)"] & \\
e & & i(e)
\end{tikzcd}$$

$i(e) \to e \quad (G7)$

(G5) and (G6) were useful in order to derive (G7).
But now they aren't needed anymore and can be removed:

$$\begin{array}{l}
\underbrace{f(x,i(e))} \to x \\
\downarrow (G7) \\
f(x,e) \\
\downarrow (G2) \\
x
\end{array}$$

(G5) and similarly for (G6).

We now introduce an improved approach for completion: Define a set of transformation rules for completion, but we do not fix the strategy for their application. By using different such strategies, one obtains different completion algorithms. These algorithms are all sound (if the strategy satisfies certain conditions).

The transformation rules operate on pairs $(\mathcal{E}, \mathcal{R})$ where $\mathcal{E}$ is a (finite) set of equations and $\mathcal{R}$ is a (finite) TRS.

In the beginning: $\mathcal{R} = \emptyset$ and $\mathcal{E}$ is the initial set of equations

In the end: $\mathcal{E} = \emptyset$ and $\mathcal{R}$ is a convergent TRS, equivalent to the original set of equations

If a transformation step modifies $(\mathcal{E}, \mathcal{R})$ to $(\mathcal{E}', \mathcal{R}')$ (denoted $(\mathcal{E}, \mathcal{R}) \vdash_c (\mathcal{E}', \mathcal{R}')$), then it is guaranteed that $\longleftrightarrow^*_{\mathcal{E} \cup \mathcal{R}} = \longleftrightarrow^*_{\mathcal{E}' \cup \mathcal{R}'}$.

$(\mathcal{E}, \emptyset) \vdash_c \dots \vdash_c (\emptyset, \mathcal{R})$ is a successful derivation.

As with BASIC_COMPLETION, we fix a reduction order $\succ$ and ensure that $\mathcal{R}$ only contains rules $l \to r$ with $l \succ r$. $\curvearrowright$ Termination of $\mathcal{R}$ is guaranteed for all pairs $(\mathcal{E}, \mathcal{R})$.

Def 6.22 (Transformation Rules for Completion)
Let $\mathcal{E}$ be a set of equations, $\mathcal{R}$ be a TRS, and $\succ$ be a reduction order. Then we define the following transformation rules on pairs $(\mathcal{E}, \mathcal{R})$: (see slide)

Generate: generates new equations from critical pairs

Orient: turns equations into terminating rules

Delete: deletes trivial equations

Reduce Equation: simplify equations by rules

BASIC_COMPLETION can be obtained by these 4 rules, using a suitable strategy:

1. Apply "Delete" as long as possible.
2. Apply "Orient" —— " ————.
3. If afterwards $\mathcal{E} \neq \emptyset$, then stop with failure.
4. Apply "Generate" until all critical pairs of $\mathcal{R}$ have been created.
5. Apply "Reduce Equation" as long as possible.
6. Apply "Delete" as long as possible.
7. If afterwards $\mathcal{E} = \emptyset$, then stop with success

and return $\mathcal{R}$.

8. Goto Step 2.

Now we extend our transformation rules by 2 additional rules in order to simplify/modify $\mathcal{R}$.

Reduce Right: The TRS remains terminating:

We have $s > t$ and $t > v$

$\rightsquigarrow$ $s > v$.

Reduce Left: If $\mathcal{R} \cup \{s \to t\}$ was terminating and $s \to_{\mathcal{R}} u$, then $\mathcal{R} \cup \{u \to t\}$ could still be non-terminating.

$\Rightarrow$ do not add a rule $u \to t$ but add an equation $u \equiv t$.

