RWTH Aachen

Department of Computer Science
Technical Report

A collection of examples for termination of term rewriting using dependency pairs

Thomas Arts and Jürgen Giesl

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2001-09

RWTH Aachen · Department of Computer Science · September 2001

A collection of examples for termination of term rewriting using dependency pairs

Thomas Arts[1] and Jürgen Giesl[2]

[1] Ericsson, Computer Science Laboratory, Box 1505, 125 25 Älvsjö, Sweden
Email: thomas@cslab.ericsson.se
[2] LuFG Informatik II, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany
Email: giesl@informatik.rwth-aachen.de

Abstract. This report contains a collection of examples to demonstrate the use and the power of the *dependency pair* technique developed by Arts and Giesl. This technique allows automated termination and innermost termination proofs for many term rewrite systems for which such proofs were not possible before.

1 Introduction

In many applications of term rewrite systems (TRSs), *termination* is an important property. A TRS is said to be terminating if it does not allow infinite reductions. Since termination is in general undecidable [HL78], several methods for proving this property have been developed; for surveys see e.g. [Der87,Ste95b]. Practically all known methods that are amenable to automation use *simplification orderings* [Der79,Der87,Ste95b,MZ97].

However, there exist numerous term rewrite systems for which termination cannot be proved by this kind of orderings. For that reason, Arts and Giesl [AG97a,AG97b,AG98,AG00,GA01,GAO01] developed the so-called *dependency pair* approach. Given a TRS, the dependency pair technique automatically generates a set of constraints and the existence of a well-founded (quasi-)ordering satisfying these constraints is sufficient for termination. The advantage is that standard (automatic) techniques can often synthesize such a well-founded ordering even if a direct termination proof with the same techniques fails. In this way, simplification orderings can now be used to prove termination of non-simply terminating TRSs.

This report contains a collection of several such systems from different areas of computer science (including many challenging problems from the literature). Moreover, applications of dependency pairs for realistic industrial problems in the area of distributed telecommunication processes are discussed in [GA01]. For an implementation of the dependency pair approach see [Art00] or [CiM99]. Dependency pairs have also been successfully applied in automatic termination proofs of logic programs, see [Ohl01,OCM00].

In Section 2 we briefly recapitulate the basic results of the dependency pair approach. Section 3 contains a collection of examples to demonstrate the use of dependency pairs for termination proofs of TRSs and Section 4 contains a corresponding collection for innermost termination proofs.

2 The dependency pair method

In the following we describe the notions relevant to the dependency pair method (where we assume the reader to be familiar with the basic notions of term rewriting [DJ90,Klo92,BN98]). In Section 2.1 we illustrate how dependency pairs are used for automatic termination proofs and in Section 2.2 we explain their use for innermost termination proofs. For motivations and further refinements see [AG00,GA01,GAO01]. We adopt the notation of [GM00] and [KNT99]. The *root* of a term $f(\ldots)$ is the leading function symbol f. For a TRS $\mathcal{R}$ over a signature $\mathcal{F}$, $\mathcal{D} = \{\mathrm{root}(l) | l \to r \in \mathcal{R}\}$ is the set of the *defined symbols* and $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$ is the set of *constructors* of $\mathcal{R}$. Let $\mathcal{F}^\sharp$ denote the union of the signature $\mathcal{F}$ and $\{f^\sharp \mid f$ is a defined symbol of $\mathcal{R}\}$, where $f^\sharp$ has the same arity as f. The functions $f^\sharp$ are called *tuple symbols*, where we often write F for $f^\sharp$, etc. Given a term $t = f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ with f defined, we write $t^\sharp$ for the term $t = f^\sharp(t_1, \ldots, t_n)$.

Definition 1 (Dependency pair). *If $l \to r \in \mathcal{R}$ and t is a subterm of r with defined root symbol, then the rewrite rule $l^\sharp \to t^\sharp$ is called a* dependency pair *of $\mathcal{R}$. The set of all dependency pairs of $\mathcal{R}$ is denoted by* $\mathrm{DP}(\mathcal{R})$.

2.1 Termination

In this section we explain how dependency pairs can be used to prove termination of TRSs.

Definition 2 (Chain). *A sequence of dependency pairs $s_1 \to t_1$, $s_2 \to t_2, \ldots$ is an $\mathcal{R}$-*chain *if there exists a substitution σ such that $t_j\sigma \to_{\mathcal{R}}^* s_{j+1}\sigma$ holds for every two consecutive pairs $s_j \to t_j$ and $s_{j+1} \to t_{j+1}$ in the sequence.*

We always assume that different (occurrences of) dependency pairs have disjoint sets of variables and we always consider substitutions whose domains may be infinite. In case $\mathcal{R}$ is clear from the context we often write *chain* instead of $\mathcal{R}$-chain. As proved in [AG97a,AG00], the absence of infinite chains is a sufficient and necessary criterion for termination.

Theorem 1 (Termination criterion). *A TRS $\mathcal{R}$ is terminating if and only if there exists no infinite $\mathcal{R}$-chain.*

Some dependency pairs can never occur twice in any chain and hence they need not be considered when proving that no infinite chain exists. For identifying these insignificant dependency pairs, the notion of *dependency graph* has been introduced [AG97a,AG00].

Definition 3 (Dependency graph). *The* dependency graph *of a TRS $\mathcal{R}$ is the directed graph whose nodes are the dependency pairs and there is an arc from $s \to t$ to $v \to w$ iff $s \to t$, $v \to w$ is a chain.*

A non-empty set $\mathcal{P}$ of dependency pairs is called a *cycle* if for any two pairs $s \to t$ and $v \to w$ in $\mathcal{P}$ there is a non-empty path from $s \to t$ to $v \to w$ which only traverses pairs from $\mathcal{P}$. Since we only regard finite TRSs, any infinite chain of dependency pairs corresponds to a cycle in the dependency graph. Hence, the dependency pairs that are not on a cycle in the dependency graph are insignificant for the termination proof. One can prove termination of a TRS in a modular way, by proving absence of infinite chains separately for every cycle [AG98,GAO01].

Theorem 2 (Modular termination criterion). *A TRS $\mathcal{R}$ is terminating if and only if for each cycle $\mathcal{P}$ in the dependency graph there exists no infinite $\mathcal{R}$-chain of dependency pairs from $\mathcal{P}$.*

This theorem can be refined by narrowing certain dependency pairs [AG00].

Definition 4 (Narrowing). *Let $\mathcal{R}$ be a TRS. A term t* narrows *to a term t' via the substitution μ if there exists a non-variable position p in t, μ is the most general unifier of $t|_p$ and l for some rewrite rule $l \to r$ of $\mathcal{R}$, and $t' = t\mu[r\mu]_p$. (Here, the variables of $l \to r$ must have been renamed to fresh variables.)*

Definition 5 (Narrowing pairs). *Let $\mathcal{R}$ be a TRS. If a term t narrows to a term t' via the substitution μ, then we say that the* pair *of terms $s \to t$ narrows to the pair $s\mu \to t'$.*

Theorem 3 (Narrowing refinement for termination). *Let $\mathcal{R}$ be a TRS and let $\mathcal{P}$ be a set of pairs of terms. Let $s \to t$ in $\mathcal{P}$ such that t is linear and for all $v \to w$ in $\mathcal{P}$ the terms t and v are not unifiable (after renaming the variables). Let*

$$\mathcal{P}' = \mathcal{P} \setminus \{s \to t\} \ \cup \ \{s' \to t' \mid s' \to t' \textit{ is a narrowing of } s \to t\}.$$

There exists an infinite $\mathcal{R}$-chain of pairs from $\mathcal{P}$ if and only if there exists an infinite $\mathcal{R}$-chain of pairs from $\mathcal{P}'$.

Strictly spoken, if in a set $\mathcal{P}$ a dependency pair is replaced by its narrowings, the resulting set is not a set of *dependency* pairs, but rather a set of pairs. The above theorem, however, states that we may use these sets of pairs instead of the original sets of dependency pairs in the other theorems stated here.

In order to check that no infinite chain of dependency pairs exists, sets of inequalities are generated. These inequalities should be satisfied by some pair $(\succsim, \succ)$ consisting of a quasi-rewrite ordering $\succsim$ (i.e., $\succsim$ must be a reflexive and transitive relation that is (weakly) monotonic and closed under substitutions) and an ordering $\succ$ with the properties

- $\succ$ is closed under substitutions and well founded
- $\succsim \circ \succ \subseteq \succ$ or $\succ \circ \succsim \subseteq \succ$.

(Note that $\succ$ need not be monotonic.) Such a pair is called a *reduction pair* [KNT99]. A termination proof for a certain TRS is transformed into the problem of finding several reduction pairs [AG98,GAO01].

Theorem 4 (Modular termination proofs I). *A TRS $\mathcal{R}$ is terminating if and only if for each cycle $\mathcal{P}$ in the dependency graph there is a reduction pair $(\succsim_{\mathcal{P}}, \succ_{\mathcal{P}})$ such that*

(a) $l \succsim_{\mathcal{P}} r$ for all rules $l \to r$ in $\mathcal{R}$,
(b) $s \succsim_{\mathcal{P}} t$ for all dependency pairs $s \to t$ from $\mathcal{P}$, and
(c) $s \succ_{\mathcal{P}} t$ for at least one dependency pair $s \to t$ from $\mathcal{P}$.

Of course, our aim is to use standard techniques to generate suitable reduction pairs satisfying the constraints of Theorem 4. However, most existing methods generate orderings which are *strongly* monotonic, whereas for the dependency pair approach we only need a *weakly* monotonic quasi-ordering. For that reason, before synthesizing a suitable ordering, some of the arguments of the function symbols can be eliminated. To perform this elimination of arguments resp. of function symbols the concept of argument filtering was introduced in [AG97a,AG00] (here we use the notation of [KNT99]).

Definition 6 (Argument filtering). *An* argument filtering *for a signature $\mathcal{F}$ is a mapping π that associates with every n-ary function symbol an argument position $i \in \{1, \ldots, n\}$ or a (possibly empty) list $[i_1, \ldots, i_m]$ of argument positions with $1 \leq i_1 < \ldots < i_m \leq n$. The signature $\mathcal{F}_\pi$ consists of all function symbols f such that $\pi(f) = [i_1, \ldots, i_m]$, where in $\mathcal{F}_\pi$ the arity of f is m. Every argument filtering π induces a mapping from $\mathcal{T}(\mathcal{F}, \mathcal{V})$ to $\mathcal{T}(\mathcal{F}_\pi, \mathcal{V})$, also denoted by π, which is defined as:*

$$\pi(t) \;=\; \begin{cases} t & \text{if } t \text{ is a variable,} \\ \pi(t_i) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } \pi(f) = i, \\ f(\pi(t_{i_1}), \ldots, \pi(t_{i_m})) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } \pi(f) = [i_1, \ldots, i_m]. \end{cases}$$

Theorem 5 (Modular termination proofs II). *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is terminating if and only if for each cycle $\mathcal{P}$ in the dependency graph there is an argument filtering $\pi_{\mathcal{P}}$ for $\mathcal{F}^\sharp$ and a reduction pair $(\succsim_{\mathcal{P}}, \succ_{\mathcal{P}})$ such that*

(a) $\pi_{\mathcal{P}}(l) \succsim_{\mathcal{P}} \pi_{\mathcal{P}}(r)$ for all rules $l \to r$ in $\mathcal{R}$,
(b) $\pi_{\mathcal{P}}(s) \succsim_{\mathcal{P}} \pi_{\mathcal{P}}(t)$ for all dependency pairs $s \to t$ from $\mathcal{P}$, and
(c) $\pi_{\mathcal{P}}(s) \succ_{\mathcal{P}} \pi_{\mathcal{P}}(t)$ for at least one dependency pair $s \to t$ from $\mathcal{P}$.

For the automation of the technique, we need to compute the dependency graph, find argument filterings, and synthesize a reduction pair for each set of inequalities. Since it is in general undecidable whether two dependency pairs form a chain, we need to estimate the dependency graph in such a way that all cycles in the real graph are also cycles in the estimated graph. Our estimation depends on two transformations that are applied to the right-hand side of a dependency pair [AG97a,AG00].

Let $\textsc{cap}(t)$ result from replacing all subterms of t that have a defined root symbol by different fresh variables and let $\textsc{ren}(t)$ result from replacing all variables in t by different fresh variables. Then, to determine whether $v \to w$ can follow $s \to t$ in a chain, we check whether $\textsc{ren}(\textsc{cap}(t))$ unifies with v.

Definition 7 (Estimated dependency graph). *The* estimated dependency graph *of a TRS $\mathcal{R}$ is the directed graph whose nodes are the dependency pairs and there is an arc from $s \to t$ to $v \to w$ if and only if* $\text{REN}(\text{CAP}(t))$ *and v are unifiable.*

With this definition Theorems 2, 4, and 5 also hold if we replace *dependency graph* by *estimated dependency graph* [GAO01].

The estimated dependency graph is computable, hence all the cycles in the graph are computable. The argument filtering can be found automatically by an exhaustive search. For every possible argument filtering one can try *quasi-simplification orderings* (QSOs) like RPO, LPO, KBO, polynomial interpretations, etc., to find a reduction pair that satisfies the inequalities [Art00]. Termination proved in this way is called *DP quasi-simple termination.* Because of the techniques we use to find a reduction pair, we restrict ourselves in the following to argument filterings such that for every pair/rule $s \to t$ we have $\mathcal{V}ar(\pi(t)) \subseteq \mathcal{V}ar(\pi(s))$ and $\pi(s) \notin \mathcal{V}$. Only for those argument filterings the techniques are potentially successful in practice.

Definition 8 (DP quasi-simple termination). *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is called* DP quasi-simply terminating *if and only if for each cycle $\mathcal{P}$ in the* **estimated** *dependency graph there exists an argument filtering $\pi_\mathcal{P}$ for $\mathcal{F}^\sharp$ and a reduction pair $(\succsim_\mathcal{P}, \succ_\mathcal{P})$ with a QSO $\succsim_\mathcal{P}$ such that*

(a) $\pi_\mathcal{P}(l) \succsim_\mathcal{P} \pi_\mathcal{P}(r)$ for all rules $l \to r$ in $\mathcal{R}$,
(b) $\pi_\mathcal{P}(s) \succsim_\mathcal{P} \pi_\mathcal{P}(t)$ for all dependency pairs $s \to t$ from $\mathcal{P}$, and
(c) $\pi_\mathcal{P}(s) \succ_\mathcal{P} \pi_\mathcal{P}(t)$ for at least one dependency pair $s \to t$ from $\mathcal{P}$.

If a quasi-simplification ordering exists such that either $s \succ t$ or s is syntactically equal to t for all inequalities $s \succsim t$, one obtains the notion of *DP simple termination.*

Definition 9 (DP simple termination). *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is called* DP simply terminating *if and only if for each cycle $\mathcal{P}$ in the* **estimated** *dependency graph there is an argument filtering π for $\mathcal{F}^\sharp$ and a simplification ordering $\succ_\mathcal{P}$ such that*

(a) $\pi_\mathcal{P}(l) \succeq_\mathcal{P} \pi_\mathcal{P}(r)$ for all rules $l \to r$ in $\mathcal{R}$,
(b) $\pi_\mathcal{P}(s) \succeq_\mathcal{P} \pi_\mathcal{P}(t)$ for all dependency pairs $s \to t$ from $\mathcal{P}$, and
(c) $\pi_\mathcal{P}(s) \succ_\mathcal{P} \pi_\mathcal{P}(t)$ for at least one dependency pair $s \to t$ from $\mathcal{P}$.

The information that systems are DP quasi-simple terminating can be used when combining these systems and proving termination of the resulting TRS [GO00,GAO01].

Theorem 6 (Modularity of DP quasi-simple termination). *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two TRSs over disjoint signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. Then their union $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ is DP quasi-simply terminating if and only if both $\mathcal{R}_1$ and $\mathcal{R}_2$ are DP quasi-simply terminating.*

For DP simple termination the modularity result for disjoint unions does not hold. The problem is that one of the two systems might have no cycle at all in the graph and is therefore, trivially, DP simply terminating. Combined with a system with a cycle, however, the inequalities corresponding to the rules should be satisfied, which is not always possible by a QSO in which the equivalence part is syntactic equivalence. Thus, Constraint (a) of Definition 9 must even be satisfied if the TRS only has the empty cycle $\mathcal{P}$. Furthermore, by restricting the argument filterings used in a suitable way, one can even extend the modularity result to constructor-sharing and composable combinations of TRSs [GAO01]. For that purpose, we introduce the notion of *$\mathcal{G}$-restricted DP simple termination.*

Definition 10 ($\mathcal{G}$-restricted DP simple termination). *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is called* $\mathcal{G}$-restricted DP simply terminating *for a signature $\mathcal{G}$ if and only if for each cycle $\mathcal{P}$ in the* **estimated** *dependency graph (including the empty one) there is an argument filtering $\pi_{\mathcal{P}}$ for $\mathcal{F}^\sharp$ and a simplification ordering $\succ_{\mathcal{P}}$ such that*

(a) $\pi_{\mathcal{P}}(l) \succeq_{\mathcal{P}} \pi_{\mathcal{P}}(r)$ for all rules $l \to r$ in $\mathcal{R}$,
(b) $\pi_{\mathcal{P}}(s) \succeq_{\mathcal{P}} \pi_{\mathcal{P}}(t)$ for all dependency pairs $s \to t$ from $\mathcal{P}$,
(c) $\pi_{\mathcal{P}}(s) \succ_{\mathcal{P}} \pi_{\mathcal{P}}(t)$ for at least one dependency pair $s \to t$ from $\mathcal{P}$ if $\mathcal{P} \neq \emptyset$,
(d) $\pi_{\mathcal{P}}(f) = [1, \ldots, n]$ for every $f \in \mathcal{F} \cap \mathcal{G}$, where n is the arity of f, and
(e) for every rule $l \to r \in \mathcal{R}$: if $\mathrm{root}(l) \notin \mathcal{G}$, then $\mathrm{root}(\pi_{\mathcal{P}}(l)) \notin \mathcal{G}$.

From the definition it is clear that $\mathcal{G}$-restricted DP simple termination implies DP simple termination. With this restricted notion of DP simple termination we obtain modularity for composable systems (and therefore also for systems with shared constructors and disjoint unions).

Theorem 7 (Modularity of $\mathcal{G}$-restricted DP simple termination). *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be composable TRSs over the signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. If $\mathcal{F}_1 \cap \mathcal{F}_2 \subseteq \mathcal{G}$, then their combined system $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ is $\mathcal{G}$-restricted DP simply terminating if and only if both $\mathcal{R}_1$ and $\mathcal{R}_2$ are $\mathcal{G}$-restricted DP simply terminating.*

2.2 Innermost termination

In [AG97b,AG00], we showed that the dependency pair approach can be modified in order to verify *innermost* termination.

Definition 11 (Innermost chain). *A sequence of dependency pairs $s_1 \to t_1$, $s_2 \to t_2$, ... is an* innermost $\mathcal{R}$-chain *if there exists a substitution σ such that all $s_j\sigma$ are in normal form and $t_j\sigma \xrightarrow{\mathrm{i}}{}^{*}_{\mathcal{R}} s_{j+1}\sigma$ holds for every two consecutive pairs $s_j \to t_j$ and $s_{j+1} \to t_{j+1}$ in the sequence. Here, '$\xrightarrow{\mathrm{i}}$' denotes innermost reductions.*

The absence of infinite innermost chains is a sufficient and necessary criterion for innermost termination.

Theorem 8 (Innermost termination criterion). *A TRS $\mathcal{R}$ is innermost terminating if and only if there exists no infinite innermost $\mathcal{R}$-chain.*

Analogous to Section 2.1, the notion of a graph is defined for innermost chains.

Definition 12 (Innermost dependency graph). *The* innermost dependency graph *of a TRS $\mathcal{R}$ is the directed graph whose nodes are the dependency pairs and there is an arc from $s \to t$ to $v \to w$ iff $s \to t$, $v \to w$ is an innermost chain.*

Similar to termination, one can also prove innermost termination of TRSs in a modular way [AG98,GAO01].

Theorem 9 (Modular innermost termination criterion). *A TRS $\mathcal{R}$ is innermost terminating if and only if for each cycle $\mathcal{P}$ in the innermost dependency graph there is no infinite innermost $\mathcal{R}$-chain of dependency pairs from $\mathcal{P}$.*

This theorem can also be refined by narrowing certain dependency pairs [AG00,GA01].

Theorem 10 (Narrowing refinement for innermost termination). *Let $\mathcal{R}$ be a TRS and let $\mathcal{P}$ be a set of pairs of terms. Let $s \to t$ in $\mathcal{P}$ such that $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(s)$ and such that for all $v \to w$ in $\mathcal{P}$ the terms t and v are not unifiable (after renaming the variables). Let*

$$\mathcal{P}' = \mathcal{P} \setminus \{s \to t\} \;\cup\; \{s' \to t' \mid s' \to t' \textit{ is a narrowing of } s \to t\}.$$

If there exists no infinite innermost chain of pairs from $\mathcal{P}'$, then there exists no infinite innermost chain of pairs from $\mathcal{P}$ either.

Moreover, if $\mathcal{R}$ is innermost terminating and non-overlapping, then the converse holds as well (i.e., if there exists no infinite innermost chain of pairs from $\mathcal{P}$, then there exists no infinite innermost chain of pairs from $\mathcal{P}'$ either).

Further refinements of this theorem as well as additional techniques for modifying dependency pairs by rewriting and by instantiation can be found in [GA01].

To prove innermost termination automatically, we again generate a set of inequalities for every cycle $\mathcal{P}$ and search for a reduction pair $(\succsim_{\mathcal{P}}, \succ_{\mathcal{P}})$ satisfying them. However, to ensure $t\sigma \succsim_{\mathcal{P}} v\sigma$ whenever $t\sigma$ reduces to $v\sigma$, now it is sufficient to require $l \succsim_{\mathcal{P}} r$ only for those rules that are *usable* in a reduction of $t\sigma$ (for *normal* substitutions σ).

Definition 13 (Usable rules). *Let $\mathcal{R}$ be a term rewrite system. For any symbol f let $Rules_{\mathcal{R}}(f) = \{l \to r \in \mathcal{R} \mid \mathrm{root}(l) = f\}$. For any term we define the* usable rules*:*

- $\mathcal{U}_{\mathcal{R}}(x) = \emptyset$,
- $\mathcal{U}_{\mathcal{R}}(f(t_1, \ldots, t_n)) = Rules_{\mathcal{R}}(f) \;\cup\; \bigcup_{l \to r \in Rules_{\mathcal{R}}(f)} \mathcal{U}_{\mathcal{R}'}(r) \;\cup\; \bigcup_{j=1}^{n} \mathcal{U}_{\mathcal{R}'}(t_j)$,

where $\mathcal{R}' = \mathcal{R} \setminus Rules_{\mathcal{R}}(f)$. *Moreover, for any set* $\mathcal{P}$ *of dependency pairs we define* $\mathcal{U}_{\mathcal{R}}(\mathcal{P}) = \bigcup_{s \to t \in \mathcal{P}} \mathcal{U}_{\mathcal{R}}(t)$.

Now we can state the theorem for innermost termination proofs.

Theorem 11 (Modular innermost termination proofs). *A TRS* $\mathcal{R}$ *over a signature* $\mathcal{F}$ *is innermost terminating if for each cycle* $\mathcal{P}$ *in the innermost dependency graph there is an argument filtering* π *for* $\mathcal{F}^\sharp$ *and a reduction pair* $(\succsim_{\mathcal{P}}, \succ_{\mathcal{P}})$ *such that*

(a) $\pi(l) \succsim_{\mathcal{P}} \pi(r)$ *for all rules* $l \to r$ *in* $\mathcal{U}_{\mathcal{R}}(\mathcal{P})$*,*
(b) $\pi(s) \succsim_{\mathcal{P}} \pi(t)$ *for all dependency pairs* $s \to t$ *from* $\mathcal{P}$*, and*
(c) $\pi(s) \succ_{\mathcal{P}} \pi(t)$ *for at least one dependency pair* $s \to t$ *from* $\mathcal{P}$*.*

For the purpose of automation we again need an estimation of the innermost dependency graph, since in general it is undecidable whether two dependency pairs $s \to t$ and $v \to w$ form an innermost chain. To this end, we again replace subterms in t with defined root symbols by new variables and check whether this modification of t unifies with v, but in contrast to Section 2.1 we do not rename multiple occurrences of the same variable.

Moreover, to eliminate defined symbols we use a modified transformation CAP_s where $\text{CAP}_s(t)$ only replaces those subterms of t by different fresh variables which have a defined root symbol and which are no subterms of s. Then to refine the approximation of innermost dependency graphs instead of $\text{CAP}(t)$ we check whether $\text{CAP}_s(t)$ unifies with v. Finally, if μ is the most general unifier (mgu) of $\text{CAP}_s(t)$ and v, then there can only be an arc from $s \to t$ to $v \to w$ in the innermost dependency graph, if both $s\mu$ and $v\mu$ are in normal form.

Definition 14 (Estimated innermost dependency graph). *The* estimated innermost dependency graph *of a TRS* $\mathcal{R}$ *is the directed graph whose nodes are the dependency pairs and there is an arc from* $s \to t$ *to* $v \to w$ *if and only if* $\text{CAP}_s(t)$ *and* v *are unifiable by a most general unifier* μ *such that* $s\mu$ *and* $v\mu$ *are normal forms.*

With this definition Theorems 9 and 11 also hold if we replace *innermost dependency graph* by *estimated innermost dependency graph.*

In [AG98,GAO01], two corollaries of the above results were presented which are particularly useful in practice.

Corollary 15 (Innermost termination for hierarchical combinations) *Let* $\mathcal{R}$ *be the hierarchical combination of* $\mathcal{R}_1$ *and* $\mathcal{R}_2$.

(a) $\mathcal{R}$ *is innermost terminating if and only if* $\mathcal{R}_1$ *is innermost terminating and there exists no infinite innermost* $\mathcal{R}$*-chain of* $\mathcal{R}_2$*-dependency pairs.*
(b) $\mathcal{R}$ *is innermost terminating if* $\mathcal{R}_1$ *is innermost terminating and if there exists a reduction pair* $(\succsim, \succ)$ *such that for all dependency pairs* $s \to t$ *of* $\mathcal{R}_2$

- $l \succsim r$ *for all rules* $l \to r$ *in* $\mathcal{U}_{\mathcal{R}}(t)$ *and*
- $s \succ t$.

The following corollary of Theorem 9 shows that the consideration of cycles in the (estimated) innermost dependency graph can also be used to decompose a TRS into modular subsystems. In the following, let $\mathcal{O}(\mathcal{P})$ denote the *origin* of the dependency pairs in $\mathcal{P}$, i.e., $\mathcal{O}(\mathcal{P})$ is a set of those rules where the dependency pairs of $\mathcal{P}$ stem from. If a dependency pair of $\mathcal{P}$ may stem from *several* rules, then it is sufficient if $\mathcal{O}(\mathcal{P})$ just contains one of them.

Corollary 16 (Modularity for subsystems) *Let* $\mathcal{R}$ *be a TRS, let* $\mathcal{P}_1, \ldots, \mathcal{P}_n$ *be the cycles in its (estimated) innermost dependency graph, and let* $\mathcal{R}_j$ *be subsystems of* $\mathcal{R}$ *such that* $\mathcal{U}_{\mathcal{R}}(\mathcal{P}_j) \cup \mathcal{O}(\mathcal{P}_j) \subseteq \mathcal{R}_j$ *(for all* $j \in \{1,\ldots,n\}$*). If* $\mathcal{R}_1, \ldots, \mathcal{R}_n$ *are innermost terminating, then* $\mathcal{R}$ *is also innermost terminating.*

For further corollaries and results on the relation of our modularity results to previous modularity results the reader is referred to [GAO01].

3 Examples for termination

This section contains a collection of TRSs where termination can be proved by the technique described above. The majority of them occurred as challenge problems in the literature, whereas the other examples are added to point out specific failures of existing techniques. Several of these examples are not simply terminating. Thus, all methods based on simplification orderings fail in proving termination of these systems. For those examples which are overlay systems with joinable critical pairs, termination can also be verified by proving innermost termination using the technique of Section 2.2.

In the examples, we refer to the sets of inequalities that result from a cycle in the estimated dependency graph and the rules of the system as "the inequalities" (cf. Theorem 4 and 5). However, in most of the examples, only the inequalities resulting from dependency pairs on cycles are mentioned. But of course, the inequalities $l \succsim r$ are also synthesized for each rewrite rule $l \to r$ in the term rewrite system. The argument filterings that we use are only described for those function symbols f with arity n for which $\pi(f) \neq [1, \ldots, n]$, i.e., only those function symbols where some arguments are really filtered.

In this collection of examples, three different techniques are used to find a reduction pair, viz. the recursive path ordering, the lexicographic path ordering, and polynomial interpretations. For Examples 3.39 – 3.46 we need the refinement of narrowing dependency pairs and Examples 3.47 – 3.57 illustrate the use of our modularity results.

3.1 Division, version 1

The TRS

$$\begin{aligned}
\mathsf{minus}(x, \mathsf{0}) &\to x \\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(x, y) \\
\mathsf{quot}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{0} \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y)))
\end{aligned}$$

is not simply terminating. In this example, we have two cycles, viz.

$$\begin{array}{l}
\{\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{MINUS}(x, y)\} \\
\{\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{QUOT}(\mathsf{minus}(x, y), \mathsf{s}(y))\}.
\end{array}$$

Apart from the four inequalities corresponding to the rewrite rules, one strict inequality is obtained per cycle. Both sets of inequalities are solved by the argument filtering $\pi(\mathsf{minus}) = [1]$ and RPO. Hence DP simple termination is proved.

