Inductive Theorem Proving meets Dependency Pairs[*]

S. Swiderski, M. Parting, J. Giesl, C. Fuhs
LuFG Informatik 2
RWTH Aachen University
Aachen, Germany

P. Schneider-Kamp
Dept. of Mathematics & CS
University of Southern Denmark
Odense, Denmark

Abstract

Current techniques and tools for automated termination analysis of term rewrite systems (TRSs) are already very powerful. However, they fail for algorithms whose termination is essentially due to an *inductive* argument. Therefore, we show how to couple one of the most popular techniques for TRS termination (the *dependency pair* method) with inductive theorem proving. As confirmed by the implementation of our new approach in the tool AProVE, now TRS termination techniques are also successful on this important class of algorithms.

1 Introduction

All current termination tools for TRSs fail on a certain class of natural algorithms like the TRS $\mathcal{R}_{sort}$. It consists of the usual rules for eq and ge on natural numbers represented using 0 and s and of the rules:

$$\begin{array}{rcl}
\mathsf{max}(\mathsf{nil}) & \to & \mathsf{0} \\
\mathsf{max}(\mathsf{co}(x,\mathsf{nil})) & \to & x \\
\mathsf{max}(\mathsf{co}(x,\mathsf{co}(y,xs))) & \to & \mathsf{if}_1(\mathsf{ge}(x,y),x,y,xs) \\[4pt]
\mathsf{if}_1(\mathsf{true},x,y,xs) & \to & \mathsf{max}(\mathsf{co}(x,xs)) \\
\mathsf{if}_1(\mathsf{false},x,y,xs) & \to & \mathsf{max}(\mathsf{co}(y,xs)) \\[4pt]
\mathsf{del}(x,\mathsf{nil}) & \to & \mathsf{nil} \\
\mathsf{del}(x,\mathsf{co}(y,xs)) & \to & \mathsf{if}_2(\mathsf{eq}(x,y),x,y,xs) \\[4pt]
\mathsf{if}_2(\mathsf{true},x,y,xs) & \to & xs \\
\mathsf{if}_2(\mathsf{false},x,y,xs) & \to & \mathsf{co}(y,\mathsf{del}(x,xs)) \\[4pt]
\mathsf{sort}(\mathsf{nil}) & \to & \mathsf{nil} \\
\mathsf{sort}(\mathsf{co}(x,xs)) & \to & \mathsf{co}(\,\mathsf{max}(\mathsf{co}(x,xs)), \\
 & & \quad \mathsf{sort}(\mathsf{del}(\,\mathsf{max}(\mathsf{co}(x,xs)),\mathsf{co}(x,xs)))\,)
\end{array} \qquad (1)$$

For any list xs, $\mathsf{max}(xs)$ computes its maximum and $\mathsf{del}(n,xs)$ deletes the first occurrence of n from xs. To sort a non-empty list ys, $\mathsf{sort}(ys)$ reduces to "$\mathsf{co}(\mathsf{max}(ys),\mathsf{sort}(\mathsf{del}(\mathsf{max}(ys),ys)))$". So $\mathsf{sort}(ys)$ starts with the maximum of ys and then sort is called recursively on the list that results from ys by deleting the first occurrence of its maximum. Note that

every non-empty list contains its maximum. (2)

Hence, the list $\mathsf{del}(\mathsf{max}(ys),ys)$ is shorter than ys and thus, $\mathcal{R}_{sort}$ is terminating. So (2) is the main argument needed for termination of $\mathcal{R}_{sort}$. For automation, one faces two problems:

(a) One has to detect the main argument needed for termination and one has to find out that the TRS is terminating provided that this argument is valid.

(b) One has to prove that the argument detected in (a) is valid.

Here, (2) requires a non-trivial induction proof that relies on the max- and del-rules. Such proofs cannot be done by TRS termination techniques, but they could be performed by state-of-the-art inductive theorem provers. So we would like to use an inductive theorem prover to solve Problem (b) and combine it with TRS termination provers to solve Problem (a). Thus, one has to extend the TRS termination techniques such that they can automatically synthesize an argument like (2) and find out that this argument suffices to complete the termination proof. Sect. 2 gives the main idea for our improvement. To be powerful in practice, we need the new result that innermost termination of many-sorted term rewriting and of unsorted term rewriting is equivalent. We expect that this observation will be useful also for other applications in term rewriting. In Sect. 3, we couple the DP method with inductive theorem proving to show termination of TRSs like $\mathcal{R}_{sort}$ automatically. We implemented our new technique in the termination prover AProVE [1], which we also extended by a small inductive theorem prover. Note that our results allow to couple *any* termination prover implementing DPs with *any* inductive theorem prover. Thus, by using a more powerful theorem prover, one could further increase the power of our new method. In Sect. 4, we evaluate our contributions.