Moreover, we require that $s \to_{\mathcal{R}} u$ is done by a rule $l \to r$ where s does not match any subterm of l:

$$\mathcal{R} = \{ \underbrace{f(x,x)}_{s} \to \underbrace{x}_{t}, \underbrace{f(x,y)}_{l} \to \underbrace{x}_{r} \}$$

l matches s, but s does not match l

$\Rightarrow$ $l \to r$ can be used to simplify

$s \to t$

We result in the equation:

$\underbrace{x}_{u} \equiv \underbrace{x}_{t}$ which can be deleted afterwards

$\mathcal{R} = \{ \underbrace{f(x,y)}_{s} \to \underbrace{x}_{t}, \underbrace{f(x,y)}_{l} \to \underbrace{y}_{r} \}$

Now $s = l$, i.e., l matches s, but s also matches l.

Now we may not use $l \to r$ to replace $s \to t$ by $\underbrace{y}_{u} \equiv \underbrace{x}_{t}$.

Reason for this restriction will become clear later.

Advantage of the improved completion algorithm: We can successfully complete TRSs where BASIC_COMPLETION fails:

Ex. 6.2.3

$\mathcal{E} = \{ h(x,y) \equiv f(x), h(x,y) \equiv f(y), g(x,y) \equiv h(x,y), g(x,y) \equiv a \}$

Let $\succ$ be the RPOS with precedence $g \sqsupset h \sqsupset f \sqsupset a$.

$(\mathcal{E}, \emptyset) \vdash_c^* (\emptyset, \mathcal{R})$ with

$\mathcal{R} = \{ h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), g(x,y) \to a \}$

Now we have to create all critical pairs:

$$(\emptyset, \mathcal{R}) \vdash_c^* (\{f(x) \equiv f(y), h(x,y) \equiv a\}, \mathcal{R})$$

$$\vdash_c (\{f(x) \equiv f(y), f(x) \equiv a\}, \mathcal{R})$$

$$\vdash_c (\{f(x) \equiv f(y)\}, \mathcal{R} \cup \{f(x) \to a\})$$

Here, BASIC_COMPL. would fail, because $f(x) \not\succ f(y)$, $f(y) \not\succ f(x)$

$$\vdash_c^2 (\{a \equiv a\}, \mathcal{R} \cup \{f(x) \to a\})$$

$$\vdash_c (\emptyset, \mathcal{R} \cup \{f(x) \to a\})$$

$$\vdash_c^* (\emptyset, \{h(x,y) \to a, g(x,y) \to a, f(x) \to a\})$$

For correctness of the improved completion alg., we have to ensure that the resulting TRS is

- terminating
- confluent
- equivalent to the original set of equations

<u>Lemma 6.24</u> (Termination of resulting TRS)

Let $\succ$ be the red. order used in the completion procedure. If $\ell \succ r$ holds for all $\ell \to r \in \mathcal{R}$ and $(\mathcal{E}, \mathcal{R}) \vdash_c (\mathcal{E}', \mathcal{R}')$, then $\ell \succ r$ also holds for all $\ell \to r \in \mathcal{R}'$.

<u>Proof</u>: Trivial $\boxtimes$

Lemma 6.2.5. (Equivalence of the resulting TRS)

$(\mathcal{E}, \mathcal{R}) \vdash_C (\mathcal{E}', \mathcal{R}')$ implies $\overset{*}{\underset{\mathcal{E} \cup \mathcal{R}}{\longleftrightarrow}} = \overset{*}{\underset{\mathcal{E}' \cup \mathcal{R}'}{\longleftrightarrow}}$.

Proof: Trivial $\boxtimes$

To express the desired confluence property, we introduce the following notions in order to handle both finite and infinite runs of the completion procedure.