3.2 Division, version 2

This TRS for division uses different minus-rules. Again, it is not simply terminating.

$$\begin{aligned}
\mathsf{pred}(\mathsf{s}(x)) &\to x \\
\mathsf{minus}(x, \mathsf{0}) &\to x \\
\mathsf{minus}(x, \mathsf{s}(y)) &\to \mathsf{pred}(\mathsf{minus}(x, y)) \\
\mathsf{quot}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{0} \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y)))
\end{aligned}$$

The cycles in the estimated dependency graph are given by:

$$\begin{array}{l}
\{\mathsf{MINUS}(x, \mathsf{s}(y)) \to \mathsf{MINUS}(x, y)\} \\
\{\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{QUOT}(\mathsf{minus}(x, y), \mathsf{s}(y))\}
\end{array}$$

Finding a suitable ordering is as easy as it was for the previous example, by choosing the argument filtering $\pi(\mathsf{minus}) = 1$ and $\pi(\mathsf{pred}) = 1$. Then DP simple termination can be shown by RPO.

3.3 Division, version 3

This TRS for division uses again different minus-rules. Similar to the preceding examples it is not simply terminating. In the examples of this collection, we often use functions like $\mathsf{if}_{\mathsf{minus}}$ to encode conditions. This ensures that conditions are evaluated first (to true or to false) and that the corresponding result is evaluated afterwards. Hence, the first argument of $\mathsf{if}_{\mathsf{minus}}$ is the condition that has to be tested and the other arguments are the original arguments of minus. Further evaluation is only possible after the condition has been reduced to true or to false.

$$\begin{aligned}
\mathsf{le}(\mathsf{0}, y) &\to \mathsf{true} \\
\mathsf{le}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{false} \\
\mathsf{le}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{le}(x, y) \\
\mathsf{minus}(\mathsf{0}, y) &\to \mathsf{0} \\
\mathsf{minus}(\mathsf{s}(x), y) &\to \mathsf{if}_{\mathsf{minus}}(\mathsf{le}(\mathsf{s}(x), y), \mathsf{s}(x), y) \\
\mathsf{if}_{\mathsf{minus}}(\mathsf{true}, \mathsf{s}(x), y) &\to \mathsf{0} \\
\mathsf{if}_{\mathsf{minus}}(\mathsf{false}, \mathsf{s}(x), y) &\to \mathsf{s}(\mathsf{minus}(x, y)) \\
\mathsf{quot}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{0} \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y)))
\end{aligned}$$

The cycles are

$$\begin{array}{l}
\{\mathsf{LE}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{LE}(x, y)\} \\
\{\mathsf{MINUS}(\mathsf{s}(x), y) \to \mathsf{IF}_{\mathsf{minus}}(\mathsf{le}(\mathsf{s}(x), y), \mathsf{s}(x), y), \\
\ \mathsf{IF}_{\mathsf{minus}}(\mathsf{false}, \mathsf{s}(x), y) \to \mathsf{MINUS}(x, y)\} \\
\{\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{QUOT}(\mathsf{minus}(x, y), \mathsf{s}(y))\}.
\end{array}$$

Note that only one of the dependency pairs on a cycle in the dependency graph should result in a strict inequality, therefore the inequality

$$\pi(\mathsf{MINUS}(\mathsf{s}(x), y)) \gtrsim \pi(\mathsf{IF}_{\mathsf{minus}}(\mathsf{le}(\mathsf{s}(x), y), \mathsf{s}(x), y))$$

need not be strict. By normalizing the inequalities with respect to the argument filtering $\pi(\mathsf{minus}) = \pi(\mathsf{MINUS}) = 1$ and $\pi(\mathsf{if}_{\mathsf{minus}}) = \pi(\mathsf{IF}_{\mathsf{minus}}) = 2$ the inequalities for DP simple termination are satisfied by the recursive path ordering.

3.4 Plus and minus

The following example demonstrates the use of the dependency graph. For that purpose we extend the TRS of Ex. 3.1 by three additional rules and write infix operators for the defined symbols minus and plus to ease readability.

$$\begin{aligned}
x - \mathsf{0} &\to x \\
\mathsf{s}(x) - \mathsf{s}(y) &\to x - y \\
\mathsf{quot}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{0} \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{quot}(x - y, \mathsf{s}(y))) \\
\mathsf{0} + y &\to y \\
\mathsf{s}(x) + y &\to \mathsf{s}(x + y) \\
(x - y) - z &\to x - (y + z)
\end{aligned}$$

In this example, termination cannot be proved with our method using a simplification ordering, unless we use the dependency graph to determine that the dependency pair $\mathsf{MINUS}(\ldots) \to \mathsf{PLUS}(\ldots)$ does not occur on any cycle. There are five cycles in the estimated dependency graph.

$$
\begin{array}{l}
\{\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{MINUS}(x, y)\} \\
\{\mathsf{MINUS}(x - y, z) \to \mathsf{MINUS}(x, y + z)\} \\
\{\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{MINUS}(x, y), \\
\ \mathsf{MINUS}(x - y, z) \to \mathsf{MINUS}(x, y + z)\} \\
\{\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{QUOT}(x - y, \mathsf{s}(y))\} \\
\{\mathsf{PLUS}(\mathsf{s}(x), y) \to \mathsf{PLUS}(x, y)\}
\end{array}
$$

After applying the argument filtering $\pi(-) = [1]$, $\pi(\mathsf{MINUS}) = [1]$, the inequalities are satisfied by the recursive path ordering and DP simple termination is proved. Note that in such examples, we need not consider all subcycles of a cycle if the inequalities in the larger cycle are all chosen to be strict.

3.5 Remainder, version 1 – 3

Similar to the TRSs for division, three versions of the following TRS are obtained, which again are not simply terminating. Only one of them is presented.

$$
\begin{aligned}
\mathsf{le}(0, y) &\to \mathsf{true} \\
\mathsf{le}(\mathsf{s}(x), 0) &\to \mathsf{false} \\
\mathsf{le}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{le}(x, y) \\
\mathsf{minus}(x, 0) &\to x \\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(x, y) \\
\mathsf{mod}(0, y) &\to 0 \\
\mathsf{mod}(\mathsf{s}(x), 0) &\to 0 \\
\mathsf{mod}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{if}_{\mathsf{mod}}(\mathsf{le}(y, x), \mathsf{s}(x), \mathsf{s}(y)) \\
\mathsf{if}_{\mathsf{mod}}(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{mod}(\mathsf{minus}(x, y), \mathsf{s}(y)) \\
\mathsf{if}_{\mathsf{mod}}(\mathsf{false}, \mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(x)
\end{aligned}
$$

The cycles are

$$
\begin{array}{l}
\{\mathsf{LE}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{LE}(x, y)\} \\
\{\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{MINUS}(x, y)\} \\
\{\mathsf{MOD}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{IF}_{\mathsf{mod}}(\mathsf{le}(y, x), \mathsf{s}(x), \mathsf{s}(y)), \\
\ \mathsf{IF}_{\mathsf{mod}}(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{MOD}(\mathsf{minus}(x, y), \mathsf{s}(y))\}.
\end{array}
$$

By applying the argument filtering, $\pi(\mathsf{minus}) = \pi(\mathsf{mod}) = \pi(\mathsf{MOD}) = 1$ and $\pi(\mathsf{if}_{\mathsf{mod}}) = \pi(\mathsf{IF}_{\mathsf{mod}}) = 2$, the inequalities obtained for DP simple termination are satisfied by the recursive path ordering.

3.6 Greatest common divisor, version 1 – 3

There are also three versions of the following TRS for the computation of the greatest common divisor, which are not simply terminating. Again, only one of them is presented.

$$
\begin{aligned}
\mathsf{le}(\mathsf{0}, y) &\to \mathsf{true}\\
\mathsf{le}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{false}\\
\mathsf{le}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{le}(x, y)\\
\mathsf{pred}(\mathsf{s}(x)) &\to x\\
\mathsf{minus}(x, \mathsf{0}) &\to x\\
\mathsf{minus}(x, \mathsf{s}(y)) &\to \mathsf{pred}(\mathsf{minus}(x, y))\\
\mathsf{gcd}(\mathsf{0}, y) &\to y\\
\mathsf{gcd}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{s}(x)\\
\mathsf{gcd}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{if}_{\mathsf{gcd}}(\mathsf{le}(y, x), \mathsf{s}(x), \mathsf{s}(y))\\
\mathsf{if}_{\mathsf{gcd}}(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{gcd}(\mathsf{minus}(x, y), \mathsf{s}(y))\\
\mathsf{if}_{\mathsf{gcd}}(\mathsf{false}, \mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{gcd}(\mathsf{minus}(y, x), \mathsf{s}(x))
\end{aligned}
$$

(Of course the ordering of the arguments in the right-hand side of the last rule could have been switched. But this version here is even more difficult: Termination of the corresponding algorithm cannot be proved by the method of Walther [Wal94], because this method cannot deal with permutations of arguments.)
The cycles in the estimated dependency graph of this TRS are

$$
\begin{aligned}
&(1)\ \{\mathsf{LE}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{LE}(x, y)\}\\
&(2)\ \{\mathsf{MINUS}(x, \mathsf{s}(y)) \to \mathsf{MINUS}(x, y)\}\\
&(3)\ \{\mathsf{GCD}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{IF}_{\mathsf{gcd}}(\mathsf{le}(y, x), \mathsf{s}(x), \mathsf{s}(y)),\\
&\qquad \mathsf{IF}_{\mathsf{gcd}}(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{GCD}(\mathsf{minus}(x, y), \mathsf{s}(y)),\\
&\qquad \mathsf{IF}_{\mathsf{gcd}}(\mathsf{false}, \mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{GCD}(\mathsf{minus}(y, x), \mathsf{s}(x))\},
\end{aligned}
$$

where (3) has two subcycles. Note that by the argument filtering $\pi(\mathsf{pred}) = \pi(\mathsf{minus}) = 1$, $\pi(\mathsf{if}_{\mathsf{gcd}}) = \pi(\mathsf{IF}_{\mathsf{gcd}}) = [2, 3]$ the inequalities are solved by RPO, also those that are related to the subcycles. In this construction, however, GCD and $\mathsf{IF}_{\mathsf{gcd}}$ have to be chosen equal in the precedence and therefore we only show DP quasi-simple termination.

This example was taken from Boyer and Moore [BM79] and Walther [Wal91]. A variant of this example could be proved terminating using Steinbach's method for the automated generation of transformation orderings [Ste95a], but there the rules for le and minus were missing.

3.7 Logarithm, version 1

The following TRS computes the dual logarithm.

$$
\begin{aligned}
\mathsf{half}(\mathsf{0}) &\to \mathsf{0}\\
\mathsf{half}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{half}(x))\\
\mathsf{log}(\mathsf{s}(\mathsf{0})) &\to \mathsf{0}\\
\mathsf{log}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{log}(\mathsf{s}(\mathsf{half}(x))))
\end{aligned}
$$

The cycles are

$$
\begin{array}{l}
\{\mathsf{HALF}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{HALF}(x)\}\\
\{\mathsf{LOG}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{LOG}(\mathsf{s}(\mathsf{half}(x)))\}.
\end{array}
$$

Without filtering arguments the inequalities are satisfied by the recursive path ordering. (Termination of the original system can also be proved using the recursive path ordering with precedence $\mathsf{log} > \mathsf{s} > \mathsf{half}$.)

3.8 Logarithm, version 2 – 4

The following TRS again computes the dual logarithm, but instead of half now the function quot is used. Depending on which version of quot one chooses, three different versions of the TRS are obtained (all of which are not simply terminating, since the quot TRS already was not simply terminating).

$$
\begin{aligned}
\mathsf{minus}(x, \mathsf{0}) &\to x\\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(x, y)\\
\mathsf{quot}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{0}\\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y)))\\
\mathsf{log}(\mathsf{s}(\mathsf{0})) &\to \mathsf{0}\\
\mathsf{log}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{log}(\mathsf{s}(\mathsf{quot}(x, \mathsf{s}(\mathsf{s}(\mathsf{0}))))))
\end{aligned}
$$

There are three cycles in the estimated dependency graph:

$$
\begin{array}{l}
\{\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{MINUS}(x, y)\}\\
\{\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{QUOT}(\mathsf{minus}(x, y), \mathsf{s}(y))\}\\
\{\mathsf{LOG}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{LOG}(\mathsf{s}(\mathsf{quot}(x, \mathsf{s}(\mathsf{s}(\mathsf{0})))))\}.
\end{array}
$$

After applying the argument filtering $\pi(\mathsf{quot}) = \pi(\mathsf{minus}) = 1$, the inequalities for DP simple termination are satisfied by the recursive path ordering.

3.9 Eliminating duplicates

The following TRS eliminates duplicates from a list. To represent lists the constructors nil and add are used, where nil represents the empty list and $\mathsf{add}(n, x)$ represents the insertion of n into the list x.

$$
\begin{aligned}
\mathsf{eq}(\mathsf{0}, \mathsf{0}) &\to \mathsf{true}\\
\mathsf{eq}(\mathsf{0}, \mathsf{s}(x)) &\to \mathsf{false}
\end{aligned}
$$

$$\begin{aligned}
\mathsf{eq}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{eq}(x, y)\\
\mathsf{rm}(n, \mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{rm}(n, \mathsf{add}(m, x)) &\to \mathsf{if}_{\mathsf{rm}}(\mathsf{eq}(n, m), n, \mathsf{add}(m, x))\\
\mathsf{if}_{\mathsf{rm}}(\mathsf{true}, n, \mathsf{add}(m, x)) &\to \mathsf{rm}(n, x)\\
\mathsf{if}_{\mathsf{rm}}(\mathsf{false}, n, \mathsf{add}(m, x)) &\to \mathsf{add}(m, \mathsf{rm}(n, x))\\
\mathsf{purge}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{purge}(\mathsf{add}(n, x)) &\to \mathsf{add}(n, \mathsf{purge}(\mathsf{rm}(n, x)))
\end{aligned}$$

The cycles are

$$\begin{array}{l}
\{\mathsf{EQ}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{EQ}(x, y)\}\\
\{\mathsf{RM}(n, \mathsf{add}(m, x)) \to \mathsf{IF}_{\mathsf{rm}}(\mathsf{eq}(n, m), n, \mathsf{add}(m, x)),\\
\ \mathsf{IF}_{\mathsf{rm}}(\mathsf{true}, n, \mathsf{add}(m, x)) \to \mathsf{RM}(n, x)\}\\
\{\mathsf{RM}(n, \mathsf{add}(m, x)) \to \mathsf{IF}_{\mathsf{rm}}(\mathsf{eq}(n, m), n, \mathsf{add}(m, x)),\\
\ \mathsf{IF}_{\mathsf{rm}}(\mathsf{false}, n, \mathsf{add}(m, x)) \to \mathsf{RM}(n, x)\}\\
\{\mathsf{RM}(n, \mathsf{add}(m, x)) \to \mathsf{IF}_{\mathsf{rm}}(\mathsf{eq}(n, m), n, \mathsf{add}(m, x)),\\
\ \mathsf{IF}_{\mathsf{rm}}(\mathsf{true}, n, \mathsf{add}(m, x)) \to \mathsf{RM}(n, x),\\
\ \mathsf{IF}_{\mathsf{rm}}(\mathsf{false}, n, \mathsf{add}(m, x)) \to \mathsf{RM}(n, x)\}\\
\{\mathsf{PURGE}(\mathsf{add}(n, x)) \to \mathsf{PURGE}(\mathsf{rm}(n, x))\}.
\end{array}$$

By applying the argument filtering $\pi(\mathsf{rm}) = \pi(\mathsf{RM}) = 2$, $\pi(\mathsf{if}_{\mathsf{rm}}) = \pi(\mathsf{IF}_{\mathsf{rm}}) = 3$, the obtained inequalities are satisfied by the recursive path ordering and DP simple termination is proved.

This example comes from Walther [Wal91] and a similar example was mentioned by Steinbach [Ste95a], but in Steinbach's version the rules for eq and $\mathsf{if}_{\mathsf{rm}}$ were missing.

If in the right-hand side of the last rule, $\mathsf{add}(n, \mathsf{purge}(\mathsf{rm}(\mathbf{n}, x)))$, the $\mathbf{n}$ is replaced by a term containing $\mathsf{add}(n, x)$ then a non-simply terminating TRS is obtained, but termination can still be proved in the same way.

3.10 Minimum sort

This TRS can be used to sort a list x by repeatedly removing its minimum. For that purpose elements of x are shifted into the second argument of $\mathsf{minsort}$, until the minimum of the list is reached. Then the function rm is used to eliminate *all* occurrences of the minimum and finally $\mathsf{minsort}$ is called recursively on the remaining list. Hence, $\mathsf{minsort}$ does not only sort a list but it also eliminates duplicates. (The corresponding version of $\mathsf{minsort}$ where duplicates are not eliminated could also be proved terminating with our technique.)

$$\begin{aligned}
\mathsf{eq}(\mathsf{0}, \mathsf{0}) &\to \mathsf{true}\\
\mathsf{eq}(\mathsf{0}, \mathsf{s}(x)) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{false}
\end{aligned}$$

$$
\begin{aligned}
\mathsf{eq}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{eq}(x,y)\\
\mathsf{le}(\mathsf{0},y) &\to \mathsf{true}\\
\mathsf{le}(\mathsf{s}(x),\mathsf{0}) &\to \mathsf{false}\\
\mathsf{le}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{le}(x,y)\\
\mathsf{app}(\mathsf{nil},y) &\to y\\
\mathsf{app}(\mathsf{add}(n,x),y) &\to \mathsf{add}(n,\mathsf{app}(x,y))\\
\mathsf{min}(\mathsf{add}(n,\mathsf{nil})) &\to n\\
\mathsf{min}(\mathsf{add}(n,\mathsf{add}(m,x))) &\to \mathsf{if}_{\mathsf{min}}(\mathsf{le}(n,m),\mathsf{add}(n,\mathsf{add}(m,x)))\\
\mathsf{if}_{\mathsf{min}}(\mathsf{true},\mathsf{add}(n,\mathsf{add}(m,x))) &\to \mathsf{min}(\mathsf{add}(n,x))\\
\mathsf{if}_{\mathsf{min}}(\mathsf{false},\mathsf{add}(n,\mathsf{add}(m,x))) &\to \mathsf{min}(\mathsf{add}(m,x))\\
\mathsf{rm}(n,\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{rm}(n,\mathsf{add}(m,x)) &\to \mathsf{if}_{\mathsf{rm}}(\mathsf{eq}(n,m),n,\mathsf{add}(m,x))\\
\mathsf{if}_{\mathsf{rm}}(\mathsf{true},n,\mathsf{add}(m,x)) &\to \mathsf{rm}(n,x)\\
\mathsf{if}_{\mathsf{rm}}(\mathsf{false},n,\mathsf{add}(m,x)) &\to \mathsf{add}(m,\mathsf{rm}(n,x))\\
\mathsf{minsort}(\mathsf{nil},\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{minsort}(\mathsf{add}(n,x),y) &\to \mathsf{if}_{\mathsf{minsort}}(\mathsf{eq}(n,\mathsf{min}(\mathsf{add}(n,x))),\mathsf{add}(n,x),y)\\
\mathsf{if}_{\mathsf{minsort}}(\mathsf{true},\mathsf{add}(n,x),y) &\to \mathsf{add}(n,\mathsf{minsort}(\mathsf{app}(\mathsf{rm}(n,x),y),\mathsf{nil}))\\
\mathsf{if}_{\mathsf{minsort}}(\mathsf{false},\mathsf{add}(n,x),y) &\to \mathsf{minsort}(x,\mathsf{add}(n,y))
\end{aligned}
$$

The cycles in the estimated dependency graph and an argument filtering that does not filter any argument result in the following set of inequalities.

$$
\begin{aligned}
\mathsf{EQ}(\mathsf{s}(x),\mathsf{s}(y)) &\succ \mathsf{EQ}(x,y)\\
\mathsf{LE}(\mathsf{s}(x),\mathsf{s}(y)) &\succ \mathsf{LE}(x,y)\\
\mathsf{APP}(\mathsf{add}(n,x),y) &\succ \mathsf{APP}(x,y)\\
\mathsf{MIN}(\mathsf{add}(n,\mathsf{add}(m,x))) &\succsim \mathsf{IF}_{\mathsf{min}}(\mathsf{le}(n,m),\mathsf{add}(n,\mathsf{add}(m,x)))\\
\mathsf{IF}_{\mathsf{min}}(\mathsf{true},\mathsf{add}(n,\mathsf{add}(m,x))) &\succ \mathsf{MIN}(\mathsf{add}(n,x))\\
\mathsf{IF}_{\mathsf{min}}(\mathsf{false},\mathsf{add}(n,\mathsf{add}(m,x))) &\succ \mathsf{MIN}(\mathsf{add}(m,x))\\
\mathsf{RM}(n,\mathsf{add}(m,x)) &\succsim \mathsf{IF}_{\mathsf{rm}}(\mathsf{eq}(n,m),n,\mathsf{add}(m,x))\\
\mathsf{IF}_{\mathsf{rm}}(\mathsf{true},n,\mathsf{add}(m,x)) &\succ \mathsf{RM}(n,x)\\
\mathsf{IF}_{\mathsf{rm}}(\mathsf{false},n,\mathsf{add}(m,x)) &\succ \mathsf{RM}(n,x)\\
\mathsf{MINSORT}(\mathsf{add}(n,x),y) &\succ \mathsf{IF}_{\mathsf{minsort}}(\mathsf{eq}(n,\mathsf{min}(\mathsf{add}(n,x))),\mathsf{add}(n,x),y)\\
\mathsf{IF}_{\mathsf{minsort}}(\mathsf{true},\mathsf{add}(n,x),y) &\succsim \mathsf{MINSORT}(\mathsf{app}(\mathsf{rm}(n,x),y),\mathsf{nil})\\
\mathsf{IF}_{\mathsf{minsort}}(\mathsf{false},\mathsf{add}(n,x),y) &\succsim \mathsf{MINSORT}(x,\mathsf{add}(n,y)).
\end{aligned}
$$

These constraints together with the constraints on the rules are satisfied by a polynomial ordering where false, true, $\mathsf{0}$, nil, eq and le are mapped to 0, $\mathsf{s}(x)$ is mapped to $x+1$, $\mathsf{min}(x)$, $\mathsf{if}_{\mathsf{min}}(b,x)$, $\mathsf{EQ}(x,y)$, $\mathsf{LE}(x,y)$, $\mathsf{MIN}(x)$, and $\mathsf{IF}_{\mathsf{min}}(b,x)$ are mapped to x, $\mathsf{add}(n,x)$ is mapped to $n+x+1$, $\mathsf{app}(x,y)$ and $\mathsf{APP}(x,y)$ are mapped to $x + y$, $\mathsf{rm}(n, x)$, $\mathsf{if}_{\mathsf{rm}}(b, n, x)$, $\mathsf{RM}(n, x)$, and $\mathsf{IF}_{\mathsf{rm}}(b, n, x)$ are mapped to x, $\mathsf{minsort}(x, y)$ and $\mathsf{if}_{\mathsf{minsort}}(b, x, y)$ are mapped to $x + y$, $\mathsf{MINSORT}(x, y)$ is mapped to $(x+y)^2+2x+y+1$, and $\mathsf{IF}_{\mathsf{minsort}}(b, x, y)$ is mapped to $(x+y)^2+2x+y$.

This example is inspired by an algorithm from Boyer and Moore [BM79] and Walther [Wal94]. In the corresponding example from Steinbach [Ste95a] the rules for eq, le, $\mathsf{if}_{\mathsf{rm}}$, and $\mathsf{if}_{\mathsf{min}}$ were missing.

Note that we have only shown DP quasi-simple termination by using this polynomial interpretation in which syntactically unequal terms are identified by the equivalence relation. (The given polynomial ordering is not a QSO, since the polynomials for symbols like eq or le do not contain all variables corresponding to their arguments. However, by using a suitable argument filtering before (where $\pi(\mathsf{eq}) = \pi(\mathsf{le}) = [\,]$, etc.), one can easily replace the current polynomial ordering by a polynomial ordering which is indeed a QSO. Similar observations also hold for the other examples where polynomial interpretations are used.)

3.11 Quicksort

The following TRS is used to sort a list by the well-known quicksort algorithm. It uses the functions $\mathsf{low}(n, x)$ (resp. $\mathsf{high}(n, x)$) which return the sublist of x containing only the elements smaller than or equal to (resp. greater than) n.

$$
\begin{aligned}
\mathsf{le}(\mathsf{0}, y) &\to \mathsf{true}\\
\mathsf{le}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{false}\\
\mathsf{le}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{le}(x, y)\\
\mathsf{app}(\mathsf{nil}, y) &\to y\\
\mathsf{app}(\mathsf{add}(n, x), y) &\to \mathsf{add}(n, \mathsf{app}(x, y))\\
\mathsf{low}(n, \mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{low}(n, \mathsf{add}(m, x)) &\to \mathsf{if}_{\mathsf{low}}(\mathsf{le}(m, n), n, \mathsf{add}(m, x))\\
\mathsf{if}_{\mathsf{low}}(\mathsf{true}, n, \mathsf{add}(m, x)) &\to \mathsf{add}(m, \mathsf{low}(n, x))\\
\mathsf{if}_{\mathsf{low}}(\mathsf{false}, n, \mathsf{add}(m, x)) &\to \mathsf{low}(n, x)\\
\mathsf{high}(n, \mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{high}(n, \mathsf{add}(m, x)) &\to \mathsf{if}_{\mathsf{high}}(\mathsf{le}(m, n), n, \mathsf{add}(m, x))\\
\mathsf{if}_{\mathsf{high}}(\mathsf{true}, n, \mathsf{add}(m, x)) &\to \mathsf{high}(n, x)\\
\mathsf{if}_{\mathsf{high}}(\mathsf{false}, n, \mathsf{add}(m, x)) &\to \mathsf{add}(m, \mathsf{high}(n, x))\\
\mathsf{quicksort}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{quicksort}(\mathsf{add}(n, x)) &\to \mathsf{app}(\mathsf{quicksort}(\mathsf{low}(n, x)),\\
&\qquad\quad \mathsf{add}(n, \mathsf{quicksort}(\mathsf{high}(n, x))))
\end{aligned}
$$

Every set of inequalities associated with a cycle in the estimated dependency graph of this TRS is satisfied when we solve the inequalities resulting from the rules together with the following inequalities

$$
\begin{aligned}
\pi(\mathsf{LE}(\mathsf{s}(x),\mathsf{s}(y))) &\succ \pi(\mathsf{LE}(x,y))\\
\pi(\mathsf{APP}(\mathsf{add}(n,x),y)) &\succ \pi(\mathsf{APP}(x,y))\\
\pi(\mathsf{LOW}(n,\mathsf{add}(m,x))) &\succsim \pi(\mathsf{IF}_{\mathsf{low}}(\mathsf{le}(m,n),n,\mathsf{add}(m,x)))\\
\pi(\mathsf{IF}_{\mathsf{low}}(\mathsf{true},n,\mathsf{add}(m,x))) &\succ \pi(\mathsf{LOW}(n,x))\\
\pi(\mathsf{IF}_{\mathsf{low}}(\mathsf{false},n,\mathsf{add}(m,x))) &\succ \pi(\mathsf{LOW}(n,x))\\
\pi(\mathsf{HIGH}(n,\mathsf{add}(m,x))) &\succsim \pi(\mathsf{IF}_{\mathsf{high}}(\mathsf{le}(m,n),n,\mathsf{add}(m,x)))\\
\pi(\mathsf{IF}_{\mathsf{high}}(\mathsf{true},n,\mathsf{add}(m,x))) &\succ \pi(\mathsf{HIGH}(n,x))\\
\pi(\mathsf{IF}_{\mathsf{high}}(\mathsf{false},n,\mathsf{add}(m,x))) &\succ \pi(\mathsf{HIGH}(n,x))\\
\pi(\mathsf{QUICKSORT}(\mathsf{add}(n,x))) &\succ \pi(\mathsf{QUICKSORT}(\mathsf{low}(n,x)))\\
\pi(\mathsf{QUICKSORT}(\mathsf{add}(n,x))) &\succ \pi(\mathsf{QUICKSORT}(\mathsf{high}(n,x))).
\end{aligned}
$$

by applying the argument filtering $\pi(\mathsf{low}) = \pi(\mathsf{high}) = 2$, $\pi(\mathsf{if}_{\mathsf{low}}) = \pi(\mathsf{if}_{\mathsf{high}}) = 3$, $\pi(\mathsf{IF}_{\mathsf{low}}) = \pi(\mathsf{IF}_{\mathsf{high}}) = [2,3]$ and RPO. Since in the inequalities $\pi(\mathsf{LOW}(\ldots)) \succsim \pi(\mathsf{IF}_{\mathsf{low}}(\ldots))$ and $\pi(\mathsf{HIGH}(\ldots)) \succsim \pi(\mathsf{IF}_{\mathsf{high}}(\ldots))$ syntactically different terms are equivalent, this only proves DP-quasi simple termination (see the remarks in Ex. 3.10 on how to turn such a polynomial ordering into a QSO).

Steinbach could prove termination of a corresponding example with transformation orderings [Ste95a], but in his example the rules for le, $\mathsf{if}_{\mathsf{low}}$, $\mathsf{if}_{\mathsf{high}}$, and app were omitted.

If in the right-hand side of the last rule,

$$\mathsf{app}(\mathsf{quicksort}(\mathsf{low}(\mathbf{n},x)),\mathsf{add}(n,\mathsf{quicksort}(\mathsf{high}(\mathbf{n},x)))),$$

one of the **n**'s is replaced by a term containing $\mathsf{add}(n,x)$ then a non-simply terminating TRS is obtained. With our technique, termination can still be proved in the same way.