[*]Supported by the DFG Research Training Group 1298 (*AlgoSyn*), the DFG grant GI 274/5-2, and the G.I.F. grant 966-116.6.

2 Many-Sorted Rewriting for Innermost Termination

Here, we only regard the *innermost* rewrite relation $\xrightarrow{\mathsf{i}}$ and prove innermost termination, For large classes of TRSs (e.g., TRSs resulting from programming languages or non-overlapping TRSs like $\mathcal{R}_{sort}$), innermost termination also implies termination. We use the DP method [2] to prove innermost termination. The set $DP(\mathcal{R}_{sort})$ contains, e.g., the following DP, where SORT is the tuple symbol for sort:

$$\mathsf{SORT}(\mathsf{co}(x,xs)) \quad\to\quad \mathsf{SORT}(\mathsf{del}(\mathsf{max}(\mathsf{co}(x,xs)),\mathsf{co}(x,xs))) \qquad (3)$$

Standard techniques suffice to simplify the initial DP problem $(DP(\mathcal{R}_{sort}),\mathcal{R}_{sort})$ to $(\{(3)\},\mathcal{R}'_{sort})$. Here, $\mathcal{R}'_{sort}$ is $\mathcal{R}_{sort}$ without the two sort-rules. Now, however, standard techniques like the reduction pair processor all fail, since termination of this DP problem essentially relies on the inductive argument (2).

To conclude innermost termination of the original TRS, our goal is to prove the absence of infinite innermost $(\mathcal{P},\mathcal{R})$-chains $s_1\sigma \xrightarrow{\mathsf{i}}_{\mathcal{P}} t_1\sigma \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} s_2\sigma \xrightarrow{\mathsf{i}}_{\mathcal{P}} t_2\sigma \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} \ldots$ where $s_i \to t_i$ are variable-renamed DPs from $\mathcal{P}$ and "$\xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}}$" denotes zero or more reduction steps to a normal form. The "classical" reduction pair processor ensures $s_1\sigma \underset{(\sim)}{\succsim} t_1\sigma \succsim s_2\sigma \underset{(\sim)}{\succsim} t_2\sigma \succsim \ldots$ and removes DPs with $s_i\sigma \succ t_i\sigma$.

However, instead of requiring a strict decrease when going from the left-hand side $s_i\sigma$ of a DP to the right-hand side $t_i\sigma$, it would also suffice to require a strict decrease when going from the right-hand side $t_i\sigma$ to the *next* left-hand side $s_{i+1}\sigma$. In other words, if *every* reduction of $t_i\sigma$ to normal form makes the term strictly smaller w.r.t. $\succ$, then we also have $t_i\sigma \succ s_{i+1}\sigma$. Hence, then the DP $s_i \to t_i$ cannot occur infinitely often and can be removed from the DP problem. Our goal is to formulate a new processor based on this idea. We can remove a DP $s \to t$ from a DP problem $(\mathcal{P},\mathcal{R})$ where $\mathcal{P}\cup\mathcal{R} \subseteq \succsim$ if

$$\text{for every normal substitution } \sigma,\ t\sigma \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} q \text{ implies } t\sigma \succ q. \qquad (4)$$

To remove (3) from $(\{(3)\},\mathcal{R}'_{sort})$ with the criterion above, we must use a reduction pair satisfying (4). Here, t is the right-hand side of (3), i.e., $t = \mathsf{SORT}(\mathsf{del}(\mathsf{max}(\mathsf{co}(x,xs)),\mathsf{co}(x,xs)))$.