Def 6.2.6. (Persistent Equations and Rules)

- For a finite transformation $(\mathcal{E}_0, \mathcal{R}_0) \vdash_C (\mathcal{E}_1, \mathcal{R}_1) \vdash_C \dots$ $\dots \vdash_C (\mathcal{E}_n, \mathcal{R}_n)$, the persistent equations are $\mathcal{E}_\omega = \mathcal{E}_n$ and the persistent rules are $\mathcal{R}_\omega = \mathcal{R}_n$.
- For an infinite transformation $(\mathcal{E}_0, \mathcal{R}_0) \vdash_C (\mathcal{E}_1, \mathcal{R}_1) \vdash_C \dots$, the persistent equations are $\mathcal{E}_\omega = \bigcup_{i \geq 0} \bigcap_{j \geq i} \mathcal{E}_j$ and the persistent rules are $\mathcal{R}_\omega = \bigcup_{i \geq 0} \bigcap_{j \geq i} \mathcal{R}_j$.

So the persistent equations/rules are those that are introduced in some step i and that are never removed again. $(\mathcal{E}_\omega, \mathcal{R}_\omega)$ is the "result" of the completion procedure.

Def 6.27 (Success and Correctness of Completion)
A (finite or infinite) transformation
$(\mathcal{E}_0, \emptyset) \vdash_C (\mathcal{E}_1, \mathcal{R}_1) \vdash_C \dots$ is successful iff
$\mathcal{E}_\omega = \emptyset$ and $\mathcal{R}_\omega$ is a convergent TRS that is equivalent to $\mathcal{E}_0$.
Such a transformation fails iff $\mathcal{E}_\omega \neq \emptyset$.
A completion procedure is an algorithm with input $\mathcal{E}_0$ and $\succ$ which computes a transformation $(\mathcal{E}_0, \emptyset) \vdash_C \dots$.
The completion proc. is correct iff it only computes transformations that are successful or failing.

An unsound completion procedure could apply the rules with a "bad" strategy, such that
$\mathcal{E}_\omega = \emptyset$, but $\mathcal{R}_\omega$ is not confluent.

E.g.: $\mathcal{E}_0 = \{ a \equiv b,\ a \equiv c \}$

$(\mathcal{E}_0, \emptyset) \vdash_C (\emptyset, \{a \to b,\ a \to c\})$.

$\mathcal{E}_\omega = \emptyset$

$R_\omega = \{a \to b, a \to c\}$ is clearly not confluent.

We have to require that the strategy applies "Generate" often enough, i.e., every "required" critical pair must have been generated.

every critical pair of R_ω must have been created at some point

Def 6.2.9 (Fairness)

A (finite or infinite) transformation $(E_0, R_0) \vdash_C (E_1, R_1) \vdash_C \ldots$ is fair

iff $CP(R_\omega) \subseteq \bigcup_{i \geq 0} E_i$ (i.e., every critical pair of the persistent rules appeared in the equations at some point)

Thm 6.2.10 (Completion Theorem)

Every completion procedure where each non-failing transformation is fair is correct.

Proof: highly non-trivial. $\boxed{\text{q}}$

This means that the transformation rules can be applied with arbitrary strategies, as long as fair.

ness is ensured.

Ex 6.2.11 : Now that we know what the "result" of an infinite transformation is, we can show why we need the conditions for the rule "Reduce left".
If we drop the restriction "l cannot be reduced with $s \to t$", then it could happen that $\mathcal{R}_\omega$ are no longer equivalent to the original set of equations.

$\mathcal{E} = \{ f(g(f(x))) \equiv f(g(x)),\ g(g(x)) \equiv g(x) \}$.

$\succ$: RPOS with $f \sqsupset g$

If we drop the restriction on "Reduce left"

$fgf(x) \to fg(x)$ can be simplified to

$fg^2(x) \equiv fg(x)$ which can the be deleted.

(see slide)

Then: $\mathcal{R}_\omega = \{ g(g(x)) \to g(x) \}$ $\leftarrow$ the fgf-rules would not be persistent

$\mathcal{E}_\omega = \emptyset$

Now $\mathcal{R}_\omega$ would not be equivalent to $\mathcal{E}_0$!

Ex. 6.2.12 Now we can do completion for the group example.
Advantage: Rules that were introduced at some

point may be simplified and deleted again afterwards.
(see slide)