3.12 Permutation of lists

This example is a TRS from Walther [Wal94] to compute a permutation of a list. For instance, $\mathsf{shuffle}([1,2,3,4,5])$ reduces to $[1,5,2,4,3]$.

$$
\begin{aligned}
\mathsf{app}(\mathsf{nil},y) &\to y\\
\mathsf{app}(\mathsf{add}(n,x),y) &\to \mathsf{add}(n,\mathsf{app}(x,y))\\
\mathsf{reverse}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{reverse}(\mathsf{add}(n,x)) &\to \mathsf{app}(\mathsf{reverse}(x),\mathsf{add}(n,\mathsf{nil}))\\
\mathsf{shuffle}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{shuffle}(\mathsf{add}(n,x)) &\to \mathsf{add}(n,\mathsf{shuffle}(\mathsf{reverse}(x)))
\end{aligned}
$$

The cycles in the estimated dependency graph are

$$
\begin{aligned}
&\{\mathsf{APP}(\mathsf{add}(n,x),y) \to \mathsf{APP}(x,y)\}\\
&\{\mathsf{REVERSE}(\mathsf{add}(n,x)) \to \mathsf{REVERSE}(x)\}\\
&\{\mathsf{SHUFFLE}(\mathsf{add}(n,x)) \to \mathsf{SHUFFLE}(\mathsf{reverse}(x))\}.
\end{aligned}
$$

A suitable polynomial interpretation of the function symbols is: nil is mapped to 0, $\mathsf{add}(n, x)$ is mapped to $x + 1$, $\mathsf{shuffle}(x)$, $\mathsf{SHUFFLE}(x)$, $\mathsf{reverse}(x)$, and $\mathsf{REVERSE}(x)$ are mapped to x, and $\mathsf{app}(x, y)$ and $\mathsf{APP}(x, y)$ are mapped to $x+y$. This proves DP-quasi simple termination.

3.13 Reachability on directed graphs

To check whether there is a path from the node x to the node y in a directed graph g, the term $\mathsf{reach}(x, y, g, \epsilon)$ must be reducible to true with the rules of the following TRS from Giesl [Gie95]. The fourth argument of reach is used to store edges that have already been examined but that are not included in the actual solution path. If an edge from u to v (with $x \neq u$) is found, then it is rejected at first. If an edge from x to v (with $v \neq y$) is found then one either searches for further edges beginning in x (then one will never need the edge from x to v again) or one tries to find a path from v to y and now all edges that were rejected before have to be considered again.

The function union is used to unite two graphs. The constructor ϵ denotes the empty graph and $\mathsf{edge}(x, y, g)$ represents the graph g extended by an edge from x to y. Nodes are labelled with natural numbers.

$$
\begin{aligned}
\mathsf{eq}(0,0) &\to \mathsf{true}\\
\mathsf{eq}(0,\mathsf{s}(x)) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{s}(x),0) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{eq}(x,y)\\
\mathsf{or}(\mathsf{true},y) &\to \mathsf{true}\\
\mathsf{or}(\mathsf{false},y) &\to y\\
\mathsf{union}(\epsilon,h) &\to h\\
\mathsf{union}(\mathsf{edge}(x,y,i),h) &\to \mathsf{edge}(x,y,\mathsf{union}(i,h))\\
\mathsf{reach}(x,y,\epsilon,h) &\to \mathsf{false}\\
\mathsf{reach}(x,y,\mathsf{edge}(u,v,i),h) &\to \mathsf{if}_{\mathsf{reach\_1}}(\mathsf{eq}(x,u),x,y,\mathsf{edge}(u,v,i),h)\\
\mathsf{if}_{\mathsf{reach\_1}}(\mathsf{true},x,y,\mathsf{edge}(u,v,i),h) &\to \mathsf{if}_{\mathsf{reach\_2}}(\mathsf{eq}(y,v),x,y,\mathsf{edge}(u,v,i),h)\\
\mathsf{if}_{\mathsf{reach\_2}}(\mathsf{true},x,y,\mathsf{edge}(u,v,i),h) &\to \mathsf{true}\\
\mathsf{if}_{\mathsf{reach\_2}}(\mathsf{false},x,y,\mathsf{edge}(u,v,i),h) &\to \mathsf{or}(\mathsf{reach}(x,y,i,h),\\
&\qquad\quad \mathsf{reach}(v,y,\mathsf{union}(i,h),\epsilon))\\
\mathsf{if}_{\mathsf{reach\_1}}(\mathsf{false},x,y,\mathsf{edge}(u,v,i),h) &\to \mathsf{reach}(x,y,i,\mathsf{edge}(u,v,h))
\end{aligned}
$$

The inequalities obtained from dependency pairs on cycles in the estimated dependency graph are given by

$$
\begin{aligned}
\mathsf{EQ}(\mathsf{s}(x),\mathsf{s}(y)) &\succ \mathsf{EQ}(x,y)\\
\mathsf{UNION}(\mathsf{edge}(x,y,i),h) &\succ \mathsf{UNION}(i,h)\\
\mathsf{REACH}(x,y,\mathsf{edge}(u,v,i),h) &\succsim \mathsf{IF}_{\mathsf{reach\_1}}(\mathsf{eq}(x,u),x,y,\mathsf{edge}(u,v,i),h)
\end{aligned}
$$

$$\begin{aligned}
\mathsf{IF}_{\mathsf{reach_1}}(\mathsf{true}, x, y, \mathsf{edge}(u, v, i), h) &\succsim \mathsf{IF}_{\mathsf{reach_2}}(\mathsf{eq}(y, v), x, y, \mathsf{edge}(u, v, i), h)\\
\mathsf{IF}_{\mathsf{reach_2}}(\mathsf{false}, x, y, \mathsf{edge}(u, v, i), h) &\succ \mathsf{REACH}(x, y, i, h)\\
\mathsf{IF}_{\mathsf{reach_2}}(\mathsf{false}, x, y, \mathsf{edge}(u, v, i), h) &\succ \mathsf{REACH}(v, y, \mathsf{union}(i, h), \epsilon)\\
\mathsf{IF}_{\mathsf{reach_1}}(\mathsf{false}, x, y, \mathsf{edge}(u, v, i), h) &\succ \mathsf{REACH}(x, y, i, \mathsf{edge}(u, v, h)).
\end{aligned}$$

A mapping to polynomials results in a suitable ordering. The interpretation is: $\mathsf{eq}(x, y)$, true, false, ϵ, and $\mathsf{0}$ are mapped to 0, $\mathsf{or}(x, y)$ is mapped to $x + y$, $\mathsf{s}(x)$ is mapped to $x + 1$, $\mathsf{EQ}(x, y)$ is mapped to x, $\mathsf{edge}(x, y, g)$ is mapped to $g + 2$, $\mathsf{union}(g, h)$ and $\mathsf{UNION}(g, h)$ are mapped to $g + h$, $\mathsf{reach}(x, y, g, h)$, $\mathsf{if}_{\mathsf{reach\_1}}(b, x, y, g, h)$, and $\mathsf{if}_{\mathsf{reach\_2}}(b, x, y, g, h)$ are mapped to 0, $\mathsf{REACH}(x, y, g, h)$ is mapped to $(g+h)^2 + 2g + h + 2$, $\mathsf{IF}_{\mathsf{reach\_1}}(b, x, y, g, h)$ is mapped to $(g + h)^2 + 2g + h + 1$, and $\mathsf{IF}_{\mathsf{reach\_2}}(b, x, y, g, h)$ is mapped to $(g + h)^2 + 2g + h$.

Note that we showed DP quasi-simple termination of this TRS, since syntactically different terms in the $\succsim$-inequalities are mapped to the same number by this polynomial interpretation.

3.14 Comparison of binary trees

This TRS is used to find out if one binary tree has less leaves than another one. It uses a function $\mathsf{concat}(x, y)$ to replace the rightmost leaf of x by y. Here, $\mathsf{cons}(u, v)$ is used to built a tree with the two direct subtrees u and v.

$$\begin{aligned}
\mathsf{concat}(\mathsf{leaf}, y) &\to y\\
\mathsf{concat}(\mathsf{cons}(u, v), y) &\to \mathsf{cons}(u, \mathsf{concat}(v, y))\\
\mathsf{less_leaves}(x, \mathsf{leaf}) &\to \mathsf{false}\\
\mathsf{less_leaves}(\mathsf{leaf}, \mathsf{cons}(w, z)) &\to \mathsf{true}\\
\mathsf{less_leaves}(\mathsf{cons}(u, v), \mathsf{cons}(w, z)) &\to \mathsf{less_leaves}(\mathsf{concat}(u, v), \mathsf{concat}(w, z))
\end{aligned}$$

The cycles in the dependency graph are:

$$\begin{aligned}
&\{\mathsf{CONCAT}(\mathsf{cons}(u, v), y) \to \mathsf{CONCAT}(v, y)\}\\
&\{\mathsf{LESS_LEAVES}(\mathsf{cons}(u, v), \mathsf{cons}(w, z)) \to \mathsf{LESS_LEAVES}(\mathsf{concat}(u, v), \mathsf{concat}(w, z))\}.
\end{aligned}$$

A suitable (polynomial) interpretation for DP-quasi simple termination is: leaf, false, and true are mapped to 0, $\mathsf{cons}(u, v)$ is mapped to $1+u+v$, $\mathsf{concat}(u, v)$ and $\mathsf{CONCAT}(u, v)$ are mapped to $u+v$, and $\mathsf{less\_leaves}(x, y)$ and $\mathsf{LESS\_LEAVES}(x, y)$ are mapped to x.

If $\mathsf{concat}(w, z)$ in the second argument of $\mathsf{less\_leaves}$ (in the right-hand side of the last rule) would be replaced by an appropriate argument, we would obtain a non-simply terminating TRS whose termination could be proved in the same way.

3.15 Average of naturals

The following locally confluent overlay system computes the average of two numbers [DH95].

$$
\begin{aligned}
\mathsf{average}(\mathsf{s}(x), y) &\to \mathsf{average}(x, \mathsf{s}(y)) \\
\mathsf{average}(x, \mathsf{s}(\mathsf{s}(\mathsf{s}(y)))) &\to \mathsf{s}(\mathsf{average}(\mathsf{s}(x), y)) \\
\mathsf{average}(\mathsf{0}, \mathsf{0}) &\to \mathsf{0} \\
\mathsf{average}(\mathsf{0}, \mathsf{s}(\mathsf{0})) &\to \mathsf{0} \\
\mathsf{average}(\mathsf{0}, \mathsf{s}(\mathsf{s}(\mathsf{0}))) &\to \mathsf{s}(\mathsf{0})
\end{aligned}
$$

The inequalities resulting from the cycles are

$$
\begin{aligned}
\mathsf{AVERAGE}(\mathsf{s}(x), y) &\succ \mathsf{AVERAGE}(x, \mathsf{s}(y)) \\
\mathsf{AVERAGE}(x, \mathsf{s}(\mathsf{s}(\mathsf{s}(y)))) &\succ \mathsf{AVERAGE}(\mathsf{s}(x), y).
\end{aligned}
$$

By the following polynomial interpretation, DP-quasi simple termination of this TRS is easily proved: $\mathsf{0}$ is mapped to 0, $\mathsf{s}(x)$ is mapped to $x + 1$, $\mathsf{average}(x, y)$ is mapped to $x + y$, and $\mathsf{AVERAGE}(x, y)$ is mapped to $2x + y$.

3.16 Plus and times

The following TRS [DH95] is a locally confluent overlay system. To ease readability we use an infix notation for $+$ and $\times$.

$$
\begin{aligned}
x \times \mathsf{0} &\to \mathsf{0} \\
x \times \mathsf{s}(y) &\to (x \times y) + x \\
x + \mathsf{0} &\to x \\
\mathsf{0} + x &\to x \\
x + \mathsf{s}(y) &\to \mathsf{s}(x + y) \\
\mathsf{s}(x) + y &\to \mathsf{s}(x + y)
\end{aligned}
$$

Applying the technique results in a set of inequalities which is satisfied by the polynomial interpretation where $\mathsf{0}$ is mapped to 0, $\mathsf{s}(x)$ is mapped to $x+1$, $x+y$ is mapped to the sum of x and y, $x \times y$ is mapped to the product of x and y, $\mathsf{TIMES}(x, y)$ is mapped to y, and $\mathsf{PLUS}(x, y)$ is mapped to the sum of x and y (where PLUS denotes '$+^\sharp$').

3.17 Summing elements of lists

This TRS, which has overlapping rules, can be used to compute the sum of all elements of a list [AG97a]. Here, $x{\bullet}l$ represents the insertion of a number x into a list l (where $x{\bullet}y{\bullet}l$ abbreviates $(x{\bullet}(y{\bullet}l))$), app computes the concatenation of lists, and $\mathsf{sum}(l)$ is used to compute the sum of all numbers in l (e.g., sum applied to the list $[1,2,3]$ returns $[1+2+3]$).

$$\begin{aligned}
\mathsf{app}(\mathsf{nil},k) &\to k\\
\mathsf{app}(l,\mathsf{nil}) &\to l\\
\mathsf{app}(x{\bullet}l,k) &\to x{\bullet}\mathsf{app}(l,k)\\
\mathsf{sum}(x{\bullet}\mathsf{nil}) &\to x{\bullet}\mathsf{nil}\\
\mathsf{sum}(x{\bullet}y{\bullet}l) &\to \mathsf{sum}((x+y){\bullet}l)\\
\mathsf{sum}(\mathsf{app}(l,x{\bullet}y{\bullet}k)) &\to \mathsf{sum}(\mathsf{app}(l,\mathsf{sum}(x{\bullet}y{\bullet}k)))\\
\mathsf{0}+y &\to y\\
\mathsf{s}(x)+y &\to \mathsf{s}(x+y)
\end{aligned}$$

While this system is not simply terminating, the inequalities generated by the technique are satisfied by the polynomial ordering where nil is mapped to the constant 0, $x{\bullet}l$ is mapped to $l+1$, $x+y$ is mapped to the sum of x and y, $\mathsf{app}(l,k)$ is mapped to $l+k+1$, $\mathsf{sum}(l)$ is mapped to the constant 1, $\mathsf{APP}(l,k)$ and $\mathsf{SUM}(l)$ are both mapped to l, and $\mathsf{PLUS}(x,y)$ is mapped to x. In this way we have shown DP quasi-simple termination. The polynomial interpretation is such that the syntactically unequal terms $\mathsf{sum}(x{\bullet}y{\bullet}l)$ and $\mathsf{sum}((x+y){\bullet}l)$ are mapped to the same value.

DP simple termination of this system can also be shown by first applying the argument filtering $\pi(\bullet)=[2]$, $\pi(\mathsf{sum})=[\,]$, $\pi(\mathsf{SUM})=\pi(\mathsf{APP})=1$. Now the inequalities

$$\begin{aligned}
\pi(\mathsf{sum}(x{\bullet}y{\bullet}l)) &\succsim \pi(\mathsf{sum}((x+y){\bullet}l))\\
\pi(\mathsf{sum}(\mathsf{app}(l,x{\bullet}y{\bullet}k))) &\succsim \pi(\mathsf{sum}(\mathsf{app}(l,\mathsf{sum}(x{\bullet}y{\bullet}k))))
\end{aligned}$$

have syntactically identical left- and right-hand sides. For all other inequalities we need to give an ordering that satisfies them in a strict way. We provide again a polynomial interpretation, viz. $\mathsf{0}$ and nil are mapped to 0, $\mathsf{s}(x)$ is mapped to $x+1$, ${\bullet}l$ is mapped to $l+2$, $\mathsf{PLUS}(x,y)$ is mapped to $x+y$, sum is mapped to 3, and both $\mathsf{app}(x,y)$ and $x+y$ are mapped to $2x+y+1$.

If the above TRS is extended by the rules

$$\begin{aligned}
\mathsf{sum}(\mathsf{0}{\bullet}x+y{\bullet}l) &\to \mathsf{pred}(\mathsf{sum}(\mathsf{s}(x){\bullet}y{\bullet}l))\\
\mathsf{pred}(\mathsf{s}(x){\bullet}\mathsf{nil}) &\to x{\bullet}\mathsf{nil},
\end{aligned}$$

then DP quasi-simple termination can still be proved by the first polynomial ordering (where the polynomial interpretation should map $\mathsf{pred}(l)$ to the constant 1).

3.18 Addition and subtraction

The following system is again overlapping and not simply terminating.

$$
\begin{aligned}
\mathsf{minus}(x,\mathsf{0}) &\to x\\
\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{minus}(x,y)\\
\mathsf{double}(\mathsf{0}) &\to \mathsf{0}\\
\mathsf{double}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))\\
\mathsf{plus}(\mathsf{0},y) &\to y\\
\mathsf{plus}(\mathsf{s}(x),y) &\to \mathsf{s}(\mathsf{plus}(x,y))\\
\mathsf{plus}(\mathsf{s}(x),y) &\to \mathsf{plus}(x,\mathsf{s}(y))\\
\mathsf{plus}(\mathsf{s}(x),y) &\to \mathsf{s}(\mathsf{plus}(\mathsf{minus}(x,y),\mathsf{double}(y)))
\end{aligned}
$$

After applying the argument filtering $\pi(\mathsf{minus}) = 1$, the inequalities generated for DP simple termination by our technique are satisfied by the lexicographic path ordering.

3.19 Addition with nested recursion, version 1

If the following additional rule is added to the above system, then it is turned into a TRS that is not an overlay system any more and which furthermore introduces nested recursion.

$$\mathsf{plus}(\mathsf{s}(\mathsf{plus}(x,y)),z) \;\to\; \mathsf{s}(\mathsf{plus}(\mathsf{plus}(x,y),z))$$

Still, the resulting inequalities are satisfied using the same argument filtering and the lexicographic path ordering.

3.20 Addition with nested recursion, version 2

The following alternative TRS for addition from Steinbach [Ste95a] has nested recursion, too.

$$
\begin{aligned}
\mathsf{0}+y &\to y\\
\mathsf{s}(x)+\mathsf{0} &\to \mathsf{s}(x)\\
\mathsf{s}(x)+\mathsf{s}(y) &\to \mathsf{s}(\mathsf{s}(x)+(y+\mathsf{0}))
\end{aligned}
$$

The 'natural' polynomial interpretation (where $+$ is mapped to the addition) maps left and right-hand sides of the rules to the same numbers. Therefore this polynomial ordering cannot be used for a direct termination proof, but it nevertheless satisfies the inequalities generated by the dependency pair technique. In this way, DP-quasi simple termination can easily be proved.

3.21 Multiplication and addition

The following example is taken from Dershowitz [Der87].

$$
\begin{aligned}
x\times(y+\mathsf{1}) &\to (x\times(y+(\mathsf{1}\times\mathsf{0})))+x\\
x\times\mathsf{1} &\to x\\
x+\mathsf{0} &\to x\\
x\times\mathsf{0} &\to \mathsf{0}
\end{aligned}
$$

The only inequality resulting from a dependency pair on a cycle in the estimated dependency graph is $\mathsf{TIMES}(x, y + 1) \succ \mathsf{TIMES}(x, y + (1 \times 0))$.

This system is not simply terminating (and Dershowitz illustrates the use of the semantic path ordering with it). However, termination of this example can be proved automatically. The inequalities obtained are satisfied by the natural polynomial ordering, where $\mathsf{TIMES}(x, y)$ is mapped to y.

By choosing the natural interpretation on numbers, the terms $x \times \mathsf{0}$ in the inequality corresponding to the last rule are equivalent to $\mathsf{0}$, even though syntactically they are not equal. Therefore, we have shown DP quasi-simple termination of this TRS.

3.22 Extended multiplication and addition

Similarly, termination of the following 'extended' version of the above system can be proved. In this system, the full rules for $+$ and $\times$ are added. Again, this system is not an overlay system.

$$\begin{aligned}
x \times (y + \mathsf{s}(z)) &\to (x \times (y + (\mathsf{s}(z) \times \mathsf{0}))) + (x \times \mathsf{s}(z)) \\
x \times \mathsf{0} &\to \mathsf{0} \\
x \times \mathsf{s}(y) &\to (x \times y) + x \\
x + \mathsf{0} &\to x \\
x + \mathsf{s}(y) &\to \mathsf{s}(x + y)
\end{aligned}$$

The generated inequalities for this extended example, i.e., the inequalities corresponding to the rewrite rules and

$$\begin{aligned}
\mathsf{TIMES}(x, y + \mathsf{s}(z)) &\succsim \mathsf{TIMES}(x, \mathsf{s}(z)) \\
\mathsf{TIMES}(x, y + \mathsf{s}(z)) &\succ \mathsf{TIMES}(x, y + (\mathsf{s}(z) \times \mathsf{0})) \\
\mathsf{TIMES}(x, \mathsf{s}(y)) &\succ \mathsf{TIMES}(x, y) \\
\mathsf{PLUS}(x, \mathsf{s}(y)) &\succ \mathsf{PLUS}(x, y)
\end{aligned}$$

are satisfied by the same polynomial ordering that has been used above (where $\mathsf{PLUS}(x, y)$ and $\mathsf{TIMES}(x, y)$ are both mapped to y).

3.23 Nested recursion, version 1

The following system was introduced by Giesl [Gie97, '$\mathsf{nest2}$'] as an example for a small TRS with nested recursion where all simplification orderings fail.

$$\begin{aligned}
\mathsf{f}(\mathsf{0}, y) &\to \mathsf{0} \\
\mathsf{f}(\mathsf{s}(x), y) &\to \mathsf{f}(\mathsf{f}(x, y), y)
\end{aligned}$$

For this example, a polynomial ordering can be used where $\mathsf{0}$ and s are interpreted as usual and both $\mathsf{f}(x, y)$ and $\mathsf{F}(x, y)$ are mapped to x.

Alternatively, one can use the argument filtering $\pi(\mathsf{f}) = 1$ and RPO to prove termination. In that way, one easily sees that the system is DP simply terminating.

3.24 Nested recursion, version 2

This system by Walther, which is similar to the preceding one, has been examined in [Ste95a].

$$\begin{aligned} \mathsf{f}(\mathsf{0}) &\to \mathsf{s}(\mathsf{0}) \\ \mathsf{f}(\mathsf{s}(\mathsf{0})) &\to \mathsf{s}(\mathsf{0}) \\ \mathsf{f}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{f}(\mathsf{f}(\mathsf{s}(x))) \end{aligned}$$

The inequalities resulting from our transformation are satisfied by the polynomial ordering, where $\mathsf{f}(x)$ is mapped to the constant 1, $\mathsf{F}(x)$ is mapped to x, and where $\mathsf{0}$ and s are interpreted as usual. In this way, we have shown DP quasi-simple termination of this TRS.

3.25 Nested recursion, version 3

The following TRS by Ferreira and Zantema [FZ93] is a string rewrite system with minimal ordinal ω^ω associated to it.

$$\begin{aligned} \mathsf{f}(\mathsf{g}(x)) &\to \mathsf{g}(\mathsf{f}(\mathsf{f}(x))) \\ \mathsf{f}(\mathsf{h}(x)) &\to \mathsf{h}(\mathsf{g}(x)) \end{aligned}$$

The cycles in the estimated dependency graph are

$$\begin{array}{l} \{\mathsf{F}(\mathsf{g}(x)) \to \mathsf{F}(x)\} \\ \{\mathsf{F}(\mathsf{g}(x)) \to \mathsf{F}(\mathsf{f}(x))\} \\ \{\mathsf{F}(\mathsf{g}(x)) \to \mathsf{F}(x), \mathsf{F}(\mathsf{g}(x)) \to \mathsf{F}(\mathsf{f}(x))\}. \end{array}$$

After applying the argument filtering $\pi(\mathsf{h}) = [\,]$, $\pi(\mathsf{f}) = 1$, all inequalities are satisfied by the recursive path ordering. This shows that the system is DP simply terminating.

3.26 Nested recursion, version 4

The following TRS is again an example of a TRS for which all kind of path orderings cannot show termination directly, but these path orderings can be used for solving the inequalities resulting from our technique.

$$\begin{aligned} \mathsf{f}(x) &\to \mathsf{s}(x) \\ \mathsf{f}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{f}(\mathsf{f}(x))) \end{aligned}$$

The inequalities to satisfy are

$$\begin{aligned} \mathsf{f}(x) &\succsim \mathsf{s}(x) \\ \mathsf{f}(\mathsf{s}(\mathsf{s}(x))) &\succsim \mathsf{s}(\mathsf{f}(\mathsf{f}(x))) \\ \mathsf{F}(\mathsf{s}(\mathsf{s}(x))) &\succ \mathsf{F}(x) \\ \mathsf{F}(\mathsf{s}(\mathsf{s}(x))) &\succ \mathsf{F}(\mathsf{f}(x)). \end{aligned}$$

An appropriate path ordering is found by choosing f and s to be equal in the precedence. Note that therefore we proved DP quasi-simple termination of the system.

3.27 Nested symbols on left-hand sides

The following example is from Dershowitz [Der93]. It has been proved terminating by a lexicographic combination of two orderings.

$$\begin{aligned} \mathsf{f}(\mathsf{f}(x)) &\to \mathsf{g}(\mathsf{f}(x)) \\ \mathsf{g}(\mathsf{g}(x)) &\to \mathsf{f}(x) \end{aligned}$$

The inequalities corresponding to dependency pairs on cycles in the estimated dependency graph are

$$\begin{aligned} \mathsf{F}(\mathsf{f}(x)) &\succ \mathsf{F}(x) \\ \mathsf{F}(\mathsf{f}(x)) &\succsim \mathsf{G}(\mathsf{f}(x)) \\ \mathsf{G}(\mathsf{g}(x)) &\succ \mathsf{F}(x). \end{aligned}$$

By choosing f and g as well as F and G equal in the precedence, the inequalities are satisfied by the recursive path ordering. Again, this shows DP quasi-simple termination of the TRS.

3.28 Nested symbols on both sides of rules

Termination of the following TRS cannot be proved by the lexicographic path ordering and therefore this is one of the systems for which the semantic path ordering has been used in literature [Der93]. However, the system can be shown to terminate using the lexicographic path ordering after applying our technique, since the demanded ordering may now be a *weakly* monotonic ordering instead of a monotonic ordering. Therefore, after mapping some function symbols to some of their arguments or to a constant the lexicographic path ordering can nevertheless be used to prove termination of the TRS.

$$\begin{aligned} (x \times y) \times z &\to x \times (y \times z) \\ (x + y) \times z &\to (x \times z) + (y \times z) \\ z \times (x + \mathsf{f}(y)) &\to \mathsf{g}(z, y) \times (x + \mathsf{a}) \end{aligned}$$

Apart from the three inequalities corresponding to the rewrite rules, four other inequalities are obtained from the cycles in the dependency graph.

$$\begin{aligned} \mathsf{TIMES}(x \times y, z) &\succ \mathsf{TIMES}(y, z) \\ \mathsf{TIMES}(x \times y, z) &\succ \mathsf{TIMES}(x, y \times z) \\ \mathsf{TIMES}(x + y, z) &\succ \mathsf{TIMES}(x, z) \\ \mathsf{TIMES}(x + y, z) &\succ \mathsf{TIMES}(y, z) \end{aligned}$$

After applying the argument filtering $\pi(\mathsf{g}) = 1$, the seven inequalities are satisfied by the lexicographic path ordering, which proves DP simple termination.

3.29 A TRS that is not left-linear, version 1

The following TRS, originally from Geerling [Gee91], cannot be proved terminating by the recursive path ordering (but one needs a generalization of the recursive path ordering as defined by Ferreira [Fer95]). It is also very easily proved terminating by the automatic technique described in this paper.

$$\mathsf{f}(\mathsf{s}(x), y, y) \to \mathsf{f}(y, x, \mathsf{s}(x))$$

The only two generated inequalities are

$$\begin{aligned} \mathsf{f}(\mathsf{s}(x), y, y) &\succsim \mathsf{f}(y, x, \mathsf{s}(x)) \\ \mathsf{F}(\mathsf{s}(x), y, y) &\succ \mathsf{F}(y, x, \mathsf{s}(x)) \end{aligned}$$

which are satisfied by mapping $\mathsf{f}(x, y, z)$ to 0, mapping $\mathsf{s}(x)$ to $x+1$, and mapping $\mathsf{F}(x, y, z)$ to $x + y$. For showing DP simple termination of this TRS, one can use the argument filtering $\pi(\mathsf{f}) = [\,]$, $\pi(\mathsf{F}) = [1, 2]$ and RPO.

3.30 Advantage of the dependency graph, version 1

The following system is from [Ste95a].

$$\begin{aligned} \mathsf{f}(\mathsf{a}, \mathsf{b}) &\to \mathsf{f}(\mathsf{a}, \mathsf{c}) \\ \mathsf{f}(\mathsf{c}, \mathsf{d}) &\to \mathsf{f}(\mathsf{b}, \mathsf{d}) \end{aligned}$$

With our method, the termination proof for this system is trivial, because its estimated dependency graph does not contain any cycles. Similar, termination of the one rule TRS $\mathsf{f}(\mathsf{g}(x)) \to \mathsf{f}(\mathsf{h}(\mathsf{g}(x)))$ from Bellegarde and Lescanne [BL88] and of the one rule system $\mathsf{f}(\mathsf{g}(x, y), y) \to \mathsf{f}(\mathsf{h}(\mathsf{g}(x, y)), \mathsf{a})$ from Steinbach [Ste95a] can also be proved by absence of cycles.

3.31 Advantage of the dependency graph, version 2

Another example where the dependency graph plays an important role is a TRS introduced by Ferreira and Zantema [FZ95] to demonstrate the technique of 'dummy elimination'.

$$\mathsf{f}(\mathsf{g}(x)) \to \mathsf{f}(\mathsf{a}(\mathsf{g}(\mathsf{g}(\mathsf{f}(x))), \mathsf{g}(\mathsf{f}(x))))$$

Since $\mathsf{F}(\mathsf{a}(y, z))$ does not unify with $\mathsf{F}(\mathsf{g}(x))$, the only two inequalities to satisfy are

$$\begin{aligned} \pi(\mathsf{f}(\mathsf{g}(x))) &\succsim \pi(\mathsf{f}(\mathsf{a}(\mathsf{g}(\mathsf{g}(\mathsf{f}(x))), \mathsf{g}(\mathsf{f}(x))))) \\ \pi(\mathsf{F}(\mathsf{g}(x))) &\succ \pi(\mathsf{F}(x)). \end{aligned}$$

For $\pi(\mathsf{a}) = [\,]$ these inequalities are trivially satisfied by the recursive path ordering and DP simple termination of the TRS is shown. For a thorough comparison of dependency pairs and dummy elimination see [GM00].