Towards automation, we will weaken (4) step by step. We currently have to regard substitutions like $\sigma(x) = \mathsf{true}$ and require that $t\sigma \succ q$ holds, although intuitively, here x stands for a number (and not a Boolean value). However, it suffices to consider "well-typed" terms. So far, we regarded untyped TRSs. We now extend the signature $\mathcal{F}$ by (monomorphic) types. For any TRS $\mathcal{R}$ over a signature $\mathcal{F}$, one can use a standard type inference algorithm to compute a typed variant $\mathcal{F}'$ of the original signature $\mathcal{F}$ such that $\mathcal{R}$ is well typed: All terms in $\mathcal{R}$ are well typed w.r.t. $\mathcal{F}'$ and for each $\ell \to r \in \mathcal{R}$, the terms ℓ and r have the same type. By using the most general typed variant, fewer terms are considered to be well typed and (4) has to be required for fewer substitutions σ. E.g., to make $\{(3)\}\cup\mathcal{R}'_{sort}$ well typed, we use:

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
\mathsf{0} & : & \mathsf{nat} & \mathsf{del},\mathsf{co} & : & \mathsf{nat}\times\mathsf{list}\to\mathsf{list} & \mathsf{ge},\mathsf{eq} & : & \mathsf{nat}\times\mathsf{nat}\to\mathsf{bool}\\
\mathsf{s} & : & \mathsf{nat}\to\mathsf{nat} & \mathsf{true},\mathsf{false} & : & \mathsf{bool} & \mathsf{SORT} & : & \mathsf{list}\to\mathsf{tuple}\\
\mathsf{max} & : & \mathsf{list}\to\mathsf{nat} & \mathsf{nil} & : & \mathsf{list} & \mathsf{if}_1,\mathsf{if}_2 & : & \mathsf{bool}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{list}\to\mathsf{list}
\end{array}$$

The following theorem shows that innermost termination is a *persistent* property.

Theorem 1. *Let $\mathcal{R}$ be a TRS over $\mathcal{F}$ and $\mathcal{V}$, let $\mathcal{R}$ be well typed w.r.t. the typed variants $\mathcal{F}'$ and $\mathcal{V}'$. $\mathcal{R}$ is innermost terminating f. all well-typed terms w.r.t. $\mathcal{F}'$ and $\mathcal{V}'$ iff $\mathcal{R}$ is innermost terminating f.* all *terms.*

As noted by [4], this property follows from [5]. To our knowledge, it has never been explicitly stated or applied before. We expect several points where Thm. 1 could simplify innermost termination proofs.[1] Here, we use Thm. 1 to weaken the condition (4) to remove a DP from a DP problem $(\mathcal{P},\mathcal{R})$. Now one can use any typed variant where $\mathcal{P}\cup\mathcal{R}$ is well typed. To remove $s \to t$ from $\mathcal{P}$, it suffices if

$$\text{for every normal substitution } \sigma \textit{ where } t\sigma \textit{ is well typed},\ t\sigma \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} q \text{ implies } t\sigma \succ q. \qquad (5)$$

[1]For example, by Thm. 1 one could switch to termination methods like [3] exploiting sorts.

3 Coupling DPs and Inductive Theorem Proving

Condition (5) is still too hard. We show (in Thm. 2) that one can often relax (5) to ground substitutions σ. Moreover, we require that for all instantiations $t\sigma$ as above, every reduction of $t\sigma$ to its normal form uses a strictly decreasing rule $\ell \to r$ on a monotonic position π. A position π in a term u is *monotonic* w.r.t. $\succ$ iff $t_1 \succ t_2$ implies $u[t_1]_\pi \succ u[t_2]_\pi$ for all t_1, t_2. To remove $s \to t$ from $\mathcal{P}$, now it suffices if

for every normal σ where $t\sigma$ is well-typed and ground, every reduction "$t\sigma \overset{\mathsf{i}}{\to}{}^{!}_{\mathcal{R}} q$" has the form

$$t\sigma \overset{\mathsf{i}}{\to}{}^{*}_{\mathcal{R}} s[\ell\delta]_\pi \overset{\mathsf{i}}{\to}_{\mathcal{R}} s[r\delta]_\pi \overset{\mathsf{i}}{\to}{}^{!}_{\mathcal{R}} q \tag{6}$$

for a rule $\ell \to r \in \mathcal{R}$ where $\ell \succ r$ and where the position π in s is monotonic w.r.t. $\succ$.

A popular class of reduction pairs $(\succsim, \succ)$ is based on *polynomial interpretations*. E.g., consider the interpretation *Pol* with $\mathsf{s}_{Pol} = 1 + x_1$, $\mathsf{co}_{Pol} = 1 + x_1 + x_2$, $\mathsf{SORT}_{Pol} = \mathsf{max}_{Pol} = x_1$, $\mathsf{if}_{1Pol} = 1 + x_2 + x_3 + x_4$, $\mathsf{del}_{Pol} = x_2$, $\mathsf{if}_{2Pol} = 1 + x_3 + x_4$, and $f_{Pol} = 0$, otherwise. For $(\succsim_{Pol}, \succ_{Pol})$, all rules of $\mathcal{R}'_{sort}$ and (3) are weakly decreasing, and (6) is satisfied for the right-hand side t of (3): In every reduction $t\sigma \overset{\mathsf{i}}{\to}{}^{!}_{\mathcal{R}} q$ where $t\sigma$ is well-typed and ground, eventually one has to apply the strictly decreasing rule (1). The del-algorithm uses (1) to delete an element, i.e., reduce the length of the list. Note that (1) is applied within a context $\mathsf{SORT}(\mathsf{co}(..., \dots \mathsf{co}(..., \Box)))$, so (1) is used on a monotonic position w.r.t. $\succ_{Pol}$.

To check automatically whether every reduction of $t\sigma$ to normal form uses a strictly decreasing rule on a monotonic position, we add new rules and function symbols to $\mathcal{R}$ to get an extended TRS $\mathcal{R}^{\succ}$, and for every term u we define a corresponding term $u^{\succ}$. For non-overlapping TRSs $\mathcal{R}$, we then have: If $u^{\succ} \overset{\mathsf{i}}{\to}{}^{*}_{\mathcal{R}^{\succ}} \mathsf{tt}$, then for all q, $u \overset{\mathsf{i}}{\to}{}^{!}_{\mathcal{R}} q$ implies $u \succ q$. To get $\mathcal{R}^{\succ}$, we first introduce a new symbol $f^{\succ}$ for every defined symbol f in $\mathcal{R}$. Now $f^{\succ}(u_1, ..., u_n)$ should reduce to tt in $\mathcal{R}^{\succ}$ whenever the reduction of $f(u_1, ..., u_n)$ in $\mathcal{R}$ uses a strictly decreasing rule on a monotonic position. If $f(\ell_1, ..., \ell_n) \to r \in \mathcal{R}$ was strictly decreasing, then we add $f^{\succ}(\ell_1, ..., \ell_n) \to \mathsf{tt}$ in $\mathcal{R}^{\succ}$. Otherwise, a strictly decreasing rule will be used on a monotonic position to reduce an instance of $f(\ell_1, \dots, \ell_n)$ if this holds for the corresponding instance of the right-hand side r. So then we add $f^{\succ}(\ell_1, ..., \ell_n) \to r^{\succ}$ in $\mathcal{R}^{\succ}$ instead. Next, we define $u^{\succ}$ for any term u over the signature of $\mathcal{R}$. For $u \in \mathcal{V}$, let $u^{\succ} = \mathsf{ff}$. If $u = f(u_1, ..., u_n)$, then we regard the subterms on the monotonic positions of u and check whether their reduction uses a strictly decreasing rule. For any n-ary symbol f, let $mon_{\succ}(f)$ contain those positions from $\{1, \dots, n\}$ where $f(x_1, ..., x_n)$ is monotonic. If $mon_{\succ}(f) = \{i_1, ..., i_m\}$, then for $u = f(u_1, ..., u_n)$ we obtain $u^{\succ} = u^{\succ}_{i_1} \vee ... \vee u^{\succ}_{i_m}$, if f is a constructor. If f is defined, then a strictly decreasing rule could also be applied at the root of u. Hence, then we have $u^{\succ} = u^{\succ}_{i_1} \vee ... \vee u^{\succ}_{i_m} \vee f^{\succ}(u_1, ..., u_n)$. Of course, $\mathcal{R} \subseteq \mathcal{R}^{\succ}$, and $\mathcal{R}^{\succ}$ also contains rules for "$\vee$".