3.32 A TRS that is not totally terminating, version 1

The most famous example of a TRS that is terminating, but not *totally* terminating is the following [Der87].

$$\begin{aligned} \mathsf{f}(\mathsf{a}) &\to \mathsf{f}(\mathsf{b}) \\ \mathsf{g}(\mathsf{b}) &\to \mathsf{g}(\mathsf{a}) \end{aligned}$$

With our approach, termination of this system is obvious, because the estimated dependency graph does not contain any cycles.

3.33 A TRS that is not totally terminating, version 2

A TRS introduced by Ferreira [Fer95] as an example of a TRS that is not totally terminating and in particular for which the recursive path ordering and the Knuth-Bendix ordering cannot be used to prove termination, is given by:

$$\begin{aligned} \mathsf{p}(\mathsf{f}(\mathsf{f}(x))) &\to \mathsf{q}(\mathsf{f}(\mathsf{g}(x))) \\ \mathsf{p}(\mathsf{g}(\mathsf{g}(x))) &\to \mathsf{q}(\mathsf{g}(\mathsf{f}(x))) \\ \mathsf{q}(\mathsf{f}(\mathsf{f}(x))) &\to \mathsf{p}(\mathsf{f}(\mathsf{g}(x))) \\ \mathsf{q}(\mathsf{g}(\mathsf{g}(x))) &\to \mathsf{p}(\mathsf{g}(\mathsf{f}(x))). \end{aligned}$$

Termination is trivially concluded from the fact that there are no cycles in the estimated dependency graph.

3.34 Systems with 'undefined' function symbols

The following well-known system from Dershowitz [Der87] is one of the smallest non-simply terminating TRSs.

$$\mathsf{f}(\mathsf{f}(x)) \;\to\; \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))$$

The only dependency pair on a cycle of the estimated dependency graph is $\mathsf{F}(\mathsf{f}(x)) \to \mathsf{F}(x)$. By the argument filtering $\pi(\mathsf{g}) = 1$ and RPO the system is shown DP simply terminating.

3.35 Mutual recursion, version 1

The following system is from Steinbach [Ste95a] again.

$$\begin{aligned} \mathsf{g}(\mathsf{s}(x)) &\to \mathsf{f}(x) \\ \mathsf{f}(\mathsf{0}) &\to \mathsf{s}(\mathsf{0}) \\ \mathsf{f}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{s}(\mathsf{g}(x))) \\ \mathsf{g}(\mathsf{0}) &\to \mathsf{0} \end{aligned}$$

The inequalities resulting from cycles are

$$\pi(\mathsf{G}(\mathsf{s}(x))) \succsim \pi(\mathsf{F}(x))$$
$$\pi(\mathsf{F}(\mathsf{s}(x))) \succ \pi(\mathsf{G}(x)).$$

After applying the argument filtering $\pi(\mathsf{g}) = 1$, the constraints are satisfied by the recursive path ordering. Since s and f have to be equal in the precedence in order to satisfy the resulting inequalities $\mathsf{s}(x) \succsim \mathsf{f}(x)$ and $\mathsf{f}(0) \succsim \mathsf{s}(0)$, this proves DP-quasi simple termination.

3.36 Mutual recursion, version 2

The following system was given to us by Kühler.

$$\begin{aligned}
\mathsf{minus}(x, 0) &\to x \\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(x, y) \\
\mathsf{f}(0) &\to \mathsf{s}(0) \\
\mathsf{f}(\mathsf{s}(x)) &\to \mathsf{minus}(\mathsf{s}(x), \mathsf{g}(\mathsf{f}(x))) \\
\mathsf{g}(0) &\to 0 \\
\mathsf{g}(\mathsf{s}(x)) &\to \mathsf{minus}(\mathsf{s}(x), \mathsf{f}(\mathsf{g}(x)))
\end{aligned}$$

The inequalities resulting from dependency pairs on cycles of the estimated dependency graph are

$$\begin{aligned}
\pi(\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y))) &\succ \pi(\mathsf{MINUS}(x, y)) \\
\pi(\mathsf{F}(\mathsf{s}(x))) &\succ \pi(\mathsf{F}(x)) \\
\pi(\mathsf{F}(\mathsf{s}(x))) &\succsim \pi(\mathsf{G}(\mathsf{f}(x))) \\
\pi(\mathsf{G}(\mathsf{s}(x))) &\succ \pi(\mathsf{G}(x)) \\
\pi(\mathsf{G}(\mathsf{s}(x))) &\succ \pi(\mathsf{F}(\mathsf{g}(x))).
\end{aligned}$$

After applying the argument filtering $\pi(\mathsf{minus}) = 1$, the resulting inequalities are satisfied by the recursive path ordering (using a precedence where f and s are equal and greater than g). Thus, the system is DP quasi-simply terminating.

3.37 Even and odd

The following (non-simply terminating) TRS can be used to find out whether a natural number is even resp. odd. More precisely, $\mathsf{evenodd}(t, 0)$ reduces to true if t is even and $\mathsf{evenodd}(t, \mathsf{s}(0))$ reduces to true if t is odd. (In other words, the second argument of $\mathsf{evenodd}$ determines whether $\mathsf{evenodd}$ computes the 'even' or the 'odd' function. Such rewrite systems are often obtained when transforming mutually recursive functions into one function without mutual recursion, cf. [Gie97].)

$$
\begin{aligned}
\mathsf{not}(\mathsf{true}) &\to \mathsf{false}\\
\mathsf{not}(\mathsf{false}) &\to \mathsf{true}\\
\mathsf{evenodd}(x,\mathsf{0}) &\to \mathsf{not}(\mathsf{evenodd}(x,\mathsf{s}(\mathsf{0})))\\
\mathsf{evenodd}(\mathsf{0},\mathsf{s}(\mathsf{0})) &\to \mathsf{false}\\
\mathsf{evenodd}(\mathsf{s}(x),\mathsf{s}(\mathsf{0})) &\to \mathsf{evenodd}(x,\mathsf{0})
\end{aligned}
$$

We obtain one cycle in the estimated dependency graph.

$$
\begin{aligned}
\{&\mathsf{EVENODD}(x,\mathsf{0}) \to \mathsf{EVENODD}(x,\mathsf{s}(\mathsf{0})),\\
&\mathsf{EVENODD}(\mathsf{s}(x),\mathsf{s}(\mathsf{0})) \to \mathsf{EVENODD}(x,\mathsf{0})\}
\end{aligned}
$$

With the argument filtering $\pi(\mathsf{not}) = [\,]$, $\pi(\mathsf{EVENODD}) = 1$ and the recursive path ordering, DP simple termination is shown.

3.38 Reversing lists

The following system is a slight variant of a TRS proposed by Huet and Hullot [HH82, 'brev']. Given a list $x\bullet l$, the function rev calls two other functions $\mathsf{rev1}$ and $\mathsf{rev2}$, where $\mathsf{rev1}(x,l)$ returns the last element of $x\bullet l$ and $\mathsf{rev2}(x,l)$ returns the reversed list $\mathsf{rev}(x\bullet l)$ *without its first element*. Hence, $\mathsf{rev}(\mathsf{rev2}(y,l))$ returns the list $y\bullet l$ without its last element. Note that this system is mutually recursive and that mutually recursive functions also occur nested.

$$
\begin{aligned}
\mathsf{rev}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{rev}(x\bullet l) &\to \mathsf{rev1}(x,l)\bullet\mathsf{rev2}(x,l)\\
\mathsf{rev1}(\mathsf{0},\mathsf{nil}) &\to \mathsf{0}\\
\mathsf{rev1}(\mathsf{s}(x),\mathsf{nil}) &\to \mathsf{s}(x)\\
\mathsf{rev1}(x,y\bullet l) &\to \mathsf{rev1}(y,l)\\
\mathsf{rev2}(x,\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{rev2}(x,y\bullet l) &\to \mathsf{rev}(x\bullet\mathsf{rev}(\mathsf{rev2}(y,l)))
\end{aligned}
$$

The inequalities resulting from the cycles of the estimated dependency graph are

$$
\begin{aligned}
\pi(\mathsf{REV}(x\bullet l)) &\succ \pi(\mathsf{REV2}(x,l))\\
\pi(\mathsf{REV1}(x,y\bullet l)) &\succ \pi(\mathsf{REV1}(y,l))\\
\pi(\mathsf{REV2}(x,y\bullet l)) &\succ \pi(\mathsf{REV2}(y,l))\\
\pi(\mathsf{REV2}(x,y\bullet l)) &\succ \pi(\mathsf{REV}(\mathsf{rev2}(y,l)))\\
\pi(\mathsf{REV2}(x,y\bullet l)) &\succsim \pi(\mathsf{REV}(x\bullet\mathsf{rev}(\mathsf{rev2}(y,l)))).
\end{aligned}
$$

By using the argument filtering $\pi(\bullet) = [2]$, $\pi(\mathsf{s}) = [\,]$, $\pi(\mathsf{rev}) = \pi(\mathsf{REV}) = 1$, $\pi(\mathsf{rev1}) = \pi(\mathsf{rev2}) = \pi(\mathsf{REV1}) = \pi(\mathsf{REV2}) = 2$, the resulting constraints are satisfied by the recursive path ordering. This proves DP simple termination of the TRS.

3.39 Narrowing of dependency pairs

The following example [AG00] demonstrates the need for narrowing dependency pairs. We replace the last rule of the TRS in Ex. 3.4 by a 'commutativity' rule:

$$\begin{aligned}
x - \mathsf{0} &\to x \\
\mathsf{s}(x) - \mathsf{s}(y) &\to x - y \\
\mathsf{quot}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{0} \\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{quot}(x - y, \mathsf{s}(y))) \\
\mathsf{0} + y &\to y \\
\mathsf{s}(x) + y &\to \mathsf{s}(x + y) \\
(x - \mathsf{s}(\mathsf{0})) + (y - \mathsf{s}(\mathsf{s}(z))) &\to (y - \mathsf{s}(\mathsf{s}(z))) + (x - \mathsf{s}(\mathsf{0})).
\end{aligned}$$

Without the use of narrowing, we would obtain the constraint

$$\pi(\mathsf{PLUS}(x - \mathsf{s}(\mathsf{0}), y - \mathsf{s}(\mathsf{s}(z)))) \succ \pi(\mathsf{PLUS}(y - \mathsf{s}(\mathsf{s}(z)), x - \mathsf{s}(\mathsf{0}))),$$

because the dependency pair $\mathsf{PLUS}(x - \mathsf{s}(\mathsf{0}), y - \mathsf{s}(\mathsf{s}(z))) \to \mathsf{PLUS}(y - \mathsf{s}(\mathsf{s}(z)), x - \mathsf{s}(\mathsf{0}))$ forms a cycle of the estimated dependency graph. In order to use a simplification ordering we have to chose an argument filtering π such that $\pi(-) = [1]$ or $\pi(-) = 1$. However, then this constraint is not satisfied by any well-founded ordering closed under substitution. Therefore we replace this dependency pair by its narrowings

$$\begin{aligned}
&\mathsf{PLUS}(x - \mathsf{s}(\mathsf{0}), \mathsf{s}y - \mathsf{s}(\mathsf{s}(z))) \to \mathsf{PLUS}(y - \mathsf{s}(z), x - \mathsf{s}(\mathsf{0})) \\
&\mathsf{PLUS}(\mathsf{s}(x) - \mathsf{s}(\mathsf{0}), y - \mathsf{s}(\mathsf{s}(z))) \to \mathsf{PLUS}(y - \mathsf{s}(\mathsf{s}(z)), x - \mathsf{0}).
\end{aligned}$$

Now the resulting constraints are again satisfied by the recursive path ordering if we use the argument filtering $\pi(-) = \pi(-^\sharp) = 1$.

3.40 Narrowing to approximate the dependency graph

Narrowing of dependency pairs may also be helpful in examples where the failure of the automation is due to our approximation of dependency graphs. For example, let us add the following second 'commutation' rule to the TRS from Ex. 3.39

$$(x + \mathsf{s}(\mathsf{0})) \ + \ (y + \mathsf{s}(\mathsf{s}(z))) \to (y + \mathsf{s}(\mathsf{s}(z))) \ + \ (x + \mathsf{s}(\mathsf{0})).$$

Now we obtain three additional dependency pairs.

$$\begin{aligned}
&\mathsf{PLUS}(x + \mathsf{s}(\mathsf{0}), y + \mathsf{s}(\mathsf{s}(z))) \to \mathsf{PLUS}(y, \mathsf{s}(\mathsf{s}(z))) && (1) \\
&\mathsf{PLUS}(x + \mathsf{s}(\mathsf{0}), y + \mathsf{s}(\mathsf{s}(z))) \to \mathsf{PLUS}(x, \mathsf{s}(\mathsf{0})) && (2) \\
&\mathsf{PLUS}(x + \mathsf{s}(\mathsf{0}), y + \mathsf{s}(\mathsf{s}(z))) \to \mathsf{PLUS}(y + \mathsf{s}(\mathsf{s}(z)), x + \mathsf{s}(\mathsf{0})) && (3)
\end{aligned}$$

We have to compute a graph containing the dependency graph. For that purpose, we draw an arc from a dependency pair $s \to t$ to $v \to w$ whenever $\textsc{ren}(\textsc{cap}(t))$ and v are unifiable. However, for some examples this approximation is too rough.

Note that in our approximation of the dependency graph there would be an arc from (3) to itself, because after replacing $y + \mathsf{s}(\mathsf{s}(z))$ and $x + \mathsf{s}(0)$ by new variables, the right- and the left-hand side of (3) obviously unify. Hence, we have to demand that the dependency pair (3) is strictly decreasing, i.e.,

$$\pi(\mathsf{PLUS}(x + \mathsf{s}(0), y + \mathsf{s}(\mathsf{s}(z)))) \succ \pi(\mathsf{PLUS}(y + \mathsf{s}(\mathsf{s}(z)), x + \mathsf{s}(0))).$$

But this constraint is not satisfied by any polynomial or any path ordering amenable to automation[1].

However, in the *real* dependency graph, there is no arc from (3) to itself, because there is no substitution σ such that $y + \mathsf{s}(\mathsf{s}(z))\sigma$ reduces to $x + \mathsf{s}(0)\sigma$. Hence, there is no cycle consisting of (3) only and therefore it is sufficient if (3) is just *weakly* decreasing. In this way, the constraints resulting from this example would again be satisfied by the recursive path ordering (after applying the argument filtering mentioned in Ex. 3.39).

Note that the narrowing refinement [AG00] also serves to compute a better approximation of the dependency graph. The right-hand side of (3) is linear and it does not unify with the left-hand side of any dependency pair. Hence, we may replace (3) by its narrowings:

$$\mathsf{PLUS}(x + \mathsf{s}(0), 0 + \mathsf{s}(\mathsf{s}(z))) \to \mathsf{PLUS}(\mathsf{s}(\mathsf{s}(z)), x + \mathsf{s}(0)) \tag{4}$$
$$\mathsf{PLUS}(x + \mathsf{s}(0), \mathsf{s}(y) + \mathsf{s}(\mathsf{s}(z))) \to \mathsf{PLUS}(\mathsf{s}(y + \mathsf{s}(z)), x + \mathsf{s}(0)) \tag{5}$$
$$\mathsf{PLUS}(0 + \mathsf{s}(0), y + \mathsf{s}(\mathsf{s}(z))) \to \mathsf{PLUS}(y + \mathsf{s}(\mathsf{s}(z)), \mathsf{s}(0)) \tag{6}$$
$$\mathsf{PLUS}(\mathsf{s}(x) + \mathsf{s}(0), y + \mathsf{s}(\mathsf{s}(z))) \to \mathsf{PLUS}(y + \mathsf{s}(\mathsf{s}(z)), \mathsf{s}(x + 0)). \tag{7}$$

Now it is immediately clear that (4) - (7) are not on a cycle of the estimated dependency graph, because application of $\textsc{ren}$ and $\textsc{cap}$ to their right-hand sides yields terms of the form $\mathsf{PLUS}(\mathsf{s}(\ldots), \ldots)$ or $\mathsf{PLUS}(\ldots, \mathsf{s}(\ldots))$ which do not unify with $\mathsf{PLUS}(\ldots + \ldots, \ldots + \ldots)$.

3.41 Factorial

The following non-simply terminating TRS for computing the factorial of a natural number (cf. [Ste95a,Zan95])

[1] This inequality is not satisfied by any path ordering (that can be generated automatically), because neither a lexicographic comparison nor a comparison as multisets makes $(x + \mathsf{s}(0), y + \mathsf{s}(\mathsf{s}(z)))$ greater than $(y + \mathsf{s}(\mathsf{s}(z)), x + \mathsf{s}(0))$. When using polynomial orderings, PLUS is mapped to some polynomial p. Then we either have $\lim_{y \to \infty}(\, p(y, x) - p(x, y)\,) = \infty$ or $\lim_{y \to \infty}(\, p(y, x) - p(x, y)\,) = -\infty$. In the first case, $\mathsf{PLUS}(y + \mathsf{s}(\mathsf{s}(z)), x + \mathsf{s}(0)) \succ \mathsf{PLUS}(x + \mathsf{s}(0), y + \mathsf{s}(\mathsf{s}(z)))$ holds for large enough y and in the second case $\mathsf{PLUS}(y + \mathsf{s}(\mathsf{s}(z)), x + \mathsf{s}(0)) \succ \mathsf{PLUS}(x + \mathsf{s}(0), y + \mathsf{s}(\mathsf{s}(z)))$ holds for large enough x.

$$
\begin{aligned}
\mathsf{p}(\mathsf{s}(x)) &\to x \\
\mathsf{fac}(\mathsf{0}) &\to \mathsf{s}(\mathsf{0}) \\
\mathsf{fac}(\mathsf{s}(x)) &\to \mathsf{s}(x) \times \mathsf{fac}(\mathsf{p}(\mathsf{s}(x)))
\end{aligned}
$$

cannot be proved terminating by the technique described in [AG97a], since there narrowing dependency pairs was not considered. By using narrowing, the dependency pair

$$\mathsf{FAC}(\mathsf{s}(x)) \to \mathsf{FAC}(\mathsf{p}(\mathsf{s}(x)))$$

is replaced by the pair

$$\mathsf{FAC}(\mathsf{s}(x)) \to \mathsf{FAC}(x)$$

resulting in inequalities which can easily be satisfied.

3.42 Binary numbers

The following non-simply terminating example is due to Geser [BL90,Ste95a].

$$
\begin{aligned}
\mathsf{half}(\mathsf{0}) &\to \mathsf{0} \\
\mathsf{half}(\mathsf{s}(\mathsf{0})) &\to \mathsf{0} \\
\mathsf{half}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{half}(x)) \\
\mathsf{lastbit}(\mathsf{0}) &\to \mathsf{0} \\
\mathsf{lastbit}(\mathsf{s}(\mathsf{0})) &\to \mathsf{s}(\mathsf{0}) \\
\mathsf{lastbit}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{lastbit}(x) \\
\mathsf{conv}(\mathsf{0}) &\to \mathsf{nil}\bullet\mathsf{0} \\
\mathsf{conv}(\mathsf{s}(x)) &\to \mathsf{conv}(\mathsf{half}(\mathsf{s}(x)))\bullet\mathsf{lastbit}(\mathsf{s}(x))
\end{aligned}
$$

Narrowing the dependency pair $\mathsf{CONV}(\mathsf{s}(x)) \to \mathsf{CONV}(\mathsf{half}(\mathsf{s}(x)))$ results in $\mathsf{CONV}(\mathsf{s}(\mathsf{0})) \to \mathsf{CONV}(\mathsf{0})$ and $\mathsf{CONV}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{CONV}(\mathsf{s}(\mathsf{half}(x)))$. After this replacement, the pairs on a cycle in the estimated dependency graph are

$$
\begin{aligned}
&\{\mathsf{HALF}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{HALF}(x)\} \\
&\{\mathsf{LASTBIT}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{LASTBIT}(x)\} \\
&\{\mathsf{CONV}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{CONV}(\mathsf{s}(\mathsf{half}(x)))\}.
\end{aligned}
$$

After applying the argument filtering $\pi(\mathsf{half}) = \pi(\bullet) = 1$, the constraints are satisfied by the recursive path ordering.

3.43 Termination by narrowing, version 1

The following TRS by Plaisted [Pla86,Ste95a]

$$
\begin{aligned}
\mathsf{f}(\mathsf{c}) &\to \mathsf{g}(\mathsf{h}(\mathsf{c})) \\
\mathsf{h}(\mathsf{g}(x)) &\to \mathsf{g}(\mathsf{h}(\mathsf{f}(x))) \\
\mathsf{k}(x, \mathsf{h}(x), \mathsf{c}) &\to \mathsf{h}(x) \\
\mathsf{k}(\mathsf{f}(x), y, x) &\to \mathsf{f}(x)
\end{aligned}
$$

can automatically be proved terminating by only replacing the dependency pair $\mathsf{H}(\mathsf{g}(x)) \to \mathsf{H}(\mathsf{f}(x))$ by its narrowing $\mathsf{H}(\mathsf{g}(\mathsf{c})) \to \mathsf{H}(\mathsf{g}(\mathsf{h}(\mathsf{c})))$ and computing the estimated dependency graph. As there is no cycle consisting of the resulting pairs, the TRS is terminating.

3.44 Termination by narrowing, version 2

To prove termination of the following TRS from Bachmair [Bac87,Ste95a]

$$\begin{aligned} \mathsf{f}(\mathsf{h}(x)) &\to \mathsf{f}(\mathsf{i}(x)) \\ \mathsf{g}(\mathsf{i}(x)) &\to \mathsf{g}(\mathsf{h}(x)) \\ \mathsf{h}(\mathsf{a}) &\to \mathsf{b} \\ \mathsf{i}(\mathsf{a}) &\to \mathsf{b} \end{aligned}$$

the dependency pairs

$$\begin{aligned} \mathsf{F}(\mathsf{h}(x)) &\to \mathsf{F}(\mathsf{i}(x)) \\ \mathsf{G}(\mathsf{i}(x)) &\to \mathsf{G}(\mathsf{h}(x)) \end{aligned}$$

are replaced by their narrowings

$$\begin{aligned} \mathsf{F}(\mathsf{h}(\mathsf{a})) &\to \mathsf{F}(\mathsf{b}) \\ \mathsf{G}(\mathsf{i}(\mathsf{a})) &\to \mathsf{G}(\mathsf{b}). \end{aligned}$$

Then termination is automatically proved by the fact that the estimated dependency graph has no cycles.

3.45 Termination by narrowing, version 3

For the following TRS we also need narrowing in order to prove its termination using a quasi-simplification ordering.

$$\begin{aligned} \mathsf{f}(\mathsf{s}(x)) &\to \mathsf{f}(x) \\ \mathsf{g}(\mathsf{0}\bullet y) &\to \mathsf{g}(y) \\ \mathsf{g}(\mathsf{s}(x)\bullet y) &\to \mathsf{s}(x) \\ \mathsf{h}(x\bullet y) &\to \mathsf{h}(\mathsf{g}(x\bullet y)) \end{aligned}$$

Narrowing the dependency pair $\mathsf{H}(x\bullet y) \to \mathsf{H}(\mathsf{g}(x\bullet y))$ results in

$$\begin{aligned} &\mathsf{H}(\mathsf{0}\bullet y) \to \mathsf{H}(\mathsf{g}(y)) \\ &\mathsf{H}(\mathsf{s}(x)\bullet y) \to \mathsf{H}(\mathsf{s}(x)). \end{aligned}$$

Now the cycles are

$$\begin{aligned} &\{\mathsf{F}(\mathsf{s}(x)) \to \mathsf{F}(x)\} \\ &\{\mathsf{G}(\mathsf{0}\bullet y) \to \mathsf{G}(y)\} \\ &\{\mathsf{H}(\mathsf{0}\bullet y) \to \mathsf{H}(\mathsf{g}(y))\}. \end{aligned}$$

After applying the argument filtering $\pi(\mathsf{h}) = [\,]$, the resulting constraints are satisfied by the recursive path ordering.

3.46 A non-totally terminating TRS

The following example is from Steinbach [Ste95a].

$$\begin{aligned} \mathsf{f}(x,x) &\to \mathsf{f}(\mathsf{a},\mathsf{b}) \\ \mathsf{b} &\to \mathsf{c} \end{aligned}$$

This TRS is not totally terminating and without using narrowing, the inequalities generated by our technique are not satisfied by any total well-founded weakly monotonic quasi-ordering. However, after applying one narrowing step to $\mathsf{F}(x,x) \to \mathsf{F}(\mathsf{a},\mathsf{b})$, the pair $\mathsf{F}(x,x) \to \mathsf{F}(\mathsf{a},\mathsf{c})$ is obtained, whose right-hand side is not unifiable with $\mathsf{F}(x,x)$. Hence, there is no cycle in the dependency graph. Thus, the TRS is terminating.

3.47 An overlapping system

The following TRS is a leading example of [AG98] and [GAO01] which is not simply terminating.

$$\begin{aligned} \mathsf{f}(x,\mathsf{c}(y)) &\to \mathsf{f}(x,\mathsf{s}(\mathsf{f}(y,y))) \\ \mathsf{f}(\mathsf{s}(x),y) &\to \mathsf{f}(x,\mathsf{s}(\mathsf{c}(y))) \end{aligned}$$

The cycles in the estimated dependency graph are:

$$\begin{array}{l} \{\mathsf{F}(x,\mathsf{c}(y)) \to \mathsf{F}(y,y)\} \\ \{\mathsf{F}(\mathsf{s}(x),y) \to \mathsf{F}(x,\mathsf{s}(\mathsf{c}(y)))\} \end{array}$$

and the two sets of generated inequalities are:

$$\begin{aligned} \pi_1(\mathsf{f}(x,\mathsf{c}(y))) &\succsim_1 \pi_1(\mathsf{f}(x,\mathsf{s}(\mathsf{f}(y,y)))) \\ \pi_1(\mathsf{f}(\mathsf{s}(x),y)) &\succsim_1 \pi_1(\mathsf{f}(x,\mathsf{s}(\mathsf{c}(y)))) \\ \pi_1(\mathsf{F}(x,\mathsf{c}(y))) &\succ_1 \pi_1(\mathsf{F}(y,y)) \end{aligned}$$

$$\begin{aligned} \pi_2(\mathsf{f}(x,\mathsf{c}(y))) &\succsim_2 \pi_2(\mathsf{f}(x,\mathsf{s}(\mathsf{f}(y,y)))) \\ \pi_2(\mathsf{f}(\mathsf{s}(x),y)) &\succsim_2 \pi_2(\mathsf{f}(x,\mathsf{s}(\mathsf{c}(y)))) \\ \pi_2(\mathsf{F}(\mathsf{s}(x),y)) &\succ_2 \pi_2(\mathsf{F}(x,\mathsf{s}(\mathsf{c}(y)))). \end{aligned}$$

By choosing the argument filterings $\pi_1(\mathsf{f}) = 1$, $\pi_1(\mathsf{F}) = 2$ and $\pi_2(\mathsf{f}) = \pi_2(\mathsf{F}) = 1$ the inequalities are solved by RPO and the TRS is proved to be DP simply terminating.

Note that the constraints obtained without using our modularity results would include $\pi(\mathsf{F}(x,\mathsf{c}(y))) \succ \pi(\mathsf{F}(y,y))$ and $\pi(\mathsf{F}(\mathsf{s}(x),y)) \succ \pi(\mathsf{F}(x,\mathsf{s}(\mathsf{c}(y))))$. In this example π cannot eliminate the arguments of s or c. Then no simplification ordering satisfies the above constraints, as they imply

$$\pi(\mathsf{F}(x,\mathsf{c}(\mathsf{s}(x)))) \succ \pi(\mathsf{F}(\mathsf{s}(x),\mathsf{s}(x))) \succ \pi(\mathsf{F}(x,\mathsf{s}(\mathsf{c}(\mathsf{s}(x))))).$$

Note also that the system is overlapping (and not locally confluent). Hence, we cannot prove termination by verifying innermost termination, but we really have to use Thm. 5 for the termination proof instead.

3.48 Another overlapping system

The following system is an overlapping TRS which is inspired by Ex. 4.35 for renaming in the Lambda Calculus.

$$\begin{aligned}
\mathsf{f}(\mathsf{0}) &\to \mathsf{true}\\
\mathsf{f}(\mathsf{1}) &\to \mathsf{false}\\
\mathsf{f}(\mathsf{s}(x)) &\to \mathsf{f}(x)\\
\mathsf{if}(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(x)\\
\mathsf{if}(\mathsf{false}, \mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(y)\\
\mathsf{g}(x, \mathsf{c}(y)) &\to \mathsf{c}(\mathsf{g}(x, y))\\
\mathsf{g}(x, \mathsf{c}(y)) &\to \mathsf{g}(x, \mathsf{if}(\mathsf{f}(x), \mathsf{c}(\mathsf{g}(\mathsf{s}(x), y)), \mathsf{c}(y)))
\end{aligned}$$

The system is not simply terminating as the last rule is self-embedding. As it is overlapping (and not locally confluent), here it is not sufficient to prove innermost termination only. Without modularity, the automated termination proof would fail, because the third argument of if and the argument of c cannot be eliminated. But no quasi-simplification ordering satisfies $\mathsf{G}(x, \mathsf{c}(y)) \succ \mathsf{G}(x, \mathsf{if}(\ldots, \ldots, \mathsf{c}(y)))$.