The only rules of $\mathcal{R}'_{sort}$ with a strict decrease are the last two max-rules and (1). So $\mathcal{R}'^{\succ_{Pol}}_{sort}$ contains, among others, the rules given on the right.

$$\begin{array}{rcl}
\mathsf{max}^{\succ}(\mathsf{nil}) & \to & \mathsf{ff} \\
\mathsf{max}^{\succ}(\mathsf{co}(x, \mathsf{nil})) & \to & \mathsf{tt} \\
\mathsf{max}^{\succ}(\mathsf{co}(x, \mathsf{co}(y, xs))) & \to & \mathsf{tt} \\
\mathsf{if}^{\succ}_1(\mathsf{true}, x, y, xs) & \to & \mathsf{max}^{\succ}(\mathsf{co}(x, xs)) \\
\mathsf{if}^{\succ}_1(\mathsf{false}, x, y, xs) & \to & \mathsf{max}^{\succ}(\mathsf{co}(y, xs)) \\
\mathsf{del}^{\succ}(x, \mathsf{nil}) & \to & \mathsf{ff} \\
\mathsf{del}^{\succ}(x, \mathsf{co}(y, xs)) & \to & \mathsf{if}^{\succ}_2(\mathsf{eq}(x, y), x, y, xs) \\
\mathsf{if}^{\succ}_2(\mathsf{true}, x, y, xs) & \to & \mathsf{tt} \\
\mathsf{if}^{\succ}_2(\mathsf{false}, x, y, xs) & \to & \mathsf{del}^{\succ}(x, xs)
\end{array}$$

Now we can again reformulate the condition (6). To remove $s \to t$ from $\mathcal{P}$, now it suffices if

for every normal substitution σ where $t\sigma$ is well typed and ground, we have $t^{\succ}\sigma \overset{\mathsf{i}}{\to}{}^{*}_{\mathcal{R}^{\succ}} \mathsf{tt}$. (7)

To remove (3) using $(\succsim_{Pol}, \succ_{Pol})$, we require "$t^{\succ_{Pol}}\sigma \overset{\mathsf{i}}{\to}{}^{*}_{\mathcal{R}'^{\succ_{Pol}}_{sort}} \mathsf{tt}$", where t is the right-hand side of (3). Here, $t^{\succ_{Pol}}$ is $\mathsf{del}^{\succ_{Pol}}(\mathsf{max}(\mathsf{co}(x, xs)), \mathsf{co}(x, xs))$ when simplifying disjunctions with ff. So to remove (3), we require the following for all σ where $t\sigma$ is well typed and ground: $\mathsf{del}^{\succ_{Pol}}(\mathsf{max}(\mathsf{co}(x, xs)), \mathsf{co}(x, xs))\sigma \overset{\mathsf{i}}{\to}{}^{*}_{\mathcal{R}'^{\succ_{Pol}}_{sort}} \mathsf{tt}$. Note that $\mathsf{del}^{\succ_{Pol}}$ computes the *member*-function, i.e., $\mathsf{del}^{\succ_{Pol}}(x, xs)$ holds iff x occurs in the list xs. Thus, the conjecture is equivalent to the main termination argument (2) for $\mathcal{R}_{sort}$, i.e., that every non-empty list contains its maximum. Hence, we can now use termination arguments like (2) with DPs.

Conditions like (7) correspond to the question if a suitable conjecture is *inductively valid*: For a TRS $\mathcal{R}$ and terms t,s over $\mathcal{F}$ and $\mathcal{V}$, $t = s$ is *inductively valid* ("$\mathcal{R} \models_{ind} t = s$") iff there exist typed variants $\mathcal{F}'$ and $\mathcal{V}'$ such that $\mathcal{R}$, t, s are well typed, and $t\sigma \overset{\mathsf{i}}{\leftrightarrow}{}^{*}_{\mathcal{R}} s\sigma$ holds for all substitutions σ over $\mathcal{F}'$ where $t\sigma$, $s\sigma$ are well-typed ground terms. While undecidable, $\mathcal{R} \models_{ind} t = s$ can often be proved by inductive theorem provers. From (7), we get that in a DP problem $(\mathcal{P},\mathcal{R})$ with $\mathcal{P} \cup \mathcal{R} \subseteq \succsim$, a pair $s \to t$ can be removed from $\mathcal{P}$ if $\mathcal{R}^{\succ} \models_{ind} t^{\succ} = \mathsf{tt}$. Now we formulate a new DP processor based on this criterion. It transforms $(\mathcal{P},\mathcal{R})$ not only into $(\mathcal{P} \setminus \{s \to t\},\mathcal{R})$, but it also generates the problem $(DP(\mathcal{R}),\mathcal{R})$ to ensure innermost termination of $\mathcal{R}$. Moreover, $(\mathcal{P},\mathcal{R})$ must have the *tuple property*, i.e., for all $s \to t \in \mathcal{P}$, $\mathrm{root}(s)$ and $\mathrm{root}(t)$ are tuple symbols and tuple symbols occur nowhere else in $\mathcal{P}$ or $\mathcal{R}$.