There is just one cycle in the estimated dependency graph which contains an F-dependency pair, viz. $\{\mathsf{F}(\mathsf{s}(x)) \to \mathsf{F}(x)\}$. Absence of infinite chains of this dependency pair can be proved by RPO, if we use the argument filtering $\pi(\mathsf{c}) = \pi(\mathsf{g}) = [\,]$. Then all rules are weakly decreasing (using the precedence $\mathsf{f} > \mathsf{true}$, $\mathsf{f} > \mathsf{false}$, $\mathsf{g} > \mathsf{c}$). For all other cycles one can eliminate the arguments of s, f, and if before using RPO.

3.49 Maximal cycles

One could think of formulating Thm. 5 (and also the other modularity theorems) in an alternative way by just considering *maximal* cycles for modularity. Here, a cycle $\mathcal{P}$ is called *maximal* if there is no proper superset of $\mathcal{P}$ which is also a cycle. As an example consider the following system:

$$\begin{aligned}
\mathsf{f}(\mathsf{c}(\mathsf{s}(x), y)) &\to \mathsf{f}(\mathsf{c}(x, \mathsf{s}(y)))\\
\mathsf{f}(\mathsf{c}(\mathsf{s}(x), \mathsf{s}(y))) &\to \mathsf{g}(\mathsf{c}(x, y))\\
\mathsf{g}(\mathsf{c}(x, \mathsf{s}(y))) &\to \mathsf{g}(\mathsf{c}(\mathsf{s}(x), y))\\
\mathsf{g}(\mathsf{c}(\mathsf{s}(x), \mathsf{s}(y))) &\to \mathsf{f}(\mathsf{c}(x, y))
\end{aligned}$$

We obtain the following dependency pairs:

$$\begin{aligned}
\mathsf{F}(\mathsf{c}(\mathsf{s}(x), y) &\to \mathsf{F}(\mathsf{c}(x, \mathsf{s}(y))) \qquad (8)\\
\mathsf{F}(\mathsf{c}(\mathsf{s}(x), \mathsf{s}(y))) &\to \mathsf{G}(\mathsf{c}(x, y)) \qquad (9)\\
\mathsf{G}(\mathsf{c}(x, \mathsf{s}(y))) &\to \mathsf{G}(\mathsf{c}(\mathsf{s}(x), y)) \qquad (10)\\
\mathsf{G}(\mathsf{c}(\mathsf{s}(x), \mathsf{s}(y))) &\to \mathsf{F}(\mathsf{c}(x, y)) \qquad (11)
\end{aligned}$$

The cycles of the estimated dependency graph are $\{(8)\}, \{(10)\}, \{(9),(11)\}, \{(8),(9),(11)\}, \{(9),(10),(11)\}$, and $\{(8),(9),(10),(11)\}$. So the only maximal cycle in this example is $\{(8),(9),(10),(11)\}$. A simple way to compute the set of all maximal cycles is to eliminate all edges and all dependency pairs in the estimated dependency graph which are not part of any cycle. Then the remaining unconnected graphs correspond to the maximal cycles.

Now a modification of Thm. 2 would be that a TRS is terminating iff for each *maximal* cycle $\mathcal{P}$ there exists no infinite $\mathcal{R}$-chain of dependency pairs from $\mathcal{P}$. Then, for each subcycle $\mathcal{P}'$ of $\mathcal{P}$ one would have to use the same quasi-ordering $\succsim_{\mathcal{P}}$ to prove the absence of infinite chains from $\mathcal{P}'$.

However, to use the same quasi-ordering for all subcycles of the maximal cycle can be too weak. In our example, *all* dependency pairs are on the maximal cycle. However, if one would have to use the same quasi-ordering for all subcycles of this maximal cycle, then the resulting constraints would not be satisfied by any path ordering amenable to automation or by any polynomial ordering.

Due to our modularity result we can prove absence of infinite chains separately for every cycle. We use polynomial orderings where both $\mathsf{f}(x,y)$ and $\mathsf{g}(x,y)$ are mapped to 0 and $\mathsf{s}(x)$ is mapped to $x+1$. For the cycle $\{(8)\}$, $\mathsf{c}(x,y)$ is mapped to x, whereas for the cycle $\{(10)\}$ we map $\mathsf{c}(x,y)$ to y. For the other cycles, $\mathsf{c}(x,y)$ is mapped to $x+y$. Then these polynomial orderings can be used to prove absence of infinite chains for all cycles.

3.50 DP quasi-simple, but not DP simple, version 1

The following is an example of a TRS that is DP quasi-simply terminating, but not DP simply terminating (cf. [GAO01]).

$$\begin{aligned}
\mathsf{f}(\mathsf{f}(x)) &\to \mathsf{f}(\mathsf{c}(\mathsf{f}(x)))\\
\mathsf{f}(\mathsf{f}(x)) &\to \mathsf{f}(\mathsf{d}(\mathsf{f}(x)))\\
\mathsf{g}(\mathsf{c}(x)) &\to x\\
\mathsf{g}(\mathsf{d}(x)) &\to x\\
\mathsf{g}(\mathsf{c}(\mathsf{0})) &\to \mathsf{g}(\mathsf{d}(\mathsf{1}))\\
\mathsf{g}(\mathsf{c}(\mathsf{1})) &\to \mathsf{g}(\mathsf{d}(\mathsf{0}))
\end{aligned}$$

The only cycle in the estimated dependency graph is

$$\{\mathsf{F}(\mathsf{f}(x)) \to \mathsf{F}(x)\}.$$

In order to show DP quasi-simple termination, we choose the argument filtering $\pi(\mathsf{c}) = \pi(\mathsf{d}) = 1$ and use RPO with $\mathsf{0}$ and $\mathsf{1}$ equal in the precedence. However, the TRS is not DP simply terminating, because due to the first four rules, the argument filtering must reduce $\mathsf{c}(x)$ and $\mathsf{d}(x)$ to their arguments. But then $\mathsf{g}(\mathsf{0}) \succeq \mathsf{g}(\mathsf{1})$ and $\mathsf{g}(\mathsf{1}) \succeq \mathsf{g}(\mathsf{0})$ lead to a contradiction.

3.51 DP quasi-simple, but not DP simple, version 2

The definition of argument filtering could be modified by not only eliminating arguments but by also identifying different function symbols. This would change the notion of DP simple termination, but DP simple termination and DP quasi-simple termination would still not coincide. This is demonstrated by the following example [GAO01].

$$\begin{aligned}
\mathsf{f}(\mathsf{f}(x)) &\to \mathsf{f}(\mathsf{c}(\mathsf{f}(x)))\\
\mathsf{f}(\mathsf{f}(x)) &\to \mathsf{f}(\mathsf{d}(\mathsf{f}(x)))\\
\mathsf{g}(\mathsf{c}(x)) &\to x\\
\mathsf{g}(\mathsf{d}(x)) &\to x\\
\mathsf{g}(\mathsf{c}(\mathsf{h}(\mathsf{0}))) &\to \mathsf{g}(\mathsf{d}(\mathsf{1}))\\
\mathsf{g}(\mathsf{c}(\mathsf{1})) &\to \mathsf{g}(\mathsf{d}(\mathsf{h}(\mathsf{0})))\\
\mathsf{g}(\mathsf{h}(x)) &\to \mathsf{g}(x).
\end{aligned}$$

The dependency graph of this TRS has two cycles:

$$\begin{array}{l}
\{\mathsf{F}(\mathsf{f}(x)) \to \mathsf{F}(x)\}\\
\{\mathsf{G}(\mathsf{h}(x)) \to \mathsf{G}(x)\}.
\end{array}$$

For the first cycle we use the argument filtering $\pi(\mathsf{c}) = \pi(\mathsf{d}) = \pi(\mathsf{h}) = 1$ and RPO with $\mathsf{0}$ and $\mathsf{1}$ equal in the precedence. For the second cycle we cannot choose $\pi(\mathsf{h}) = 1$. Without any filtering on arguments, but with a polynomial interpretation that maps $\mathsf{0}$ to 0, $\mathsf{1}$ to 1, $\mathsf{h}(x)$ to $x+1$, and all other symbols to the identity, the inequalities are solved.

However, even with the new definition of argument filtering, the system is still not DP simply terminating. The reason is that again, the argument filtering π must map c and d to their arguments. Then the third and fourth g-rule imply $\pi(\mathsf{g}(\mathsf{h}(\mathsf{0}))) = \pi(\mathsf{g}(\mathsf{1}))$. Since $\pi(\mathsf{g}) \neq []$ due to the first g-rule, this implies $\pi(\mathsf{h}(\mathsf{0})) = \pi(\mathsf{1})$. Due to the dependency pair $\mathsf{G}(\mathsf{h}(x)) \to \mathsf{G}(x)$, π may neither map h to its argument nor to any constant like $\mathsf{1}$. Hence, even with this alternative definition of argument filtering, these constraints are not satisfiable.

3.52 A TRS that is not left-linear, version 2

The following TRS occurs in [GAO01].

$$\begin{aligned}
\mathsf{f}(\mathsf{0},\mathsf{1},x) &\to \mathsf{f}(\mathsf{s}(x),x,x)\\
\mathsf{f}(x,y,\mathsf{s}(z)) &\to \mathsf{s}(\mathsf{f}(\mathsf{0},\mathsf{1},z)).
\end{aligned}$$

The cycles in the estimated dependency graph are

$$\begin{aligned}
\{\mathsf{F}(\mathsf{0},\mathsf{1},x) &\to \mathsf{F}(\mathsf{s}(x),x,x), && (12)\\
\mathsf{F}(x,y,\mathsf{s}(z)) &\to \mathsf{F}(\mathsf{0},\mathsf{1},z)\} && (13)\\
\{\mathsf{F}(x,y,\mathsf{s}(z)) &\to \mathsf{F}(\mathsf{0},\mathsf{1},z)\}. && (14)
\end{aligned}$$

Therefore, it suffices to find an argument filtering such that the following inequalities are satisfied:

$$\begin{aligned}
\pi(\mathsf{f}(\mathsf{0},\mathsf{1},x)) &\succsim \pi(\mathsf{f}(\mathsf{s}(x),x,x))\\
\pi(\mathsf{f}(x,y,\mathsf{s}(z))) &\succsim \pi(\mathsf{s}(\mathsf{f}(\mathsf{0},\mathsf{1},z)))\\
\pi(\mathsf{F}(\mathsf{0},\mathsf{1},x)) &\succsim \pi(\mathsf{F}(\mathsf{s}(x),x,x))\\
\pi(\mathsf{F}(x,y,\mathsf{s}(z))) &\succ \pi(\mathsf{F}(\mathsf{0},\mathsf{1},z)).
\end{aligned}$$

A suitable argument filtering is $\pi(\mathsf{f}) = \pi(\mathsf{F}) = 3$. By using RPO, DP simple termination of the TRS is proved.

3.53 Disjoint systems, DP quasi-simple termination

By Theorem 6 we may conclude DP quasi-simple termination of several combinations of the above TRSs. To mention only a few:

– the combination of the TRSs in Ex. 3.1, 3.12, and 3.14 is DP quasi-simply terminating,
– the TRS $\mathsf{g}(x,y) \to x$, $\mathsf{g}(x,y) \to y$ in combination with either Ex. 3.29 or Ex. 3.52 is DP quasi-simply terminating.

3.54 Disjoint systems, $\mathcal{G}$-restricted DP simple termination

If $\mathcal{G}$ is chosen to be the empty set, $\emptyset$-restricted DP simple termination of the TRS consisting of Ex. 3.25 and Ex. 3.29 follows immediately from Theorem 7 (where unary f and ternary f are different symbols).

3.55 Shared constructors, $\mathcal{G}$-restricted DP simple termination 1

By Theorem 7 we may conclude $\mathcal{G}$-restricted DP simple termination of the combination of the division example (Ex. 3.1) and the quicksort example (Ex. 3.11) where $\mathcal{G} = \{\mathsf{0}, \mathsf{s}\}$.

3.56 Shared constructors, $\mathcal{G}$-restricted DP simple termination 2

The TRS

$$\mathsf{g}(\mathsf{c}(x,\mathsf{s}(y))) \to \mathsf{g}(\mathsf{c}(\mathsf{s}(x),y))$$

is simply terminating, as can for example be shown by LPO comparing subterms right-to-left. The TRS

$$\begin{aligned}
\mathsf{f}(\mathsf{c}(\mathsf{s}(x),y)) &\to \mathsf{f}(\mathsf{c}(x,\mathsf{s}(y)))\\
\mathsf{f}(\mathsf{f}(x)) &\to \mathsf{f}(\mathsf{d}(\mathsf{f}(x)))\\
\mathsf{f}(x) &\to x
\end{aligned}$$

also has c and s as constructors. DP simple termination of the TRS can be shown by an argument filtering $\pi(\mathsf{d}) = [\,]$ and LPO comparing subterms left-to-right. A simple check confirms that both systems are $\{\mathsf{s}, \mathsf{c}\}$-restricted DP simply terminating. Hence, the combination is also $\{\mathsf{s}, \mathsf{c}\}$-restricted DP simply terminating.

DP simple termination of both $\mathcal{R}_1$ and $\mathcal{R}_2$ can be proved with a standard technique like LPO, whereas such standard orderings fail if one wants to prove DP simple termination of their union directly. The reason is that the constraints for the cycle $\{\mathsf{G}(\mathsf{c}(x, \mathsf{s}(y))) \to \mathsf{G}(\mathsf{c}(\mathsf{s}(x), y))\}$ are not satisfied by LPO (nor by RPO nor by any polynomial ordering). Thus, there are indeed TRSs where termination of the subsystems can be shown with dependency pairs and LPO, but (without our modularity result) termination of their union cannot be proved with dependency pairs and LPO.

3.57 Composable systems, $\mathcal{G}$-restricted DP simple termination

The TRSs of Ex. 3.4 and of Ex. 3.17 are both $\{\mathsf{0}, \mathsf{s}, +\}$-restricted DP simply terminating. Note that the resulting TRSs are composable, since they both contain the same constructors $\mathsf{0}$ and s and they also share the defined symbol $+$, but both TRSs contain the same $+$-rules. As both TRSs are $\{\mathsf{0}, \mathsf{s}, +\}$-restricted DP simply terminating, Theorem 7 allows us to conclude $\{\mathsf{0}, \mathsf{s}, +\}$-restricted DP simple termination of the combined system.

4 Examples for innermost termination

This section contains a collection of examples to demonstrate the use of the innermost termination technique presented in Sect. 2.2. The examples 4.1 – 4.21 are term rewrite systems that are innermost terminating, but not terminating. The remainder of the examples (4.22 – 4.37) are non-overlapping term rewrite systems for which innermost termination suffices to guarantee termination. Note that for the examples 4.6 – 4.9, 4.14 – 4.21, and 4.25 – 4.37 we used refinements which were not included in the method of [AG97b]. In particular, the examples 4.19 – 4.21 and 4.32 – 4.37 are TRSs, where an innermost termination proof without modularity is impossible with quasi-simplification orderings (or, in some examples, at least with the standard path orderings amenable to automation), whereas with our modularity results innermost termination can easily be verified automatically.

4.1 Toyama example

A famous example of a TRS that is innermost terminating, but not terminating, is the following system by Toyama [Toy87].

$$\mathsf{f}(\mathsf{0}, \mathsf{1}, x) \to \mathsf{f}(x, x, x)$$
$$\mathsf{g}(x, y) \to x$$
$$\mathsf{g}(x, y) \to y$$

This TRS has only one dependency pair, viz. $\mathsf{F}(\mathsf{0},\mathsf{1},x) \to \mathsf{F}(x,x,x)$. This dependency pair does not occur on a cycle in the innermost dependency graph, since $\mathsf{F}(x_1,x_1,x_1)$ does not unify with $\mathsf{F}(\mathsf{0},\mathsf{1},x_2)$. Thus, no inequalities are generated and therefore the TRS is innermost terminating.

4.2 Variations on the Toyama example, version 1

The following example is a non-terminating TRS

$$\begin{aligned} \mathsf{f}(\mathsf{g}(x),\mathsf{s}(\mathsf{0}),y) &\to \mathsf{f}(y,y,\mathsf{g}(x)) \\ \mathsf{g}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{g}(x)) \\ \mathsf{g}(\mathsf{0}) &\to \mathsf{0} \end{aligned}$$

with only one dependency pair on a cycle in the innermost dependency graph, viz. $\mathsf{G}(\mathsf{s}(x)) \to \mathsf{G}(x)$. Since no defined symbols occur in $\mathsf{G}(x)$, there are no usable rules. Therefore, the only constraint on the ordering is given by

$$\mathsf{G}(\mathsf{s}(x)) \succ \mathsf{G}(x)$$

which is easily satisfied by the recursive path ordering. Hence, the TRS is innermost terminating.

4.3 Variations on the Toyama example, version 2

Similar to the preceding example, the following modification of the Toyama example

$$\begin{aligned} \mathsf{f}(\mathsf{g}(x,y),x,z) &\to \mathsf{f}(z,z,z) \\ \mathsf{g}(x,y) &\to x \\ \mathsf{g}(x,y) &\to y \end{aligned}$$

is not a constructor system, since the subterm $\mathsf{g}(x,y)$ occurs in the left-hand side of the first rule. Again the innermost dependency graph does not contain any cycles and hence, this TRS is innermost terminating. This TRS is, however, not terminating.

4.4 Variations on the Toyama example, version 3

The non-terminating TRS

$$\begin{aligned} \mathsf{f}(\mathsf{g}(x),x,y) &\to \mathsf{f}(y,y,\mathsf{g}(y)) \\ \mathsf{g}(\mathsf{g}(x)) &\to \mathsf{g}(x) \end{aligned}$$

is no constructor system either. The pair $\mathsf{F}(\mathsf{g}(x),x,y) \to \mathsf{F}(y,y,\mathsf{g}(y))$ cannot occur in an infinite innermost chain, since $\textsc{cap}_{\mathsf{F}(\mathsf{g}(x_1),x_1,y_1)}(\mathsf{F}(y_1,y_1,\mathsf{g}(y_1)))$ does not unify with $\mathsf{F}(\mathsf{g}(x_2),x_2,y_2)$. The dependency pair $\mathsf{G}(\mathsf{g}(x)) \to \mathsf{G}(x)$ cannot occur in an infinite innermost chain either, since by unifying the right projection of this dependency pair with a renaming of it, the left projection is instantiated in such a way that it is not a normal form. Hence, there are no cycles in the innermost dependency graph and therefore the TRS is innermost terminating.

4.5 Redex in left-hand side

The following system

$$\mathsf{f}(\mathsf{0}) \to \mathsf{f}(\mathsf{0})$$
$$\mathsf{0} \to \mathsf{1}$$

is innermost terminating, because there is no cycle in the innermost dependency graph. The reason is that the left-hand side $\mathsf{F}(\mathsf{0})$ of the (only) dependency pair is not a normal form.

4.6 Narrowing required, version 1

In the following, again non-terminating, variant of the Toyama example

$$\begin{aligned} \mathsf{f}(\mathsf{0}, \mathsf{1}, x) &\to \mathsf{f}(\mathsf{g}(x, x), x, x) \\ \mathsf{g}(x, y) &\to x \\ \mathsf{g}(x, y) &\to y \end{aligned}$$

one narrowing step is needed to determine that there are no cycles in the innermost dependency graph (because $\mathsf{F}(\mathsf{0}, \mathsf{1}, x) \to \mathsf{F}(\mathsf{g}(x, x), x, x)$ narrows to $\mathsf{F}(\mathsf{0}, \mathsf{1}, x)$ $\to \mathsf{F}(x, x, x)$). Thus, this TRS is also innermost terminating.

4.7 Narrowing required, version 2

The following example can be solved in a similar way:

$$\begin{aligned} \mathsf{f}(\mathsf{s}(x)) &\to \mathsf{f}(\mathsf{g}(x, x)) \\ \mathsf{g}(\mathsf{0}, \mathsf{1}) &\to \mathsf{s}(\mathsf{0}) \\ \mathsf{0} &\to \mathsf{1}. \end{aligned}$$

The dependency pair $\mathsf{F}(\mathsf{s}(x)) \to \mathsf{F}(\mathsf{g}(x, x))$ may be deleted as it cannot be narrowed. Hence, there is no dependency pair left and therefore, innermost termination is proved.

4.8 Narrowing required, version 3

Consider the following TRS

$$\begin{aligned} x + \mathsf{0} &\to x \\ x + \mathsf{s}(y) &\to \mathsf{s}(x + y) \\ \mathsf{f}(\mathsf{0}, \mathsf{s}(\mathsf{0}), x) &\to \mathsf{f}(x, x + x, x) \\ \mathsf{g}(x, y) &\to x \\ \mathsf{g}(x, y) &\to y \end{aligned}$$

which is not terminating as can be seen by the infinite reduction

$$\begin{aligned}
\mathsf{f}(0,\mathsf{s}(0),\mathsf{g}(0,\mathsf{s}(0))) &\to \mathsf{f}(\mathsf{g}(0,\mathsf{s}(0)),\mathsf{g}(0,\mathsf{s}(0))+\mathsf{g}(0,\mathsf{s}(0)),\mathsf{g}(0,\mathsf{s}(0))) \\
&\to \mathsf{f}(0,\mathsf{g}(0,\mathsf{s}(0))+\mathsf{g}(0,\mathsf{s}(0)),\mathsf{g}(0,\mathsf{s}(0))) \\
&\to \mathsf{f}(0,\mathsf{s}(0)+\mathsf{g}(0,\mathsf{s}(0)),\mathsf{g}(0,\mathsf{s}(0))) \\
&\to \mathsf{f}(0,\mathsf{s}(0)+0,\mathsf{g}(0,\mathsf{s}(0))) \\
&\to \mathsf{f}(0,\mathsf{s}(0),\mathsf{g}(0,\mathsf{s}(0))) \\
&\to \ldots
\end{aligned}$$

Innermost termination of this TRS can be proved if the dependency pair $\mathsf{F}(0,\mathsf{s}(0),x) \to \mathsf{F}(x,x+x,x)$ is replaced by its narrowings

$$\begin{aligned}
&\mathsf{F}(0,\mathsf{s}(0),0) \to \mathsf{F}(0,0,0) \\
&\mathsf{F}(0,\mathsf{s}(0),\mathsf{s}(y)) \to \mathsf{F}(\mathsf{s}(y),\mathsf{s}(\mathsf{s}(y)+y),\mathsf{s}(y)).
\end{aligned}$$

Now our approximation determines that these dependency pairs are not on cycles in the innermost dependency graph. Therefore, the only inequality generated for this TRS is

$$\mathsf{PLUS}(x,\mathsf{s}(y)) \succ \mathsf{PLUS}(x,y)$$

which is satisfied by the recursive path ordering. Hence, this TRS is proved innermost terminating.

4.9 Narrowing required, version 4

The following modification of the above TRS

$$\begin{aligned}
x+0 &\to x \\
x+\mathsf{s}(y) &\to \mathsf{s}(x+y) \\
\mathsf{double}(x) &\to x+x \\
\mathsf{f}(0,\mathsf{s}(0),x) &\to \mathsf{f}(x,\mathsf{double}(x),x) \\
\mathsf{g}(x,y) &\to x \\
\mathsf{g}(x,y) &\to y
\end{aligned}$$

is also non-terminating. Similar to the example above, we now need two narrowing steps to derive that the narrowings of the dependency pair

$$\mathsf{F}(0,\mathsf{s}(0),x) \to \mathsf{F}(x,\mathsf{double}(x),x)$$

do not occur on cycles in the innermost dependency graph. The generated inequality is therefore the same as for the above example, which is satisfied by the recursive path ordering. Hence, this TRS is proved innermost terminating.

4.10 Non-normal most general unifier

The following TRS

$$\mathsf{f}(x, \mathsf{g}(x)) \to \mathsf{f}(1, \mathsf{g}(x))$$
$$\mathsf{g}(1) \to \mathsf{g}(0)$$

is obviously not terminating as $\mathsf{f}(1, \mathsf{g}(1))$ can be reduced to itself. The dependency pair

$$\mathsf{F}(x, \mathsf{g}(x)) \to \mathsf{F}(1, \mathsf{g}(x))$$

does not occur on a cycle in the innermost dependency graph, because

$$\textsc{cap}_{\mathsf{F}(x_1, \mathsf{g}(x_1))}(\mathsf{F}(1, \mathsf{g}(x_1))) = \mathsf{F}(1, \mathsf{g}(x_1))$$

and the most general unifier of $\mathsf{F}(1, \mathsf{g}(x_1))$ and $\mathsf{F}(x_2, \mathsf{g}(x_2))$ replaces x_1 and x_2 by 1. Hence, the instantiation of the left projection is not a normal form. Obviously, the other dependency pairs $\mathsf{F}(x, \mathsf{g}(x)) \to \mathsf{G}(x)$ and $\mathsf{G}(1) \to \mathsf{G}(0)$ cannot occur on cycles either. Thus, there are no cycles in the innermost dependency graph. Hence, the TRS is innermost terminating.

4.11 Innermost chains of arbitrary finite length

The following non-terminating TRS has an innermost chain of any finite length, but it has no infinite innermost chain, hence it is innermost terminating.

$$\mathsf{h}(x, z) \to \mathsf{f}(x, \mathsf{s}(x), z)$$
$$\mathsf{f}(x, y, \mathsf{g}(x, y)) \to \mathsf{h}(0, \mathsf{g}(x, y))$$
$$\mathsf{g}(0, y) \to 0$$
$$\mathsf{g}(x, \mathsf{s}(y)) \to \mathsf{g}(x, y)$$

An infinite reduction is given by

$$\mathsf{h}(0, \mathsf{g}(0, \mathsf{s}(0))) \to \mathsf{f}(0, \mathsf{s}(0), \mathsf{g}(0, \mathsf{s}(0))) \to \mathsf{h}(0, \mathsf{g}(0, \mathsf{s}(0))) \to \ldots$$

So the TRS is not terminating.

The inequality resulting from the dependency pair on the only cycle in the innermost dependency graph is

$$\mathsf{G}(x, \mathsf{s}(y)) \succ \mathsf{G}(x, y).$$

(The reason is that the most general unifier of $\textsc{cap}_{\mathsf{H}(x_1, z_1)}(\mathsf{F}(x_1, \mathsf{s}(x_1), z_1))$ and $\mathsf{F}(x_2, y_2, \mathsf{g}(x_2, y_2))$ does not instantiate the latter term to a normal form.)

There are no usable rules. Thus, innermost termination is easily proved by the recursive path ordering.

4.12 Negative coefficients

The following non-terminating TRS has two dependency pairs on a cycle in the innermost dependency graph, but it has no infinite innermost chain. Hence, it is innermost terminating.

$$\begin{aligned}
\mathsf{h}(\mathsf{0}, x) &\to \mathsf{f}(\mathsf{0}, x, x) \\
\mathsf{f}(\mathsf{0}, \mathsf{1}, x) &\to \mathsf{h}(x, x) \\
\mathsf{g}(x, y) &\to x \\
\mathsf{g}(x, y) &\to y
\end{aligned}$$

An infinite reduction is given by

$$\begin{aligned}
\mathsf{f}(\mathsf{0}, \mathsf{1}, \mathsf{g}(\mathsf{0}, \mathsf{1})) &\to \mathsf{h}(\mathsf{g}(\mathsf{0}, \mathsf{1}), \mathsf{g}(\mathsf{0}, \mathsf{1})) \\
&\to \mathsf{h}(\mathsf{0}, \mathsf{g}(\mathsf{0}, \mathsf{1})) \\
&\to \mathsf{f}(\mathsf{0}, \mathsf{g}(\mathsf{0}, \mathsf{1}), \mathsf{g}(\mathsf{0}, \mathsf{1})) \\
&\to \mathsf{f}(\mathsf{0}, \mathsf{1}, \mathsf{g}(\mathsf{0}, \mathsf{1})) \;\to\; \ldots
\end{aligned}$$

The inequalities resulting from the dependency pairs on a cycle in the innermost dependency graph are

$$\begin{aligned}
\mathsf{H}(\mathsf{0}, x) &\succsim \mathsf{F}(\mathsf{0}, x, x) \\
\mathsf{F}(\mathsf{0}, \mathsf{1}, x) &\succ \mathsf{H}(x, x)
\end{aligned}$$

and there are no usable rules. These inequalities are satisfied by the polynomial interpretation where $\mathsf{0}$ and $\mathsf{1}$ are interpreted as usual and where $\mathsf{H}(x, y)$ and $\mathsf{F}(x, y, z)$ are both mapped to $(x - y)^2$.

Note that the inequalities obtained in this example are not satisfied by any weakly monotonic total well-founded quasi-ordering. For that reason a polynomial ordering with negative coefficients has been used. In innermost termination proofs this is possible if the quasi-ordering is weakly monotonic on all symbols apart from the tuple symbols and if it satisfies the condition

$$x_1 \succsim y_1 \wedge \ldots \wedge x_n \succsim y_n \;\Rightarrow\; C[x_1, \ldots, x_n] \succsim C[y_1, \ldots, y_n],$$

for all dependency pairs $s \to C[f_1(\boldsymbol{u}_1), \ldots, f_n(\boldsymbol{u}_n)]$, where C is a context without defined symbols and $f_1, \ldots, f_n$ are defined symbols.

In a similar way one can also prove innermost termination of the system where the first rule has been changed to

$$\mathsf{h}(x, y) \to \mathsf{f}(x, y, x).$$

4.13 Drosten example

A confluent and innermost terminating TRS that is not terminating was given by Drosten [Dro89].

$$\begin{aligned}
\mathsf{f}(\mathsf{0},\mathsf{1},x) &\to \mathsf{f}(x,x,x)\\
\mathsf{f}(x,y,z) &\to \mathsf{2}\\
\mathsf{0} &\to \mathsf{2}\\
\mathsf{1} &\to \mathsf{2}\\
\mathsf{g}(x,x,y) &\to y\\
\mathsf{g}(x,y,y) &\to x
\end{aligned}$$

As there exists no cycle in the innermost dependency graph, the TRS is innermost terminating.

4.14 Better approximations of the innermost dependency graph, version 1

For the approximation of innermost dependency graphs we use the function $\textsc{cap}_s$ (instead of just the function $\textsc{cap}$). An example where this refinement is needed can be obtained from Ex. 4.2 by modification of the first rule.

$$\begin{aligned}
\mathsf{f}(\mathsf{g}(x),\mathsf{s}(\mathsf{0})) &\to \mathsf{f}(\mathsf{g}(x),\mathsf{g}(x))\\
\mathsf{g}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{g}(x))\\
\mathsf{g}(\mathsf{0}) &\to \mathsf{0}
\end{aligned}$$

If we would approximate the innermost dependency graph by just using $\textsc{cap}$ then in our approximation we would draw an arc from the dependency pair

$$\mathsf{F}(\mathsf{g}(x),\mathsf{s}(\mathsf{0})) \to \mathsf{F}(\mathsf{g}(x),\mathsf{g}(x))$$

to itself, because $\textsc{cap}(\mathsf{F}(\mathsf{g}(x),\mathsf{g}(x))) = \mathsf{F}(x_1,x_2)$ unifies with its left-hand side. But then we would have to demand that this dependency pair is strictly decreasing, i.e., $\mathsf{F}(\mathsf{g}(x),\mathsf{s}(\mathsf{0})) \succ \mathsf{F}(\mathsf{g}(x),\mathsf{g}(x))$. However, then the resulting constraints would imply

$$\mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})),\mathsf{s}(\mathsf{0})) \succ \mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})),\mathsf{g}(\mathsf{s}(\mathsf{0}))) \succsim \mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})),\mathsf{s}(\mathsf{g}(\mathsf{0}))) \succsim \mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})),\mathsf{s}(\mathsf{0})).$$

Hence, they would not be satisfied by any well-founded ordering closed under substitution. Therefore the approach of [AG97b] would fail with this example.

However, by the refined approximation of using $\textsc{cap}_s$ we can immediately determine that this dependency pair is not on a cycle of the innermost dependency graph. The reason is that $\textsc{cap}_{\mathsf{F}(\mathsf{g}(x_1),\mathsf{s}(\mathsf{0}))}(\mathsf{F}(\mathsf{g}(x_1),\mathsf{g}(x_1)) = \mathsf{F}(\mathsf{g}(x_1),\mathsf{g}(x_1))$ does not unify with $\mathsf{F}(\mathsf{g}(x_2),\mathsf{s}(\mathsf{0}))$. (This example could also be solved by narrowing the dependency pair. But there are also examples where the innermost termination proof using CAP_s succeeds whereas it would not succeed when using narrowing and CAP, cf. the next example, Ex. 4.15.) Now the only remaining constraint is

$$\mathsf{G}(\mathsf{s}(x)) \succ \mathsf{G}(x)$$

from the second rule of the TRS. For example, this constraint is satisfied by the recursive path ordering.

In a similar way we can also handle the following modification of Ex. 4.4:

$$\begin{aligned} \mathsf{f}(\mathsf{g}(x), x) &\to \mathsf{f}(\mathsf{g}(x), \mathsf{g}(x)) \\ \mathsf{g}(\mathsf{g}(x)) &\to \mathsf{g}(x). \end{aligned}$$

4.15 Better approximations of the innermost dependency graph, version 2

This is a variation of the Toyama example where the approximation using CAP_s is necessary to perform the innermost termination proof. In contrast to the preceding example, here narrowing the dependency pairs (and just using CAP instead of CAP_s) would not help.

$$\begin{aligned} \mathsf{f}(\mathsf{0}, \mathsf{1}, \mathsf{g}(x,y), z) &\to \mathsf{f}(\mathsf{g}(x,y), \mathsf{g}(x,y), \mathsf{g}(x,y), \mathsf{h}(x)) \\ \mathsf{g}(\mathsf{0}, \mathsf{1}) &\to \mathsf{0} \\ \mathsf{g}(\mathsf{0}, \mathsf{1}) &\to \mathsf{1} \\ \mathsf{h}(\mathsf{g}(x,y)) &\to \mathsf{h}(x) \end{aligned}$$

The dependency pair

$$\mathsf{F}(\mathsf{0}, \mathsf{1}, \mathsf{g}(x,y), z) \to \mathsf{F}(\mathsf{g}(x,y), \mathsf{g}(x,y), \mathsf{g}(x,y), \mathsf{h}(x))$$

is not on a cycle of the innermost dependency graph. This can also be determined by our approximation, because $\text{CAP}_{\mathsf{F}(\mathsf{0},\mathsf{1},\mathsf{g}(x,y),z)}(\mathsf{F}(\mathsf{g}(x,y), \mathsf{g}(x,y), \mathsf{g}(x,y), \mathsf{h}(x)))$ $= \mathsf{F}(\mathsf{g}(x,y), \mathsf{g}(x,y), \mathsf{g}(x,y), \mathsf{h}(x))$ does not unify with $\mathsf{F}(\mathsf{0}, \mathsf{1}, \ldots)$.

However, if we use just the approximation with CAP, then we would have an arc from this dependency pair to itself. Now the resulting constraints would imply

$$\mathsf{F}(\mathsf{0}, \mathsf{1}, \mathsf{g}(\mathsf{0}, \mathsf{1}), \mathsf{h}(\mathsf{0})) \succ \mathsf{F}(\mathsf{g}(\mathsf{0}, \mathsf{1}), \mathsf{g}(\mathsf{0}, \mathsf{1}), \mathsf{g}(\mathsf{0}, \mathsf{1}), \mathsf{h}(\mathsf{0})) \succsim \mathsf{F}(\mathsf{0}, \mathsf{1}, \mathsf{g}(\mathsf{0}, \mathsf{1}), \mathsf{h}(\mathsf{0})).$$

Hence, they would not be satisfied by any well-founded ordering closed under substitution.

Note that in this example narrowing the dependency pair would not help, because the narrowings would include the pair

$$\mathsf{F}(\mathsf{0}, \mathsf{1}, \mathsf{g}(\mathsf{g}(x', y'), y), z) \to \mathsf{F}(\mathsf{g}(\mathsf{g}(x', y'), y), \mathsf{g}(\mathsf{g}(x', y'), y), \mathsf{g}(\mathsf{g}(x', y'), y), \mathsf{h}(x'))$$

which would lead to the same problem. (The same statement holds for repeated applications of narrowing.) Hence, this example demonstrates that we really need the refinement of CAP_s to approximate innermost dependency graphs.

4.16 Instantiation with normal form

The following TRS

$$\mathsf{f}(\mathsf{s}(\mathsf{0}), \mathsf{g}(x)) \to \mathsf{f}(x, \mathsf{g}(x))$$
$$\mathsf{g}(\mathsf{s}(x)) \to \mathsf{g}(x)$$

is obviously not terminating as can be seen by the following infinite reduction

$$\mathsf{f}(\mathsf{s}(\mathsf{0}), \mathsf{g}(\mathsf{s}(\mathsf{0}))) \to \mathsf{f}(\mathsf{s}(\mathsf{0}), \mathsf{g}(\mathsf{s}(\mathsf{0}))) \to \dots$$

The dependency pair

$$\mathsf{F}(\mathsf{s}(\mathsf{0}), \mathsf{g}(x)) \to \mathsf{F}(x, \mathsf{g}(x))$$

is not on a cycle of the innermost dependency graph, as $\text{CAP}_{\mathsf{F}(\mathsf{s}(\mathsf{0}), \mathsf{g}(x_1))}(\mathsf{F}(x_1, \mathsf{g}(x_1)))$ and $\mathsf{F}(\mathsf{s}(\mathsf{0}), \mathsf{g}(x_2))$ unify using a most general unifier that instantiates $\mathsf{F}(\mathsf{s}(\mathsf{0}), \mathsf{g}(x_2))$ in such a way that it is not a normal form. (However, this would not have been determined by the approximation of innermost dependency graphs as presented in [AG97b].) The only dependency pair that occurs on a cycle in the innermost dependency graph is $\mathsf{G}(\mathsf{s}(x)) \to \mathsf{G}(x)$, resulting in the inequality

$$\mathsf{G}(\mathsf{s}(x)) \succ \mathsf{G}(x)$$

which is easily satisfied by the recursive path ordering.

4.17 Narrowing of pairs where right-hand sides unify with left-hand sides

In the following example we have to narrow a pair whose right-hand side unifies with a left-hand side of a dependency pair. When proving innermost termination, we may indeed perform this narrowing as long as the mgu does not instantiate the left-hand sides of the dependency pairs under consideration to normal forms.

$$\mathsf{f}(\mathsf{g}(x), \mathsf{s}(\mathsf{0}), y) \to \mathsf{f}(\mathsf{g}(\mathsf{s}(\mathsf{0})), y, \mathsf{g}(x))$$
$$\mathsf{g}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{g}(x))$$
$$\mathsf{g}(\mathsf{0}) \to \mathsf{0}$$

The dependency pair

$$\mathsf{F}(\mathsf{g}(x), \mathsf{s}(\mathsf{0}), y) \to \mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})), y, \mathsf{g}(x))$$

does not form a cycle in the innermost dependency graph, because an instantiation of its right-hand side can only reduce to an instantiation of its left-hand side where x is instantiated by $\mathsf{s}(\mathsf{0})$. But then this instantiated left-hand side would contain the redex $\mathsf{g}(\mathsf{s}(\mathsf{0}))$.

However, in our *approximation* there would be an arc from this dependency pair to itself, because $\text{CAP}_{\mathsf{F}(\mathsf{g}(x_1), \mathsf{s}(\mathsf{0}), y_1)}(\mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})), y_1, \mathsf{g}(x_1))) = \mathsf{F}(z, y_1, \mathsf{g}(x_1))$

unifies with $\mathsf{F}(\mathsf{g}(x_2), \mathsf{s}(\mathsf{0}), y_2)$ (and the mgu instantiates the left-hand sides to normal forms). So one would have to demand that this dependency pair should be strictly decreasing, i.e., one would obtain the constraint $\mathsf{F}(\mathsf{g}(x), \mathsf{s}(\mathsf{0}), y) \succ \mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})), y, \mathsf{g}(x))$. However, together with the remaining constraints, this inequality is not satisfied by any well-founded ordering closed under substitution, because we would have

$$\begin{aligned}\mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})), \mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0})) &\succ \mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})), \mathsf{s}(\mathsf{0}), \mathsf{g}(\mathsf{s}(\mathsf{0})))\\ &\succsim \mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})), \mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{g}(\mathsf{0})))\\ &\succsim \mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})), \mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0})).\end{aligned}$$

So we have to narrow this dependency pair. Note that the right-hand side unifies with the left-hand side of this dependency pair. However, the mgu instantiates the left-hand side to a term containing the redex $\mathsf{g}(\mathsf{s}(\mathsf{0}))$. Hence, by Thm. 10 we may indeed replace this dependency pair by its narrowings.

$$\begin{aligned}\mathsf{F}(\mathsf{g}(x), \mathsf{s}(\mathsf{0}), y) &\to \mathsf{F}(\mathsf{s}(\mathsf{g}(\mathsf{0})), y, \mathsf{g}(x))\\ \mathsf{F}(\mathsf{g}(\mathsf{s}(x)), \mathsf{s}(\mathsf{0}), y) &\to \mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})), y, \mathsf{s}(\mathsf{g}(x)))\\ \mathsf{F}(\mathsf{g}(\mathsf{0}), \mathsf{s}(\mathsf{0}), y) &\to \mathsf{F}(\mathsf{g}(\mathsf{s}(\mathsf{0})), y, \mathsf{0})\rangle\end{aligned}$$

None of these new pairs is on a cycle of the estimated innermost dependency graph. Hence, the only constraint in this example is

$$\mathsf{G}(\mathsf{s}(x)) \succ \mathsf{G}(x)$$

from the second rule of the TRS. A well-founded ordering satisfying this constraint can of course be synthesized easily (e.g., the recursive path ordering).

4.18 Smallest normalizing non-terminating one-rule string rewrite system

The following example from Geser [Ges00] is the smallest normalizing non-terminating one-rule string rewrite system.

$$\mathsf{a}(\mathsf{b}(\mathsf{a}(\mathsf{b}(x)))) \to \mathsf{b}(\mathsf{a}(\mathsf{b}(\mathsf{a}(\mathsf{a}(\mathsf{b}(x))))))$$

The dependency pairs in this example are

$$\begin{aligned}\mathsf{A}(\mathsf{b}(\mathsf{a}(\mathsf{b}(x)))) &\to \mathsf{A}(\mathsf{b}(x))\\ \mathsf{A}(\mathsf{b}(\mathsf{a}(\mathsf{b}(x)))) &\to \mathsf{A}(\mathsf{a}(\mathsf{b}(x)))\\ \mathsf{A}(\mathsf{b}(\mathsf{a}(\mathsf{b}(x)))) &\to \mathsf{A}(\mathsf{b}(\mathsf{a}(\mathsf{a}(\mathsf{b}(x))))).\end{aligned}$$

The second and the third dependency pair can be narrowed to

$$\begin{aligned}\mathsf{A}(\mathsf{b}(\mathsf{a}(\mathsf{b}(\mathsf{a}(\mathsf{b}(x)))))) &\to \mathsf{A}(\mathsf{b}(\mathsf{a}(\mathsf{b}(\mathsf{a}(\mathsf{a}(\mathsf{b}(x)))))))\\ \mathsf{A}(\mathsf{b}(\mathsf{a}(\mathsf{b}(\mathsf{a}(\mathsf{b}(x)))))) &\to \mathsf{A}(\mathsf{b}(\mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{a}(\mathsf{a}(\mathsf{b}(x))))))))).\end{aligned}$$

These dependency pairs are not on cycles of the innermost dependency graph, because their left-hand sides contain redexes. Hence, the only constraint in this example is

$$\mathsf{A}(\mathsf{b}(\mathsf{a}(\mathsf{b}(x)))) \succ \mathsf{A}(\mathsf{b}(x))$$

which is satisfied by the recursive path ordering.

4.19 An innermost terminating system which requires modularity

The following system is a variant of the well-known example of Toyama [Toy87] which requires modularity results for its innermost termination proof.

$$\begin{aligned}
\mathsf{f}(x,\mathsf{c}(x),\mathsf{c}(y)) &\to \mathsf{f}(y,y,\mathsf{f}(y,x,y))\\
\mathsf{f}(\mathsf{s}(x),y,z) &\to \mathsf{f}(x,\mathsf{s}(\mathsf{c}(y)),\mathsf{c}(z))\\
\mathsf{f}(\mathsf{c}(x),x,y) &\to \mathsf{c}(y)\\
\mathsf{g}(x,y) &\to x\\
\mathsf{g}(x,y) &\to y
\end{aligned}$$

The system is not terminating as can be seen from the following infinite (cycling) reduction.

$$\begin{aligned}
\mathsf{f}(x,\mathsf{c}(x),\mathsf{c}(\mathsf{g}(x,\mathsf{c}(x)))) &\to\\
\mathsf{f}(\mathsf{g}(x,\mathsf{c}(x)),\mathsf{g}(x,\mathsf{c}(x)),\mathsf{f}(\mathsf{g}(x,\mathsf{c}(x)),x,\mathsf{g}(x,\mathsf{c}(x)))) &\to^*\\
\mathsf{f}(x,\mathsf{c}(x),\mathsf{f}(\mathsf{c}(x),x,\mathsf{g}(x,\mathsf{c}(x)))) &\to\\
\mathsf{f}(x,\mathsf{c}(x),\mathsf{c}(\mathsf{g}(x,\mathsf{c}(x)))) &\to \ldots
\end{aligned}$$

However, this is not an innermost reduction, because the first term contains the redex $\mathsf{g}(\ldots)$ as a proper subterm.

Here, we can use Cor. 16 for the innermost termination proof. The estimated innermost dependency graph only contains two non-empty cycles consisting of $\mathsf{F}(x,\mathsf{c}(x),\mathsf{c}(y)) \to \mathsf{F}(y,x,y)$ and $\mathsf{F}(\mathsf{s}(x),y,z) \to \mathsf{F}(x,\mathsf{s}(\mathsf{c}(y)),\mathsf{c}(z))$, respectively. (In this example, the estimated *innermost* dependency graph is not identical to the estimated dependency graph, because in the latter there would also be an arc from $\mathsf{F}(x,\mathsf{c}(x),\mathsf{c}(y)) \to \mathsf{F}(y,y,\mathsf{f}(y,x,y))$ to itself.)

As both cycles consist of dependency pairs without usable rules, it suffices to prove innermost termination of the two one-rules systems consisting of the first and the second rule respectively. In fact, these subsystems are even simply terminating. For

$$\mathsf{f}(x,\mathsf{c}(x),\mathsf{c}(y)) \to \mathsf{f}(y,y,\mathsf{f}(y,x,y))$$

one can use a polynomial interpretation mapping $\mathsf{f}(x,y,z)$ to $x+y+z$ and $\mathsf{c}(x)$ to $5x+1$ and for

$$\mathsf{f}(\mathsf{s}(x),y,z) \to \mathsf{f}(x,\mathsf{s}(\mathsf{c}(y)),\mathsf{c}(z))$$

one can use LPO with the precedence $\mathsf{f} > \mathsf{s}$ and $\mathsf{f} > \mathsf{c}$. Hence, Cor. 16 allows us to split a non-terminating, but innermost terminating system into two simply terminating subsystems.

Alternatively, with Thm. 11 we would obtain the following constraints for our example:

$$\mathsf{F}(x, \mathsf{c}(x), \mathsf{c}(y)) \succ_1 \mathsf{F}(y, x, y) \qquad \mathsf{F}(\mathsf{s}(x), y, z) \succ_2 \mathsf{F}(x, \mathsf{s}(\mathsf{c}(y)), \mathsf{c}(z)).$$

For $\succ_1$ we may use LPO comparing subterms right-to-left and for $\succ_2$ we may use LPO comparing subterms left-to-right. Hence, innermost termination of this example can easily be proved automatically.

Note that without our modularity result, no simplification ordering would satisfy the resulting constraints $\mathsf{F}(x, \mathsf{c}(x), \mathsf{c}(y)) \succ \mathsf{F}(y, x, y)$ and $\mathsf{F}(\mathsf{s}(x), y, z) \succ \mathsf{F}(x, \mathsf{s}(\mathsf{c}(y)), \mathsf{c}(z))$. The reason is that one cannot use an argument filtering which eliminates the arguments of c or s, and hence, these constraints imply

$$\mathsf{F}(x, \mathsf{c}(x), \mathsf{c}(\mathsf{s}(x))) \succ \mathsf{F}(\mathsf{s}(x), x, \mathsf{s}(x)) \succ \mathsf{F}(x, \mathsf{s}(\mathsf{c}(x)), \mathsf{c}(\mathsf{s}(x))).$$

4.20 Different eliminations, version 1

The following TRS is also a short example for a system where modularity is necessary.

$$\begin{aligned} \mathsf{f}(\mathsf{f}(x)) &\to \mathsf{f}(x) \\ \mathsf{g}(\mathsf{0}) &\to \mathsf{g}(\mathsf{f}(\mathsf{0})) \end{aligned}$$

The system is not simply terminating and an automated innermost termination proof using dependency pairs requires the use of our modularity results. The reason is that due to $\mathsf{F}(\mathsf{f}(x)) \succ \mathsf{F}(x)$, the argument of f cannot be eliminated and hence, no quasi-simplification ordering satisfies the constraint $\mathsf{G}(\mathsf{0}) \succ \mathsf{G}(\mathsf{f}(\mathsf{0}))$.

But innermost termination can easily be proved using Cor. 15. The $\mathcal{R}_0$-system (consisting of the f-rule) is obviously terminating and for the $\mathcal{R}_1$-constraints the argument of f is eliminated. Then these constraints are satisfied by RPO (using the precedence $\mathsf{0} > \mathsf{f}$).
A similar innermost termination proof is also possible for the TRS

$$\begin{aligned} \mathsf{f}(\mathsf{f}(x)) &\to \mathsf{f}(x) \\ \mathsf{f}(\mathsf{s}(x)) &\to \mathsf{f}(x) \\ \mathsf{g}(\mathsf{s}(\mathsf{0})) &\to \mathsf{g}(\mathsf{f}(\mathsf{s}(\mathsf{0}))). \end{aligned}$$

4.21 Different eliminations, version 2

By adding two symmetrical rules, the TRS of Ex. 4.20 is turned into a system which is no hierarchical combination any more.

$$\begin{aligned} \mathsf{f}(\mathsf{1}) &\to \mathsf{f}(\mathsf{g}(\mathsf{1})) \\ \mathsf{f}(\mathsf{f}(x)) &\to \mathsf{f}(x) \\ \mathsf{g}(\mathsf{0}) &\to \mathsf{g}(\mathsf{f}(\mathsf{0})) \\ \mathsf{g}(\mathsf{g}(x)) &\to \mathsf{g}(x). \end{aligned}$$

The dependency pairs in this example are

$$\mathsf{F}(1) \to \mathsf{F}(\mathsf{g}(1)) \tag{15}$$
$$\mathsf{F}(1) \to \mathsf{G}(1) \tag{16}$$
$$\mathsf{F}(\mathsf{f}(x)) \to \mathsf{F}(x) \tag{17}$$
$$\mathsf{G}(0) \to \mathsf{G}(\mathsf{f}(0)) \tag{18}$$
$$\mathsf{G}(0) \to \mathsf{F}(0) \tag{19}$$
$$\mathsf{G}(\mathsf{g}(x)) \to \mathsf{G}(x). \tag{20}$$

The cycles are $\{(15)\}, \{(17)\}, \{(15), (17)\}, \{(18)\}, \{(20)\}, \{(18), (20)\}$. For the constraints resulting from the first three cycles we eliminate the arguments of g, whereas for the last three cycles we eliminate the arguments of f. Then the constraints are satisfied by RPO.

4.22 Another division example, version 1

The TRS

$$\begin{aligned}
\mathsf{quot}(0, \mathsf{s}(y), \mathsf{s}(z)) &\to 0\\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y), z) &\to \mathsf{quot}(x, y, z)\\
\mathsf{quot}(x, 0, \mathsf{s}(z)) &\to \mathsf{s}(\mathsf{quot}(x, \mathsf{s}(z), \mathsf{s}(z)))
\end{aligned}$$

is a non-simply terminating system. This TRS cannot be proved terminating automatically by the technique of Sect. 2.1. The only two generated inequalities are

$$\begin{aligned}
\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y), z) &\succ \mathsf{QUOT}(x, y, z)\\
\mathsf{QUOT}(x, 0, \mathsf{s}(z)) &\succsim \mathsf{QUOT}(x, \mathsf{s}(z), \mathsf{s}(z)),
\end{aligned}$$

since there are no usable rules. By using the argument filtering $\pi(\mathsf{QUOT}) = 1$, the obtained inequalities are satisfied by the recursive path ordering. Thus, the TRS is innermost terminating. Termination of the TRS can now be concluded from the fact that it is non-overlapping.

4.23 Narrowing to approximate the innermost dependency graph

Similar to Ex. 3.40, narrowing of pairs also helps to obtain a better approximation of the *innermost* dependency graph. To illustrate this, let us replace the last rule of the TRS in Ex. 4.22 by the following three rules.

$$\begin{aligned}
0 + y &\to y\\
\mathsf{s}(x) + y &\to \mathsf{s}(x + y)\\
\mathsf{quot}(x, 0, \mathsf{s}(z)) &\to \mathsf{s}(\mathsf{quot}(x, z + \mathsf{s}(0), \mathsf{s}(z)))
\end{aligned}$$

Now instead of dependency pair

$$\mathsf{QUOT}(x, \mathsf{0}, \mathsf{s}(z)) \to \mathsf{QUOT}(x, \mathsf{s}(z), \mathsf{s}(z)) \tag{21}$$

we obtain the dependency pair

$$\mathsf{QUOT}(x, \mathsf{0}, \mathsf{s}(z)) \to \mathsf{QUOT}(x, z + \mathsf{s}(\mathsf{0}), \mathsf{s}(z)). \tag{22}$$

Note that in the estimated innermost dependency graph there would be an arc from (22) to itself, because after replacing $z + \mathsf{s}(\mathsf{0})$ by a new variable, the right- and the left-hand side of (22) obviously unify (and an instantiation with the mgu is a normal form). Hence, due to Thm. 11 we would have to find an ordering such that (22) is strictly decreasing. But then no linear or weakly monotonic polynomial ordering satisfies all resulting inequalities in this example (and the recursive path ordering does not succeed either).

However, in the *real* innermost dependency graph, there is no arc from (22) to itself, because, similar to the original dependency pair (21), there is no substitution σ such that $(z + \mathsf{s}(\mathsf{0}))\sigma$ reduces to $\mathsf{0}$. Hence, there is no cycle consisting of (22) only and therefore it is sufficient if (22) is just *weakly* decreasing. For this reason we replace the dependency pair (22) by its narrowings, viz.

$$\mathsf{QUOT}(x, \mathsf{0}, \mathsf{s}(\mathsf{0})) \to \mathsf{QUOT}(x, \mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0})) \tag{23}$$

$$\mathsf{QUOT}(x, \mathsf{0}, \mathsf{s}(\mathsf{s}(z))) \to \mathsf{QUOT}(x, \mathsf{s}(z + \mathsf{s}(\mathsf{0})), \mathsf{s}(\mathsf{0})) \tag{24}$$

and compute the innermost dependency graph afterwards. Now neither (23) nor (24) are on a cycle in the estimated innermost dependency graph. Hence, if in our example we perform at least one narrowing step, then we can determine that the dependency pair (22) does not form a cycle in the innermost dependency graph and then termination can again be verified using the recursive path ordering.

4.24 Intervals of natural numbers

The following TRS from Steinbach [Ste95a]

$$\begin{aligned}
\mathsf{intlist}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{intlist}(x \bullet y) &\to \mathsf{s}(x) \bullet \mathsf{intlist}(y)\\
\mathsf{int}(\mathsf{0}, \mathsf{0}) &\to \mathsf{0} \bullet \mathsf{nil}\\
\mathsf{int}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{0} \bullet \mathsf{int}(\mathsf{s}(\mathsf{0}), \mathsf{s}(y))\\
\mathsf{int}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{nil}\\
\mathsf{int}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{intlist}(\mathsf{int}(x, y))
\end{aligned}$$

is non-overlapping, too. The set of usable rules is empty and the generated inequalities are

$$\begin{aligned}
\mathsf{INTLIST}(x \bullet y) &\succ \mathsf{INTLIST}(y)\\
\mathsf{INT}(\mathsf{0}, \mathsf{s}(y)) &\succsim \mathsf{INT}(\mathsf{s}(\mathsf{0}), \mathsf{s}(y))\\
\mathsf{INT}(\mathsf{s}(x), \mathsf{s}(y)) &\succ \mathsf{INT}(x, y).
\end{aligned}$$

By using the argument filtering $\pi(\mathsf{INT}) = 2$ these inequalities are satisfied by the recursive path ordering. Thus, the TRS is terminating. Again, termination of this system cannot be proved automatically using the method of Sect. 2.1.

4.25 Another non-totally terminating TRS

To prove termination of the system

$$\begin{aligned} \mathsf{f}(x,x) &\to \mathsf{f}(\mathsf{g}(x),x)\\ \mathsf{g}(x) &\to \mathsf{s}(x), \end{aligned}$$

we apply narrowing on the dependency pair $\mathsf{F}(x,x) \to \mathsf{F}(\mathsf{g}(x),x)$. In this way we can directly determine that the innermost dependency graph does not contain any cycles.

4.26 Narrowing of dependency pairs for innermost termination

In the following example we also have to apply narrowing of dependency pairs.

$$\begin{aligned} \mathsf{p}(\mathsf{0}) &\to \mathsf{0}\\ \mathsf{p}(\mathsf{s}(x)) &\to x\\ \mathsf{le}(\mathsf{0},y) &\to \mathsf{true}\\ \mathsf{le}(\mathsf{s}(x),\mathsf{0}) &\to \mathsf{false}\\ \mathsf{le}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{le}(x,y)\\ \mathsf{minus}(x,y) &\to \mathsf{if}(\mathsf{le}(x,y),x,y)\\ \mathsf{if}(\mathsf{true},x,y) &\to \mathsf{0}\\ \mathsf{if}(\mathsf{false},x,y) &\to \mathsf{s}(\mathsf{minus}(\mathsf{p}(x),y)) \end{aligned}$$

Note that without narrowing, the resulting constraints would imply $\mathsf{MINUS}(\mathsf{s}(x), \mathsf{0}) \succ \mathsf{MINUS}(\mathsf{p}(\mathsf{s}(x)),\mathsf{0})$. Therefore an automatic innermost termination proof using quasi-simplification orderings would fail.

However, if we replace the dependency pair $\mathsf{MINUS}(x,y) \to \mathsf{IF}(\mathsf{le}(x,y),x,y)$ by its narrowings

$$\begin{aligned} &\mathsf{MINUS}(\mathsf{0},y) \to \mathsf{IF}(\mathsf{true},\mathsf{0},y),\\ &\mathsf{MINUS}(\mathsf{s}(x),\mathsf{0}) \to \mathsf{IF}(\mathsf{false},\mathsf{s}(x),\mathsf{0}),\\ &\mathsf{MINUS}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{IF}(\mathsf{le}(x,y),\mathsf{s}(x),\mathsf{s}(y)) \end{aligned}$$

then this also enables a narrowing of the dependency pair $\mathsf{IF}(\mathsf{false},x,y) \to \mathsf{MINUS}(\mathsf{p}(x),y)$ (whose right-hand side unified with a left-hand side before). Hence, now this dependency pair can be replaced by

$$\begin{aligned} &\mathsf{IF}(\mathsf{false},\mathsf{0},y) \to \mathsf{MINUS}(\mathsf{0},y),\\ &\mathsf{IF}(\mathsf{false},\mathsf{s}(x),y) \to \mathsf{MINUS}(x,y). \end{aligned}$$

Note that the first narrowing step would not have been possible with the method of Sect. 2.1, because the right-hand side is not linear. The inequalities corresponding to cycles are

$$\begin{aligned}
\mathsf{LE}(\mathsf{s}(x),\mathsf{s}(y)) &\succ \mathsf{LE}(x,y)\\
\mathsf{MINUS}(\mathsf{s}(x),\mathsf{0}) &\succsim \mathsf{IF}(\mathsf{false},\mathsf{s}(x),\mathsf{0})\\
\mathsf{MINUS}(\mathsf{s}(x),\mathsf{s}(y)) &\succsim \mathsf{IF}(\mathsf{le}(x,y),\mathsf{s}(x),\mathsf{s}(y))\\
\mathsf{IF}(\mathsf{false},\mathsf{s}(x),y) &\succ \mathsf{MINUS}(x,y).
\end{aligned}$$

Using the argument filtering $\pi(\mathsf{IF}) = [2,3]$, the resulting constraints are satisfied by the recursive path ordering. As the TRS is non-overlapping, in this way we have also proved its termination.