Theorem 2 (Induction Processor)**.** *Let $(\succsim,\succ)$ be a reduction pair, let $(\mathcal{P},\mathcal{R})$ have the tuple property, let $\mathcal{R}$ be non-overlapping, and let there be no critical pairs between $\mathcal{R}$ and $\mathcal{P}$. Then Proc is sound:*

$$Proc(\,(\mathcal{P},\mathcal{R})\,) = \begin{cases} \{\ (\mathcal{P} \setminus \{s \to t\},\mathcal{R}),\ \ (DP(\mathcal{R}),\mathcal{R})\ \}, & \textit{if } \mathcal{R}^{\succ} \models_{ind} t^{\succ} = \mathsf{tt} \textit{ and } \mathcal{P} \cup \mathcal{R} \subseteq \succsim \\ \{\ (\mathcal{P},\mathcal{R})\ \}, & \textit{otherwise} \end{cases}$$

In our example, we want to remove the DP (3) from the DP problem $(\{(3)\},\mathcal{R}'_{sort})$. We prove $\mathcal{R}'^{\succ_{Pol}}_{sort} \models_{ind} \mathsf{del}^{\succ_{Pol}}(\mathsf{max}(\mathsf{co}(x,xs)),\mathsf{co}(x,xs)) = \mathsf{tt}$ by an inductive theorem prover. The small induction prover in AProVE, e.g., find this proof automatically. Then the induction processor returns the trivial problem $(\varnothing,\mathcal{R}'_{sort})$ and the easily solved problem $(DP(\mathcal{R}'_{sort}),\mathcal{R}'_{sort})$. Thus, termination of $\mathcal{R}_{sort}$ is verified.

4 Experiments and Conclusion

We introduced a new DP processor for TRSs that terminate because of an inductive property. This property is extracted automatically and transformed into a conjecture that can be verified by current inductive theorem provers. To increase power, we showed that it suffices to prove this conjecture only for well-typed terms, even if the original TRS is untyped. We implemented our contributions in our termination tool AProVE [1] and evaluated it on 19 typical TRSs for classical algorithms where the termination proof requires an inductive argument. So far, all tools in the *Termination Competition* failed on these examples, whereas our new version of AProVE automatically proves termination of 16 of them within a timeout of 1 minute per example. Thus, our method substantially advances automated termination proving, since it allows the first combination of powerful TRS termination tools with inductive theorem provers. For details on our experiments and to run our implementation, we refer to `http://aprove.informatik.rwth-aachen.de/eval/Induction/`. A longer version of this paper appeared in [6].

Acknowledgements. We are very grateful to Aart Middeldorp and Hans Zantema for suggesting the proof idea of Thm. 1 and for pointing us to [5].

References

[1] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the DP framework. In *Proc. IJCAR'06*, LNAI 4130, pp. 281-286, 2006.

[2] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155-203, 2006.

[3] S. Lucas and J. Meseguer. Order-sorted dependency pairs. In *Proc. PPDP'08*, pp. 108-119, ACM Press, 2008.

[4] A. Middeldorp and H. Zantema. Personal communication, 2008.

[5] J. van de Pol. Modularity in many-sorted term rewriting. Master's Thesis, Utrecht University, 1992.

[6] S. Swiderski, M. Parting, J. Giesl, C. Fuhs, and P. Schneider-Kamp. Termination analysis by dependency pairs and inductive theorem proving. In *Proc. CADE'09*, LNAI 5663, pp. 322-338, 2009.