4.27 Subtraction and predecessor

The following system is an alternative way to define subtraction using the predecessor function. Again this TRS is terminating, but not simply terminating.

$$\begin{aligned}
\mathsf{p}(\mathsf{0}) &\to \mathsf{0}\\
\mathsf{p}(\mathsf{s}(x)) &\to x\\
\mathsf{le}(\mathsf{0},y) &\to \mathsf{true}\\
\mathsf{le}(\mathsf{s}(x),\mathsf{0}) &\to \mathsf{false}\\
\mathsf{le}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{le}(x,y)\\
\mathsf{minus}(x,\mathsf{0}) &\to x\\
\mathsf{minus}(x,\mathsf{s}(y)) &\to \mathsf{if}(\mathsf{le}(x,\mathsf{s}(y)),\mathsf{0},\mathsf{p}(\mathsf{minus}(x,\mathsf{p}(\mathsf{s}(y)))))\\
\mathsf{if}(\mathsf{true},x,y) &\to x\\
\mathsf{if}(\mathsf{false},x,y) &\to y
\end{aligned}$$

If we narrow the dependency pair $\mathsf{MINUS}(x,\mathsf{s}(y)) \to \mathsf{MINUS}(x,\mathsf{p}(\mathsf{s}(y)))$, then we obtain the new pair $\mathsf{MINUS}(x,\mathsf{s}(y)) \to \mathsf{MINUS}(x,y)$. Now (as there are no usable rules any more) the only constraints are

$$\begin{aligned}
\mathsf{LE}(\mathsf{s}(x),\mathsf{s}(y)) &\succ \mathsf{LE}(x,y)\\
\mathsf{MINUS}(x,\mathsf{s}(y)) &\succ \mathsf{MINUS}(x,y),
\end{aligned}$$

which are satisfied by the recursive path ordering. Hence, innermost termination (and thereby, termination) has been proved, as the TRS is non-overlapping.

A similar example was mentioned by Steinbach [Ste95a], but there the rules for le and if were missing.

4.28 Length of bit representation

The following non-simply terminating TRS corresponds to the logarithm example (Ex. 3.7). Here, $\mathsf{bits}(x)$ computes the number of bits that are necessary to represent all numbers smaller than or equal to x.

$$\begin{aligned}
\mathsf{half}(\mathsf{0}) &\to \mathsf{0}\\
\mathsf{half}(\mathsf{s}(\mathsf{0})) &\to \mathsf{0}\\
\mathsf{half}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{half}(x))\\
\mathsf{bits}(\mathsf{0})) &\to \mathsf{0}\\
\mathsf{bits}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{bits}(\mathsf{half}(\mathsf{s}(x))))
\end{aligned}$$

After narrowing $\mathsf{BITS}(\mathsf{s}(x)) \to \mathsf{BITS}(\mathsf{half}(\mathsf{s}(x)))$ to $\mathsf{BITS}(\mathsf{s}(\mathsf{0})) \to \mathsf{BITS}(\mathsf{0})$ and $\mathsf{BITS}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{BITS}(\mathsf{s}(\mathsf{half}(x))$ we obtain the inequalities

$$\begin{aligned}
\mathsf{HALF}(\mathsf{s}(\mathsf{s}(x))) &\succ \mathsf{HALF}(x)\\
\mathsf{BITS}(\mathsf{s}(\mathsf{s}(x))) &\succ \mathsf{BITS}(\mathsf{s}(\mathsf{half}(x)).
\end{aligned}$$

The resulting constraints are satisfied by the recursive path ordering.

4.29 Multiplication for even and odd numbers

The following non-simply terminating example is inspired by Walther [Wal91].

$$\begin{aligned}
\mathsf{even}(\mathsf{0}) &\to \mathsf{true}\\
\mathsf{even}(\mathsf{s}(\mathsf{0})) &\to \mathsf{false}\\
\mathsf{even}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{even}(x)\\
\mathsf{half}(\mathsf{0}) &\to \mathsf{0}\\
\mathsf{half}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{half}(x))\\
\mathsf{plus}(\mathsf{0}, y) &\to y\\
\mathsf{plus}(\mathsf{s}(x), y) &\to \mathsf{s}(\mathsf{plus}(x, y))\\
\mathsf{times}(\mathsf{0}, y) &\to \mathsf{0}\\
\mathsf{times}(\mathsf{s}(x), y) &\to \mathsf{if}_{\mathsf{times}}(\mathsf{even}(\mathsf{s}(x)), \mathsf{s}(x), y)\\
\mathsf{if}_{\mathsf{times}}(\mathsf{true}, \mathsf{s}(x), y) &\to \mathsf{plus}(\mathsf{times}(\mathsf{half}(\mathsf{s}(x)), y), \mathsf{times}(\mathsf{half}(\mathsf{s}(x)), y))\\
\mathsf{if}_{\mathsf{times}}(\mathsf{false}, \mathsf{s}(x), y) &\to \mathsf{plus}(y, \mathsf{times}(x, y))
\end{aligned}$$

To prove termination using a quasi-simplification ordering, we have to narrow the dependency pair $\mathsf{IF}_{\mathsf{times}}(\mathsf{true}, \mathsf{s}(x), y) \to \mathsf{TIMES}(\mathsf{half}(\mathsf{s}(x)), y)$ to

$$\mathsf{IF}_{\mathsf{times}}(\mathsf{true}, \mathsf{s}(\mathsf{s}(x)), y) \to \mathsf{TIMES}(\mathsf{s}(\mathsf{half}(x)), y).$$

Now the inequalities corresponding to cycles are the following.

$$\begin{aligned}
\mathsf{EVEN}(\mathsf{s}(\mathsf{s}(x))) &\succ \mathsf{EVEN}(x)\\
\mathsf{HALF}(\mathsf{s}(\mathsf{s}(x))) &\succ \mathsf{HALF}(x)\\
\mathsf{PLUS}(\mathsf{s}(x), y) &\succ \mathsf{PLUS}(x, y)\\
\mathsf{TIMES}(\mathsf{s}(x), y) &\succsim \mathsf{IF}_{\mathsf{times}}(\mathsf{even}(\mathsf{s}(x)), \mathsf{s}(x), y)\\
\mathsf{IF}_{\mathsf{times}}(\mathsf{true}, \mathsf{s}(\mathsf{s}(x)), y) &\succ \mathsf{TIMES}(\mathsf{s}(\mathsf{half}(x)), y)\\
\mathsf{IF}_{\mathsf{times}}(\mathsf{false}, \mathsf{s}(x), y) &\succ \mathsf{TIMES}(x, y)
\end{aligned}$$

If an argument filtering $\pi(\mathsf{IF}_{\mathsf{times}}) = [2,3]$ is used, then the resulting constraints are satisfied by the recursive path ordering.

4.30 Narrowing for division, remainder, and gcd

The TRSs for division (Ex. 3.1–3.3) can also be transformed into systems where we need narrowing for the (innermost) termination proof. We only present one of them.

$$\begin{aligned}
\mathsf{minus}(x, \mathsf{0}) &\to x \\
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{minus}(x, y) \\
\mathsf{le}(\mathsf{0}, y) &\to \mathsf{true} \\
\mathsf{le}(\mathsf{s}(x), \mathsf{0}) &\to \mathsf{false} \\
\mathsf{le}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{le}(x, y) \\
\mathsf{quot}(x, \mathsf{s}(y)) &\to \mathsf{if}_{\mathsf{quot}}(\mathsf{le}(\mathsf{s}(y), x), x, \mathsf{s}(y)) \\
\mathsf{if}_{\mathsf{quot}}(\mathsf{true}, x, y) &\to \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), y)) \\
\mathsf{if}_{\mathsf{quot}}(\mathsf{false}, x, y) &\to \mathsf{0}
\end{aligned}$$

Again this system is not simply terminating. After narrowing the dependency pair $\mathsf{QUOT}(x, \mathsf{s}(y)) \to \mathsf{IF}_{\mathsf{quot}}(\mathsf{le}(\mathsf{s}(y), x), x, \mathsf{s}(y))$ to

$$\begin{aligned}
&\mathsf{QUOT}(\mathsf{0}, \mathsf{s}(y)) \to \mathsf{IF}_{\mathsf{quot}}(\mathsf{false}, \mathsf{0}, \mathsf{s}(y)) \\
&\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{IF}_{\mathsf{quot}}(\mathsf{le}(y, x), \mathsf{s}(x), \mathsf{s}(y))
\end{aligned}$$

we can narrow $\mathsf{IF}_{\mathsf{quot}}(\mathsf{true}, x, y) \to \mathsf{QUOT}(\mathsf{minus}(x, y), y)$ to

$$\begin{aligned}
&\mathsf{IF}_{\mathsf{quot}}(\mathsf{true}, x, \mathsf{0}) \to \mathsf{QUOT}(x, y) \\
&\mathsf{IF}_{\mathsf{quot}}(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{QUOT}(\mathsf{minus}(x, y), \mathsf{s}(y)).
\end{aligned}$$

Now the inequalities corresponding to cycles are

$$\begin{aligned}
\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y)) &\succ \mathsf{MINUS}(x, y) \\
\mathsf{LE}(\mathsf{s}(x), \mathsf{s}(y)) &\succ \mathsf{LE}(x, y) \\
\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) &\succsim \mathsf{IF}_{\mathsf{quot}}(\mathsf{le}(y, x), \mathsf{s}(x), \mathsf{s}(y)) \\
\mathsf{IF}_{\mathsf{quot}}(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) &\succ \mathsf{QUOT}(\mathsf{minus}(x, y), \mathsf{s}(y)).
\end{aligned}$$

Using the argument filtering $\pi(\mathsf{minus}) = 1$, $\pi(\mathsf{IF}) = [2,3]$ the constraints are satisfied by the recursive path ordering. Hence, in this way (innermost) termination of this TRS is proved.

A simpler modification of the quotient TRS where one should also use narrowing is obtained if instead of the last three rules the following rules are used.

$$\begin{aligned}
&\mathsf{quot}(\mathsf{0}, \mathsf{s}(y)) \to \mathsf{0} \\
&\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{s}(\mathsf{quot}(\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{s}(y)))
\end{aligned}$$

A similar modification is also possible for the remainder TRSs (Ex. 3.5), i.e., the rule $\mathsf{if}_{\mathsf{mod}}(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{mod}(\mathsf{minus}(x, y), \mathsf{s}(y))$ may be replaced by

$$\mathsf{if}_{\mathsf{mod}}(\mathsf{true}, x, y) \to \mathsf{mod}(\mathsf{minus}(x, y), y).$$

In an analogous way, in the greatest common divisor TRSs (Ex. 3.6) one could also replace the last two rules by

$$\begin{aligned} \mathsf{if}_{\mathsf{gcd}}(\mathsf{true}, x, y) &\to \mathsf{gcd}(\mathsf{minus}(x, y), y) \\ \mathsf{if}_{\mathsf{gcd}}(\mathsf{false}, x, y) &\to \mathsf{gcd}(\mathsf{minus}(y, x), x). \end{aligned}$$

All these modified TRSs could again be proved (innermost) terminating by using narrowing first.

4.31 Braid problem

The following string rewrite system (which encodes a braid problem from topology) was given by Zantema as a challenge during the 3rd International Termination Workshop. As shown by Geser, it is not simply terminating.

$$\begin{aligned} \mathsf{a}(\mathsf{d}(x)) &\to \mathsf{d}(\mathsf{c}(\mathsf{b}(\mathsf{a}(x)))) \\ \mathsf{b}(\mathsf{c}(x)) &\to \mathsf{c}(\mathsf{d}(\mathsf{a}(\mathsf{b}(x)))) \\ \mathsf{a}(\mathsf{c}(x)) &\to x \\ \mathsf{b}(\mathsf{d}(x)) &\to x \end{aligned}$$

The dependency pairs in this example are

$$\mathsf{A}(\mathsf{d}(x)) \to \mathsf{A}(x) \tag{25}$$
$$\mathsf{A}(\mathsf{d}(x)) \to \mathsf{B}(\mathsf{a}(x)) \tag{26}$$
$$\mathsf{B}(\mathsf{c}(x)) \to \mathsf{B}(x) \tag{27}$$
$$\mathsf{B}(\mathsf{c}(x)) \to \mathsf{A}(\mathsf{b}(x)). \tag{28}$$

Dependency pair (26) can be replaced by its narrowings

$$\begin{aligned} \mathsf{A}(\mathsf{d}(\mathsf{d}(x))) &\to \mathsf{B}(\mathsf{d}(\mathsf{c}(\mathsf{b}(\mathsf{a}(x))))) \\ \mathsf{A}(\mathsf{d}(\mathsf{c}(x))) &\to \mathsf{B}(x) \end{aligned}$$

and dependency pair (28) can be narrowed to

$$\begin{aligned} \mathsf{B}(\mathsf{c}(\mathsf{c}(x))) &\to \mathsf{A}(\mathsf{c}(\mathsf{d}(\mathsf{a}(\mathsf{b}(x))))) \\ \mathsf{B}(\mathsf{c}(\mathsf{d}(x))) &\to \mathsf{A}(x). \end{aligned}$$

As there are no usable rules, the resulting constraints are

$$\begin{aligned} \mathsf{A}(\mathsf{d}(x)) &\succ \mathsf{A}(x) \\ \mathsf{A}(\mathsf{d}(\mathsf{c}(x))) &\succsim \mathsf{B}(x) \\ \mathsf{B}(\mathsf{c}(x)) &\succ \mathsf{B}(x) \\ \mathsf{B}(\mathsf{c}(\mathsf{d}(x))) &\succsim \mathsf{A}(x), \end{aligned}$$

which are satisfied by the recursive path ordering. Hence, as the TRS is non-overlapping, its termination is proved.

4.32 A non-overlapping system which requires modularity

The following system is a non-overlapping variant of Ex. 3.47, which can be obtained by replacing y in the second rule by $\mathsf{s}(y)$.

$$\mathsf{f}(x, \mathsf{c}(y)) \to \mathsf{f}(x, \mathsf{s}(\mathsf{f}(y, y)))$$
$$\mathsf{f}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{f}(x, \mathsf{s}(\mathsf{c}(\mathsf{s}(y))))$$

Again the system is not simply terminating (we have the same reduction as in Ex. 3.47). Similar to that example, an automatic termination or innermost termination proof without modularity fails, because the resulting constraints imply $\mathsf{F}(x, \mathsf{c}(\mathsf{s}(x))) \succ \mathsf{F}(x, \mathsf{s}(\mathsf{c}(\mathsf{s}(x))))$, which is not satisfied by any simplification ordering.

In this example, we obtain the estimated dependency graph in Fig. 1 (which is identical to the estimated innermost dependency graph).

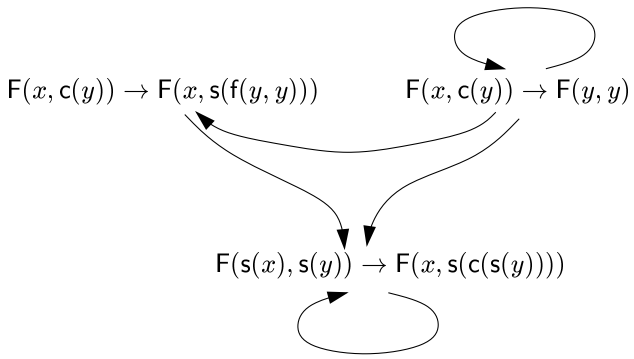

Fig. 1. The estimated (innermost) dependency graph in Ex. 4.32.

This example is non-overlapping and hence, we can prove termination by verifying innermost termination. For that purpose we may use Cor. 16. As the sets of usable rules are empty for both dependency pairs $\mathsf{F}(x, \mathsf{c}(y)) \to \mathsf{F}(y, y)$ and $\mathsf{F}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{F}(x, \mathsf{s}(\mathsf{c}(\mathsf{s}(y))))$, we can split the original TRS into the two subsystems consisting of one of the rules respectively. Now termination of

$$\mathsf{f}(x, \mathsf{c}(y)) \to \mathsf{f}(x, \mathsf{s}(\mathsf{f}(y, y)))$$

is proved using the lexicographic or the recursive path ordering with precedence $\mathsf{c} > \mathsf{s}$ and $\mathsf{c} > \mathsf{f}$. Termination of

$$\mathsf{f}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{f}(x, \mathsf{s}(\mathsf{c}(\mathsf{s}(y))))$$

is proved using the lexicographic path ordering with precedence $\mathsf{f} > \mathsf{s}$ and $\mathsf{f} > \mathsf{c}$. In this way, the two simply terminating subsystems imply termination of the whole (non-simply terminating) TRS.

4.33 Sum and weight

The following TRS computes the weighted sum of a list.

$$
\begin{aligned}
\mathsf{sum}(\mathsf{s}(n){\bullet}x, m{\bullet}y) &\to \mathsf{sum}(n{\bullet}x, \mathsf{s}(m){\bullet}y)\\
\mathsf{sum}(\mathsf{0}{\bullet}x, y) &\to \mathsf{sum}(x, y)\\
\mathsf{sum}(\mathsf{nil}, y) &\to y\\
\mathsf{weight}(n{\bullet}m{\bullet}x) &\to \mathsf{weight}(\mathsf{sum}(n{\bullet}m{\bullet}x, \mathsf{0}{\bullet}x))\\
\mathsf{weight}(n{\bullet}\mathsf{nil}) &\to n
\end{aligned}
$$

The system is a hierarchical combination of the sum-rules ($\mathcal{R}_0$) and the weight-rules ($\mathcal{R}_1$). Note that it is not a proper extension and $\mathcal{R}_1$ is not oblivious of $\mathcal{R}_0$. Moreover, the TRS is obviously not simply terminating. Its estimated dependency graph (which is identical to the estimated innermost dependency graph) is sketched in Fig. 2.

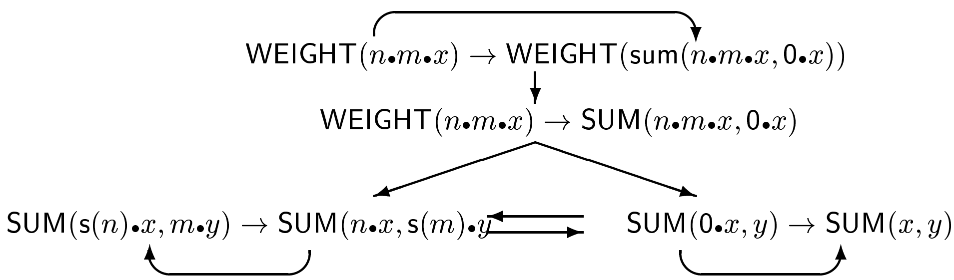

Fig. 2. The estimated (innermost) dependency graph in Ex. 4.33.

As the TRS is non-overlapping, it suffices to prove innermost termination. However, without modularity, the resulting constraints would not be satisfied by any quasi-simplification ordering: Due to the constraint $\mathsf{SUM}(\mathsf{s}(n){\bullet}x, m{\bullet}y) \succ \mathsf{SUM}(n{\bullet}x, \mathsf{s}(m){\bullet}y)$, neither the argument of s nor the first argument of '$\bullet$' can be eliminated. As we cannot eliminate all arguments of sum (due to the constraint $\mathsf{sum}(\mathsf{nil}, y) \succsim y$), the constraint $\mathsf{sum}(\mathsf{s}(n){\bullet}x, m{\bullet}y) \succsim \mathsf{sum}(n{\bullet}x, \mathsf{s}(m){\bullet}y)$ enforces that the first argument of sum may not be deleted either. But $\mathsf{WEIGHT}(n{\bullet}m{\bullet}x) \succ \mathsf{WEIGHT}(\mathsf{sum}(n{\bullet}m{\bullet}x, \ldots))$ does not hold for any quasi-simplification ordering.

Termination of the sum and weight-example can be proved by Cor. 15. The sum-subsystem ($\mathcal{R}_0$) is terminating (this can be proved by LPO with the precedence $\mathsf{sum} > \bullet$ and $\mathsf{sum} > \mathsf{s}$). For the weight-subsystem ($\mathcal{R}_1$) we obtain the constraints

$$
\begin{aligned}
\mathsf{sum}(\mathsf{s}(n){\bullet}x, m{\bullet}y) &\succsim \mathsf{sum}(n{\bullet}x, \mathsf{s}(m){\bullet}y)\\
\mathsf{sum}(\mathsf{0}{\bullet}x, y) &\succsim \mathsf{sum}(x, y)\\
\mathsf{sum}(\mathsf{nil}, y) &\succsim y\\
\mathsf{WEIGHT}(n{\bullet}m{\bullet}x) &\succ \mathsf{WEIGHT}(\mathsf{sum}(n{\bullet}m{\bullet}x, \mathsf{0}{\bullet}x)),
\end{aligned}
$$

which are also satisfied by LPO after deleting the first arguments of sum and '$\bullet$'. This time we have to use the precedence $\bullet > \mathsf{sum}$.

Note that the constraints for termination (according to Sect. 2.1) are not satisfied by any quasi-simplification ordering amenable to automation, i.e., this example again shows that proving *innermost* termination is essentially easier than proving termination.

To see this, regard the constraints for the cycle consisting of the first SUM-dependency pair. We show that they are not satisfied by any argument filtering π and any reduction pair $(\succsim, \succ)$ where $\succsim$ is a path or a polynomial quasi-simplification ordering. The constraint $\pi(\mathsf{SUM}(\mathsf{s}(n)\bullet x, m\bullet y)) \succ \pi(\mathsf{SUM}(n\bullet x, \mathsf{s}(m)\bullet y))$ implies $\pi(\mathsf{s}(0)\bullet\mathsf{nil}) \succ^s \pi(0\bullet\mathsf{nil})$ where $\succ^s$ is the strict part of the quasi-ordering $\succsim$. (For polynomial orderings this holds because then $\succsim$ is total.) Moreover, one can show that the constraints entail that $\pi(\mathsf{weight}(\mathsf{sum}(0\bullet 0\bullet\mathsf{s}(0)\bullet\mathsf{nil}, 0\bullet x)))$ contains x and if one uses polynomial orderings, this term is mapped to a polynomial which is strongly monotonic in x w.r.t. the ordering $\succ^s$. Then we obtain the following contradiction to the well-foundedness of $\succ^s$:

$$\begin{array}{lll}
& \pi(\mathsf{weight}(\mathsf{sum}(0\bullet 0\bullet\mathsf{s}(0)\bullet\mathsf{nil}, 0\bullet\mathsf{s}(0)\bullet\mathsf{nil}))) & \\
\succ^s & \pi(\mathsf{weight}(\mathsf{sum}(0\bullet 0\bullet\mathsf{s}(0)\bullet\mathsf{nil}, 0\bullet 0\bullet\mathsf{nil}))) & \text{by monotonicity} \\
& & \text{and } \pi(\mathsf{s}(0)\bullet\mathsf{nil}) \succ^s \pi(0\bullet\mathsf{nil}) \\
\succsim & \pi(\mathsf{weight}(0\bullet 0\bullet\mathsf{s}(0)\bullet\mathsf{nil})) & \text{by the subterm property} \\
\succsim & \pi(\mathsf{weight}(\mathsf{sum}(0\bullet 0\bullet\mathsf{s}(0)\bullet\mathsf{nil}, 0\bullet\mathsf{s}(0)\bullet\mathsf{nil}))) & \text{by the constraint} \\
& & \text{from the first } \mathsf{weight}\text{-rule}
\end{array}$$

(To show the strong monotonicity of $\pi(\mathsf{weight}(\mathsf{sum}(0\bullet 0\bullet\mathsf{s}(0)\bullet\mathsf{nil}, 0\bullet x)))$ note that the second weight-rule implies that $\pi(\mathsf{weight}(x))$ must depend on x. Moreover, the constraint for the last sum-rule implies that $\pi(\mathsf{sum}(0\bullet 0\bullet\mathsf{s}(0)\bullet\mathsf{nil}, x))$ must depend on x. It remains to show that $\pi(0\bullet x)$ depends on x. To this end, note that the constraint from the first sum-rule implies that $\pi(\mathsf{sum}(x, 0\bullet\mathsf{nil}))$ depends on x. Therefore, the constraint from the second sum-rule implies that $\pi(\mathsf{sum}(0\bullet x, 0\bullet\mathsf{nil}))$ also depends on x. But then $\pi(0\bullet x)$ must also depend on x.)

4.34 Renaming in the Lambda Calculus (simplified variant)

The following TRS is a shortened and simplified variant of a system for renaming in the lambda calculus. The full system is presented in Ex. 4.35.

$$\begin{aligned}
\mathsf{f}(0) &\to \mathsf{true} \\
\mathsf{f}(1) &\to \mathsf{false} \\
\mathsf{f}(\mathsf{s}(x)) &\to \mathsf{f}(x) \\
\mathsf{if}(\mathsf{true}, x, y) &\to x \\
\mathsf{if}(\mathsf{false}, x, y) &\to y \\
\mathsf{g}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{if}(\mathsf{f}(x), \mathsf{s}(x), \mathsf{s}(y)) \\
\mathsf{g}(x, \mathsf{c}(y)) &\to \mathsf{g}(x, \mathsf{g}(\mathsf{s}(\mathsf{c}(y)), y))
\end{aligned}$$

The system is not simply terminating, as the left-hand side of the last rule is embedded in its right-hand side. As it is non-overlapping, it is sufficient to prove innermost termination only. For that purpose we need modularity results, because otherwise we would have

$$\mathsf{G}(x, \mathsf{c}(\mathsf{s}(x))) \succ \mathsf{G}(x, \mathsf{g}(\mathsf{s}(\mathsf{c}(\mathsf{s}(x))), \mathsf{s}(x))) \succsim \mathsf{G}(x, \mathsf{if}(\ldots, \mathsf{s}(\mathsf{c}(\mathsf{s}(x))), \ldots))$$

and neither the argument of s nor the second argument of if can be eliminated.

The system is a hierarchical combination (but not a proper extension). Hence, we can prove innermost termination by Cor. 15. Termination of $\mathcal{R}_0$ (the f- and if-rules) can for instance be verified by RPO. For $\mathcal{R}_1$ (the g-rules) we obtain the following constraints after filtering the arguments of s and f:

$$\begin{aligned}
\mathsf{f} &\succsim \mathsf{true}\\
\mathsf{f} &\succsim \mathsf{false}\\
\mathsf{f} &\succsim \mathsf{f}\\
\mathsf{if}(\mathsf{true}, x, y) &\succsim x\\
\mathsf{if}(\mathsf{false}, x, y) &\succsim y\\
\mathsf{g}(\mathsf{s}, \mathsf{s}) &\succsim \mathsf{if}(\mathsf{f}, \mathsf{s}, \mathsf{s})\\
\mathsf{g}(x, \mathsf{c}(y)) &\succsim \mathsf{g}(x, \mathsf{g}(\mathsf{s}, y))\\
\mathsf{G}(x, \mathsf{c}(y)) &\succ \mathsf{G}(x, \mathsf{g}(\mathsf{s}, y))\\
\mathsf{G}(x, \mathsf{c}(y)) &\succ \mathsf{G}(\mathsf{s}, y).
\end{aligned}$$

These inequalities are satisfied by RPO using the precedence $\mathsf{f} > \mathsf{true}$, $\mathsf{f} > \mathsf{false}$, $\mathsf{g} > \mathsf{if}$, $\mathsf{g} > \mathsf{f}$, $\mathsf{c} > \mathsf{g}$, $\mathsf{c} > \mathsf{s}$.

4.35 Renaming in the Lambda Calculus

The following system is a variant of an algorithm from [MA96]. The purpose of the function $\mathsf{ren}(x, y, t)$ is to replace every free occurrence of the variable x in the term t by the variable y. If the substitution of x by y should be applied to a lambda term $\mathsf{lambda}(z, t)$ (which represents $\lambda z.t$), then we first apply an α-conversion step to $\mathsf{lambda}(z, t)$, i.e., we rename z to a new variable (which is different from x or y and which does not occur in $\mathsf{lambda}(z, t)$). Subsequently, the renaming of x to y is applied to the resulting term. For that reason in this TRS there is a nested recursive call of the function ren.

Variables are represented by $\mathsf{var}(l)$ where l is a list of terms. Therefore, the variable $\mathsf{var}(x \bullet y \bullet \mathsf{lambda}(z, t) \bullet \mathsf{nil})$ is distinct from x and y and from all variables occurring in $\mathsf{lambda}(z, t)$.

$$\begin{aligned}
\mathsf{and}(\mathsf{true}, y) &\to y\\
\mathsf{and}(\mathsf{false}, y) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{nil}, \mathsf{nil}) &\to \mathsf{true}
\end{aligned}$$

$$
\begin{aligned}
\mathsf{eq}(t{\bullet}l, \mathsf{nil}) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{nil}, t{\bullet}l) &\to \mathsf{false}\\
\mathsf{eq}(t{\bullet}l, t'{\bullet}l') &\to \mathsf{and}(\mathsf{eq}(t,t'), \mathsf{eq}(l,l'))\\
\mathsf{eq}(\mathsf{var}(l), \mathsf{var}(l')) &\to \mathsf{eq}(l,l')\\
\mathsf{eq}(\mathsf{var}(l), \mathsf{apply}(t,s)) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{var}(l), \mathsf{lambda}(x,t)) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{apply}(t,s), \mathsf{var}(l)) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{apply}(t,s), \mathsf{apply}(t',s')) &\to \mathsf{and}(\mathsf{eq}(t,t'), \mathsf{eq}(s,s'))\\
\mathsf{eq}(\mathsf{apply}(t,s), \mathsf{lambda}(x,t)) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{lambda}(x,t), \mathsf{var}(l)) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{lambda}(x,t), \mathsf{apply}(t,s)) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{lambda}(x,t), \mathsf{lambda}(x',t')) &\to \mathsf{and}(\mathsf{eq}(x,x'), \mathsf{eq}(t,t'))\\
\mathsf{if}(\mathsf{true}, \mathsf{var}(k), \mathsf{var}(l')) &\to \mathsf{var}(k)\\
\mathsf{if}(\mathsf{false}, \mathsf{var}(k), \mathsf{var}(l')) &\to \mathsf{var}(l')\\
\mathsf{ren}(\mathsf{var}(l), \mathsf{var}(k), \mathsf{var}(l')) &\to \mathsf{if}(\mathsf{eq}(l,l'), \mathsf{var}(k), \mathsf{var}(l'))\\
\mathsf{ren}(x,y,\mathsf{apply}(t,s)) &\to \mathsf{apply}(\mathsf{ren}(x,y,t), \mathsf{ren}(x,y,s))\\
\mathsf{ren}(x,y,\mathsf{lambda}(z,t)) &\to \mathsf{lambda}(\mathsf{var}(x{\bullet}y{\bullet}\mathsf{lambda}(z,t){\bullet}\mathsf{nil}),\\
&\qquad \mathsf{ren}(x,y,\mathsf{ren}(z,\mathsf{var}(x{\bullet}y{\bullet}\mathsf{lambda}(z,t){\bullet}\mathsf{nil}),t)))
\end{aligned}
$$

Let $\mathcal{R}_0$ consist of all rules but the last three ren-rules, and let $\mathcal{R}_1$ be the ren-subsystem. Then this TRS is a hierarchical combination of $\mathcal{R}_0$ and $\mathcal{R}_1$. The TRS is not simply terminating as the left-hand side of the last rule is embedded in its right-hand side, but it is non-overlapping. Hence, Cor. 15 can be used for the termination proof.

Termination of $\mathcal{R}_0$ can for instance be proved by RPO. To complete the termination proof, we have to find a quasi-ordering such that all rules are weakly decreasing and such that the following strict inequalities are satisfied:

$$
\begin{aligned}
\mathsf{REN}(x,y,\mathsf{apply}(t,s)) &\succ \mathsf{REN}(x,y,t)\\
\mathsf{REN}(x,y,\mathsf{apply}(t,s)) &\succ \mathsf{REN}(x,y,s)\\
\mathsf{REN}(x,y,\mathsf{lambda}(z,t)) &\succ \mathsf{REN}(x,y,\mathsf{ren}(z,\mathsf{var}(x{\bullet}y{\bullet}\mathsf{lambda}(z,t){\bullet}\mathsf{nil}),t))\\
\mathsf{REN}(x,y,\mathsf{lambda}(z,t)) &\succ \mathsf{REN}(z,\mathsf{var}(x{\bullet}y{\bullet}\mathsf{lambda}(z,t){\bullet}\mathsf{nil}),t).
\end{aligned}
$$

A well-founded ordering satisfying these constraints can be synthesized automatically. For instance, one can use the following polynomial interpretation where $\mathsf{REN}(x,y,t)$ is mapped to t, $\mathsf{ren}(x,y,t)$ is also mapped to t, $\mathsf{lambda}(x,t)$ is mapped to $t+1$, $\mathsf{apply}(t,s)$ is mapped to $t+s+1$, $\mathsf{and}(x,y)$ is mapped to y, and where nil, $\mathsf{var}(l)$, true, false, $\mathsf{eq}(t,s)$, and $\mathsf{if}(x,y,z)$ are all mapped to the constant 0.

Note that the modularity result of Cor. 15 is essential for this termination proof. If termination of the *whole* system would have to be proved at once, then the resulting inequalities would not be satisfied by any quasi-simplification ordering. The reason is that due to $\mathsf{EQ}(\mathsf{var}(l), \mathsf{var}(l')) \succ \mathsf{EQ}(l, l')$ the argument of var cannot be deleted. Hence, (as if's second argument cannot be deleted either), $\mathsf{ren}(\mathsf{var}(l), \mathsf{var}(k), \mathsf{var}(l')) \succsim \mathsf{if}(\mathsf{eq}(l, l'), \mathsf{var}(k), \mathsf{var}(l'))$ enforces that ren must depend on its second argument. Moreover, due to $\mathsf{EQ}(t{\bullet}l, t'{\bullet}l') \succ \mathsf{EQ}(t, t')$, the first argument of '$\bullet$' cannot be eliminated. But the inequality

$$\mathsf{REN}(x, y, \mathsf{lambda}(z, t)) \succ \mathsf{REN}(x, y, \mathsf{ren}(z, \mathsf{var}(x{\bullet}y{\bullet}\mathsf{lambda}(z, t){\bullet}\mathsf{nil}), t))$$

is not satisfied by any quasi-simplification ordering.

The simplified system of Ex. 4.34 is obtained from the subsystem

$$\begin{aligned}
\mathsf{eq}(\mathsf{nil}, \mathsf{nil}) &\to \mathsf{true}\\
\mathsf{eq}(\mathsf{nil}, t{\bullet}l) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{var}(l), \mathsf{var}(l')) &\to \mathsf{eq}(l, l')\\
\mathsf{if}(\mathsf{true}, \mathsf{var}(k), \mathsf{var}(l')) &\to \mathsf{var}(k)\\
\mathsf{if}(\mathsf{false}, \mathsf{var}(k), \mathsf{var}(l')) &\to \mathsf{var}(l')\\
\mathsf{ren}(\mathsf{var}(l), \mathsf{var}(k), \mathsf{var}(l')) &\to \mathsf{if}(\mathsf{eq}(l, l'), \mathsf{var}(k), \mathsf{var}(l'))\\
\mathsf{ren}(x, y, \mathsf{lambda}(z, t)) &\to \mathsf{lambda}(\mathsf{var}(x{\bullet}y{\bullet}\mathsf{lambda}(z, t){\bullet}\mathsf{nil}),\\
&\qquad\quad \mathsf{ren}(x, y, \mathsf{ren}(z, \mathsf{var}(x{\bullet}y{\bullet}\mathsf{lambda}(z, t){\bullet}\mathsf{nil}), t)))
\end{aligned}$$

by removing the first arguments of eq, ren, and lambda, by eliminating the arguments of '$\bullet$' in the second eq-rule, by replacing var by its arguments in the if-rules, by deleting a lambda and 'unnecessary' arguments of var in the last ren-rule, and by renaming the variables and function symbols (eq corresponds to f, nil corresponds to $\mathsf{0}$, '$\bullet$' corresponds to $\mathsf{1}$, var corresponds to s, ren corresponds to g, and lambda corresponds to c).

4.36 Selection sort

This TRS from [Wal94] is obviously not simply terminating. The TRS can be used to sort a list by repeatedly replacing the minimum of the list by the head of the list. It uses $\mathsf{replace}(n, m, x)$ to replace the leftmost occurrence of n in the list x by m.

$$\begin{aligned}
\mathsf{eq}(\mathsf{0}, \mathsf{0}) &\to \mathsf{true}\\
\mathsf{eq}(\mathsf{0}, \mathsf{s}(m)) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{s}(n), \mathsf{0}) &\to \mathsf{false}\\
\mathsf{eq}(\mathsf{s}(n), \mathsf{s}(m)) &\to \mathsf{eq}(n, m)\\
\mathsf{le}(\mathsf{0}, m) &\to \mathsf{true}\\
\mathsf{le}(\mathsf{s}(n), \mathsf{0}) &\to \mathsf{false}\\
\mathsf{le}(\mathsf{s}(n), \mathsf{s}(m)) &\to \mathsf{le}(n, m)\\
\mathsf{min}(\mathsf{0}{\bullet}\mathsf{nil}) &\to \mathsf{0}
\end{aligned}$$

$$
\begin{aligned}
\mathsf{min}(\mathsf{s}(n)\bullet\mathsf{nil}) &\to \mathsf{s}(n)\\
\mathsf{min}(n\bullet m\bullet x) &\to \mathsf{if}_{\mathsf{min}}(\mathsf{le}(n,m), n\bullet m\bullet x)\\
\mathsf{if}_{\mathsf{min}}(\mathsf{true}, n\bullet m\bullet x) &\to \mathsf{min}(n\bullet x)\\
\mathsf{if}_{\mathsf{min}}(\mathsf{false}, n\bullet m\bullet x) &\to \mathsf{min}(m\bullet x)\\
\mathsf{replace}(n,m,\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{replace}(n,m,k\bullet x) &\to \mathsf{if}_{\mathsf{replace}}(\mathsf{eq}(n,k), n, m, k\bullet x)\\
\mathsf{if}_{\mathsf{replace}}(\mathsf{true}, n, m, k\bullet x) &\to m\bullet x\\
\mathsf{if}_{\mathsf{replace}}(\mathsf{false}, n, m, k\bullet x) &\to k\bullet\mathsf{replace}(n,m,x)\\
\mathsf{sort}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{sort}(n\bullet x) &\to \mathsf{min}(n\bullet x)\bullet\mathsf{sort}(\mathsf{replace}(\mathsf{min}(n\bullet x), n, x))
\end{aligned}
$$

The TRS is non-overlapping and hence, verification of innermost termination is sufficient. As this is a hierarchical combination (but no proper extension and not oblivious), we can use Cor. 15.

The TRS $\mathcal{R}_0$ (consisting of all rules but the the last two ones) is innermost terminating (resp. terminating) as can be proved by the dependency pair approach. To complete the innermost termination proof we obtain the following inequality for $\mathcal{R}_1$:

$$\mathsf{SORT}(n\bullet x) \succ \mathsf{SORT}(\mathsf{replace}(\mathsf{min}(n\bullet x), n, x)).$$

Moreover, we have to demand $l \succsim r$ for all rules of $\mathcal{R}_0$, as all these rules are usable.

We use the argument filtering $\pi(\bullet) = 2$, $\pi(\mathsf{s}) = \pi(\mathsf{eq}) = \pi(\mathsf{le}) = [\,]$, and $\pi(\mathsf{replace}) = \pi(\mathsf{if}_{\mathsf{replace}}) = 3$. Then the resulting inequalities are satisfied by the recursive path ordering (where '$\bullet$' must be greater than min in the precedence).

Note that without using modularity, no path ordering like LPO or RPO which is amenable to automation would satisfy the resulting constraints. The reason is that due to $\mathsf{EQ}(\mathsf{s}(n), \mathsf{s}(m)) \succ \mathsf{EQ}(n,m)$, the argument of s cannot be eliminated and hence, $\mathsf{min}(\mathsf{s}(n)\bullet\mathsf{nil}) \succsim \mathsf{s}(n)$ implies that the first argument of '$\bullet$' cannot be deleted either. Now due to $\mathsf{if}_{\mathsf{replace}}(\mathsf{true}, n, m, k\bullet x) \succsim m\bullet x$, the third argument of $\mathsf{if}_{\mathsf{replace}}$ cannot be removed. Then $\mathsf{replace}(n,m,k\bullet x) \succsim \mathsf{if}_{\mathsf{replace}}(\ldots, n, m, k\bullet x)$ implies that $\mathsf{replace}$ must depend on its second argument and that $\mathsf{replace}$ must be greater than or equal to $\mathsf{if}_{\mathsf{replace}}$ in the precedence, i.e., $\mathsf{replace} \geq \mathsf{if}_{\mathsf{replace}}$. As $\mathsf{replace}$ depends on its second argument, $\mathsf{if}_{\mathsf{replace}}(\mathsf{false}, n, m, k\bullet x) \succsim k\bullet\mathsf{replace}(n, m, x)$ implies $\mathsf{if}_{\mathsf{replace}} \geq \bullet$. Hence, we have $\mathsf{replace} \geq \bullet$. But then $\mathsf{SORT}(n\bullet x) \succ \mathsf{SORT}(\mathsf{replace}(\ldots, n, x))$ does not hold.

However, a (non-modular) termination proof with dependency pairs would be possible by the polynomial ordering where $\mathsf{eq}(x,y)$, $\mathsf{0}$, true, false, $\mathsf{le}(x,y)$, and nil are mapped to 0, $\mathsf{s}(x)$ is mapped to $x+1$, $\mathsf{sum}(n,x)$ is mapped to $n+x+1$, $\mathsf{min}(x)$ and $\mathsf{if}_{\mathsf{min}}(b,x)$ are mapped to x, $\mathsf{replace}(n,m,x)$ and $\mathsf{if}_{\mathsf{replace}}(b,n,m,x)$ are mapped to $m+x$, $\mathsf{EQ}(x,y)$, $\mathsf{LE}(x,y)$, $\mathsf{MIN}(x)$, $\mathsf{IF}_{\mathsf{min}}(b,x)$, $\mathsf{SORT}(x)$, and $\mathsf{IF}_{\mathsf{sort}}(b,x)$ are mapped to x, and $\mathsf{REPLACE}(n,m,x)$ and $\mathsf{IF}_{\mathsf{replace}}(b,n,m,x)$ are mapped to

$m + x$. Hence, as the TRS is non-overlapping, in this way its termination is also proved. (If the first min rule would be replaced by $\mathsf{min}(\mathsf{sum}(n, \mathsf{nil})) \to \mathsf{element}(n)$, then termination could also be proved by the termination technique of Sect. 2.1 using an appropriate argument filtering and the recursive path ordering to satisfy the constraints obtained.)

4.37 Different termination arguments

The following TRS is one of the shortest systems to demonstrate the use of modularity.

$$\mathsf{f}(\mathsf{c}(\mathsf{s}(x), y)) \to \mathsf{f}(\mathsf{c}(x, \mathsf{s}(y)))$$
$$\mathsf{g}(\mathsf{c}(x, \mathsf{s}(y))) \to \mathsf{g}(\mathsf{c}(\mathsf{s}(x), y))$$

Without modularity results, termination of this system cannot be proved by path orderings like LPO or RPO that are amenable to automation and a termination proof with polynomial orderings fails, too. (The reason for the latter is that if $[f]$ is the polynomial corresponding to a function f, then $\lim_{x\to\infty}[\mathsf{c}](x, [\mathsf{s}](x)) - [\mathsf{c}]([\mathsf{s}](x), x)$ is ∞ or $-\infty$. But then (for large enough arguments) the inequalities corresponding to either the first or the second rule are not satisfied.) By Cor. 16 however, it suffices to prove termination of the two one-rule subsystems. Their termination can easily be verified (e.g., by using LPO and comparing subterms left-to-right for the first rule, whereas for the second rule they are compared right-to-left).

While termination of the above TRS could also be proved by existing modularity criteria (as it was split into subsystems with disjoint defined symbols), adding a third rule turns it into a hierarchical combination which is no proper extension and not oblivious.

$$\mathsf{f}(\mathsf{c}(\mathsf{s}(x), y)) \to \mathsf{f}(\mathsf{c}(x, \mathsf{s}(y)))$$
$$\mathsf{g}(\mathsf{c}(x, \mathsf{s}(y))) \to \mathsf{g}(\mathsf{c}(\mathsf{s}(x), y))$$
$$\mathsf{g}(\mathsf{s}(\mathsf{f}(x))) \to \mathsf{g}(\mathsf{f}(x))$$

Using Cor. 15 for the innermost termination proof, termination of $\mathcal{R}_0$ (the f-rule) is proved with LPO (comparing subterms left-to-right). For the $\mathcal{R}_1$-constraints we eliminate the arguments of f and use LPO comparing subterms right-to-left.

References

[AG97a] T. Arts and J. Giesl. Automatically proving termination where simplification orderings fail. In *Proceedings of the 7th International Joint Conference on the Theory and Practice of Software Development, TAPSOFT-97*, volume 1214 of *Lecture Notes in Computer Science*, pages 261–272, Lille, France, 1997. Springer Verlag, Berlin.

[AG97b] T. Arts and J. Giesl. Proving innermost normalisation automatically. In *Proceedings of the 8th International Conference on Rewriting Techniques and Applications, RTA-97*, volume 1232 of *Lecture Notes in Computer Science*, pages 157–171, Sitges, Spain, 1997. Springer Verlag, Berlin.

[AG98] T. Arts and J. Giesl. Modularity of termination using dependency pairs. In *Proceedings of the 9th International Conference on Rewriting Techniques and Applications, RTA-98*, volume 1379 of *Lecture Notes in Computer Science*, pages 226–240, Tsukuba, Japan, 1998. Springer Verlag, Berlin.

[AG00] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

[Art00] T. Arts. System description: The dependency pair method. In *Proceedings of the 11th International Conference on Rewriting Techniques and Applications, RTA-00*, volume 1833 of *Lecture Notes in Computer Science*, pages 261–264, Norwich, England, 2000. Springer Verlag, Berlin.

[Bac87] L. Bachmair. *Proof methods for equational theories*. PhD thesis, University of Illinois, Urbana-Champaign (Illinois), 1987.

[BL88] F. Bellegarde and P. Lescanne. Termination proofs based on transformation techniques. Technical report, Centre de Recherche en Informatique de Nancy, Nancy, France, 1988.

[BL90] F. Bellegarde and P. Lescanne. Termination by completion. *Applicable Algebra in Engineering, Communication and Computing*, 1:79–96, 1990.

[BM79] R. S. Boyer and J S. Moore. *A Computational Logic*. Academic Press, 1979.

[BN98] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[CiM99] CiME 2, 1999. Pre-release available at http://www.lri.fr/~demons/cime-2.0.html.

[Der79] N. Dershowitz. A note on simplification orderings. *Information Processing Letters*, 9(5):212–215, 1979.

[Der87] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1-2):69–116, 1987.

[Der93] N. Dershowitz. 33 examples of termination. In *Term Rewriting, Proceedings Spring School of Theoretical Computer Science*, volume 909 of *Lecture Notes in Computer Science*, pages 16–27, Font Romeux, France, 1993. Springer Verlag, Berlin.

[DH95] N. Dershowitz and C. Hoot. Natural termination. *Theoretical Computer Science*, 142(2):179–207, 1995.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, pages 243–320. North-Holland, 1990.

[Dro89] K. Drosten. *Termersetzungssysteme: Grundlagen der Prototyp-Generierung algebraischer Spezifikationen*. Springer Verlag, Berlin, Berlin, 1989.

[Fer95] M. Ferreira. *Termination of Term Rewriting, Well-foundedness, Totality and Transformations*. PhD thesis, Utrecht University, PO Box 80.089, 3508 TB Utrecht, The Netherlands, 1995.

[FZ93] M. Ferreira and H. Zantema. Total termination of term rewriting. In *Proceedings of the 5th Conference on Rewrite Techniques and Applications, RTA-93*, volume 690 of *Lecture Notes in Computer Science*, pages 213–227, Montreal, Canada, 1993. Springer Verlag, Berlin.

[FZ95] M. Ferreira and H. Zantema. Dummy elimination: Making termination easier. In *Proceedings of the 10th International Conference on Fundamentals of Computation Theory, FCT-95*, volume 965 of *Lecture Notes in Computer Science*, pages 243–252, Dresden, Germany, 1995. Springer Verlag, Berlin.

[GA01] J. Giesl and T. Arts. Verification of Erlang processes by dependency pairs. *Applicable Algebra in Engineering, Communication, and Computing*, 12(1-2):39–72, 2001.

[GAO01] J. Giesl, T. Arts, and E. Ohlebusch. Modular termination proofs for rewriting using dependency pairs. Submitted to the *Journal of Symbolic Computation*, 2001.

[Gee91] M. Geerling. Termination of term rewriting systems. Master's thesis, Utrecht University, PO Box 80.089, 3508 TB Utrecht, The Netherlands, 1991.

[Ges00] A. Geser. Note on normalizing, non-terminating one-rule string rewriting systems. *Theoretical Computer Science*, 243:489–498, 2000.

[Gie95] J. Giesl. *Automatisierung von Terminierungsbeweisen für rekursiv definierte Algorithmen.* DISKI 96. Infix Verlag, 1995. Doctoral Dissertation, TH Darmstadt, Germany.

[Gie97] J. Giesl. Termination of nested and mutually recursive algorithms. *Journal of Automated Reasoning*, 19:1–29, 1997.

[GM00] J. Giesl and A. Middeldorp. Eliminating dummy elimination. In *Proceedings of the 17th International Conference on Automated Deduction, CADE-17*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 309–323, Pittsburgh, PA, USA, 2000. Springer Verlag, Berlin.

[GO00] J. Giesl and E. Ohlebusch. Pushing the frontiers of combining rewrite systems farther outwards. In *Proceedings of the Second International Workshop on Frontiers of Combining Systems, FroCoS-98*, volume 7 of *Studies in Logic and Computation*, pages 141–160, Amsterdam, The Netherlands, 2000. Research Studies Press, John Wiley & Sons.

[HH82] G. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and System Sciences*, 25:239–299, 1982.

[HL78] G. Huet and D. Lankford. On the uniform halting problem for term rewriting systems. Technical Report 283, INRIA, Le Chesnay, France, 1978.

[Klo92] J. W. Klop. Term rewriting systems. In *Background: Computational Structures*, volume 2 of *Handbook of Logic in Computer Science*, pages 1–116. Oxford University Press, New York, 1992.

[KNT99] K. Kusakari, M. Nakamura, and Y. Toyama. Argument filtering transformation. In *Proceedings of the First International Conference on Principles and Practice of Declarative Programming, PPDP-99*, volume 1702 of *Lecture Notes in Computer Science*, pages 48–62, Paris, France, 1999. Springer Verlag, Berlin.

[MA96] D. McAllester and K. Arkoudas. Walther recursion. In *Proceedings of the 13th International Conference on Automated Deduction, CADE-13*, volume 1104 of *Lecture Notes in Computer Science*, pages 643–657, New Brunswick, NJ, USA, 1996. Springer Verlag, Berlin.

[MZ97] A. Middeldorp and H. Zantema. Simple termination of rewrite systems. *Theoretical Computer Science*, 175:127–158, 1997.

[OCM00] E. Ohlebusch, C. Claves, and C. Marché. TALP: A tool for the termination analysis of logic programs. In *Proceedings of the 11th International Conference on Rewriting Techniques and Applications, RTA-00*, volume 1833 of *Lecture Notes in Computer Science*, pages 270–273, Norwich, England, 2000. Springer Verlag, Berlin.

[Ohl01] E. Ohlebusch. Termination of logic programs: transformational methods revisited. *Applicable Algebra in Engineering, Communication, and Computing*, 12(1-2):73–116, 2001.

[Pla86] D. A. Plaisted. A simple non-termination test for the Knuth-Bendix method. In *Proceedings of the 8th International Conference on Automated Deduction*, volume 230 of *Lecture Notes in Computer Science*, pages 79–88, Oxford, England, 1986. Springer Verlag, Berlin.

[Ste95a] J. Steinbach. Automatic termination proofs with transformation orderings. In *Proceedings of the 6th International Conference on Rewriting Techniques and Applications, RTA-95*, volume 914 of *Lecture Notes in Computer Science*, pages 11–25, Kaiserslautern, Germany, 1995. Springer Verlag, Berlin. Full Version appeared as Technical Report SR-92-23, Universität Kaiserslautern, Germany, 1992.

[Ste95b] J. Steinbach. Simplification orderings: History of results. *Fundamenta Informaticae*, 24:47–87, 1995.

[Toy87] Y. Toyama. Counterexamples to the termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, 1987.

[Wal91] C. Walther. *Automatisierung von Terminierungsbeweisen.* Vieweg Verlag, Braunschweig, 1991.

[Wal94] C. Walther. On proving the termination of algorithms by machine. *Artificial Intelligence*, 71(1):101–157, 1994.

[Zan95] H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

95-11 *	M. Staudt / K. von Thadden: Subsumption Checking in Knowledge Bases
95-12 *	G.V. Zemanek / H.W. Nissen / H. Hubert / M. Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology
95-13 *	M. Staudt / M. Jarke: Incremental Maintenance of Externally Materialized Views
95-14 *	P. Peters / P. Szczurko / M. Jeusfeld: Business Process Oriented Information Management: Conceptual Models at Work
95-15 *	S. Rams / M. Jarke: Proceedings of the Fifth Annual Workshop on Information Technologies & Systems
95-16 *	W. Hans / St. Winkler / F. Sáenz: Distributed Execution in Functional Logic Programming
96-1 *	Jahresbericht 1995
96-2	M. Hanus / Chr. Prehofer: Higher-Order Narrowing with Definitional Trees
96-3 *	W. Scheufele / G. Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates
96-4	K. Pohl: PRO-ART: Enabling Requirements Pre-Traceability
96-5	K. Pohl: Requirements Engineering: An Overview
96-6 *	M. Jarke / W. Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools
96-7	O. Chitil: The ς-Semantics: A Comprehensive Semantics for Functional Programs
96-8 *	S. Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics
96-9	M. Hanus (Ed.): Proceedings of the Poster Session of ALP'96 — Fifth International Conference on Algebraic and Logic Programming
96-10	R. Conradi / B. Westfechtel: Version Models for Software Configuration Management
96-11 *	C. Weise / D. Lenzkes: A Fast Decision Algorithm for Timed Refinement
96-12 *	R. Dömges / K. Pohl / M. Jarke / B. Lohmann / W. Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models
96-13 *	K. Pohl / R. Klamma / K. Weidenhaupt / R. Dömges / P. Haumer / M. Jarke: A Framework for Process-Integrated Tools

96-14 * R. Gallersdörfer / K. Klabunde / A. Stolz / M. Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996

96-15 * H. Schimpe / M. Staudt: VAREX: An Environment for Validating and Refining Rule Bases

96-16 * M. Jarke / M. Gebhardt, S. Jacobs, H. Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization

96-17 M. Jeusfeld / T. X. Bui: Decision Support Components on the Internet

96-18 M. Jeusfeld / M. Papazoglou: Information Brokering: Design, Search and Transformation

96-19 * P. Peters / M. Jarke: Simulating the impact of information flows in networked organizations

96-20 M. Jarke / P. Peters / M. Jeusfeld: Model-driven planning and design of cooperative information systems

96-21 * G. de Michelis / E. Dubois / M. Jarke / F. Matthes / J. Mylopoulos / K. Pohl / J. Schmidt / C. Woo / E. Yu: Cooperative information systems: a manifesto

96-22 * S. Jacobs / M. Gebhardt, S. Kethers, W. Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality

96-23 * M. Gebhardt / S. Jacobs: Conflict Management in Design

97-01 Jahresbericht 1996

97-02 J. Faassen: Using full parallel Boltzmann Machines for Optimization

97-03 A. Winter / A. Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems

97-04 M. Mohnen / S. Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler

97-05 * S. Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge

97-06 M. Nicola / M. Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries

97-07 P. Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen

97-08 D. Blostein / A. Schürr: Computing with Graphs and Graph Rewriting

97-09 C.-A. Krapp / B. Westfechtel: Feedback Handling in Dynamic Task Nets

97-10 M. Nicola / M. Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases

97-13 M. Mohnen: Optimising the Memory Management of Higher-Order Functional Programs

97-14 R. Baumann: Client/Server Distribution in a Structure-Oriented Database Management System

97-15 G. H. Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms

98-01 * Jahresbericht 1997

98-02 S. Gruner/ M. Nagel / A. Schürr: Fine-grained and Structure-oriented Integration Tools are Needed for Product Development Processes

98-03 S. Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

98-04 * O. Kubitz: Mobile Robots in Dynamic Environments

98-05 M. Leucker / St. Tobies: Truth — A Verification Platform for Distributed Systems

98-07 M. Arnold / M. Erdmann / M. Glinz / P. Haumer / R. Knoll / B. Paech / K. Pohl / J. Ryser / R. Studer / K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects

98-08 * H. Aust: Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen

98-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien

98-10 * M. Nicola / M. Jarke: Performance Modeling of Distributed and Replicated Databases

98-11 * A. Schleicher / B. Westfechtel / D. Jäger: Modeling Dynamic Software Processes in UML

98-12 * W. Appelt / M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web

98-13 K. Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation

99-01 * Jahresbericht 1998

99-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version

99-03 * R. Gallersdörfer / M. Jarke / M. Nicola: The ADR Replication Manager

99-04 M. Alpuente / M. Hanus / S. Lucas / G. Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing

99-07 Th. Wilke: CTL+ is exponentially more succinct than CTL

99-08 O. Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures

2000-01 * Jahresbericht 1999

2000-02 Jens Vöge / Marcin Jurdziński: A Discrete Strategy Improvement Algorithm for Solving Parity Games

2000-04 Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach

2000-05 * Mareike Schoop: Cooperative Document Management

2000-06 * Mareike Schoop, Christoph Quix (Ed.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling

2000-07 * Markus Mohnen / Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages

2000-08	Thomas Arts / Thomas Noll: Verifying Generic Erlang Client-Server Implementations
2001-01 *	Jahresbericht 2000
2001-02	Benedikt Bollig / Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03	Thierry Cachat: The power of one-letter rational languages
2001-04	Benedikt Bollig / Martin Leucker / Michael Weber: Local Parallel Model Checking for the Alternation free μ-calculus
2001-05	Benedikt Bollig / Martin Leucker / Thomas Noll: Regular MSC languages
2001-06	Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07	Martin Grohe / Stefan Wöhrle: An Existential Locality Theorem

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.