Modularity of Termination Using Dependency Pairs*

Thomas Arts[1] and Jürgen Giesl[2]

[1] Computer Science Laboratory, Ericsson Telecom AB, 126 25 Stockholm, Sweden, E-mail: thomas@cslab.ericsson.se
[2] FB Informatik, Darmstadt University of Technology, Alexanderstr. 10, 64283 Darmstadt, Germany, E-mail: giesl@informatik.th-darmstadt.de

Abstract. The framework of dependency pairs allows *automated* termination and innermost termination proofs for many TRSs where such proofs were not possible before. In this paper we present a refinement of this framework in order to prove termination in a *modular* way. Our modularity results significantly increase the class of term rewriting systems where termination resp. innermost termination can be proved automatically. Moreover, the modular approach to dependency pairs yields new modularity criteria which extend previous results in this area considerably. In particular, existing results for modularity of innermost termination can easily be obtained as direct consequences of our new criteria.

1 Introduction

Termination is one of the most important properties of a term rewriting system (TRS). While in general this problem is undecidable [HL78], several methods for proving termination have been developed (for surveys see e.g. [Der87, Ste95b, DH95]). However, most methods that are amenable to automation are restricted to the generation of *simplification orderings* and there exist numerous important TRSs whose termination cannot be proved by orderings of this restricted class.

For that reason we developed the framework of *dependency pairs* [AG96, AG97a, AG97b] which allows to apply the standard methods for termination proofs to such TRSs where they failed up to now. In this way, termination of many (also non-simply terminating) systems could be proved automatically.

When proving termination, one benefits from *modularity* results that ensure termination of the whole TRS as soon as it is proved for parts of the TRS. The aim of this paper is to refine the dependency pair approach in order to allow *modular* termination proofs using dependency pairs.

Toyama [Toy87] showed that termination is not modular for the *direct sum*, i.e. the partition of a TRS into subsystems with disjoint signatures. Barendregt and Klop (adapted by Toyama [Toy87]) and Drosten [Dro89] even gave counterexamples where the subsystems are both complete (confluent and terminating). But for TRSs of a special form termination is in fact a modular property

* Technical Report IBN 97/45, TU Darmstadt. Extended version of a paper presented at RTA '98. This work was partially supported by the Deutsche Forschungsgemeinschaft under grants no. Wa 652/7-1,2 as part of the focus program 'Deduktion'.

for direct sums [Rus87, Mid89, Gra94, TKB95, SMP95]. For a survey see e.g. [Mid90, Ohl94, Gra96a].

However, a TRS often cannot be split into subsystems with disjoint signatures. Therefore, other partitions have also been considered. In many cases it is desirable to have at least constructors in common in both parts. For the subclass of constructor systems, termination is modular provided that both parts are complete and have disjoint sets of defined symbols [MT93]. This result can also be generalized to overlay systems [Gra95]. Simple termination is modular for TRSs with shared constructors and disjoint defined symbols [KO92] and this result can be extended to composable TRSs [MZ97].

Nevertheless, in practice these results often cannot be applied for automated termination proofs. For example, many systems are hierarchical combinations of TRSs that have not only constructors in common, but where one subsystem contains defined symbols of the other subsystem. Termination is only proved modular for hierarchical combinations of several restricted forms [Der94, FJ95].

The modularity results for *innermost* termination are less restrictive than those for termination. Innermost termination is modular for direct sums and for TRSs with shared constructors [Gra95], for composable constructor systems [MT93], for composable TRSs [Ohl95], and for proper extensions [KR95], which are a special class of hierarchical combinations. As innermost termination implies termination for several classes of TRSs [Gra95, Gra96b], these results can also be used for termination proofs of such systems. In particular, this holds for locally confluent overlay systems (and in particular for non-overlapping TRSs).

In this paper we show that the modular approach using dependency pairs extends previous modularity results and we demonstrate that in our framework the existing modularity results for innermost termination of TRSs with shared constructors, for composable TRSs, and for proper extensions are obtained as easy consequences.

In Sect. 2 we present the dependency pair approach and introduce a new termination criterion which allows to use this framework in a modular way. Similarly, in Sect. 3 we present a modular approach for innermost termination proofs using dependency pairs. As shown in Sect. 4, these results imply new modularity criteria (which can also be used independent from the dependency pair technique). In Sect. 5 we give a comparison with related work and demonstrate that our results extend existing criteria for modularity of innermost termination. We conclude in Sect. 6 and give a collection of examples to demonstrate the power of our results in Sect. 7.

2 Modular Termination with Dependency Pairs

In [AG97a] we introduced the dependency pair technique to prove termination automatically. In this section we briefly recapitulate its basic concepts and present a new modular approach for automated termination proofs.

In the following, the *root* of a term $f(\ldots)$ is the leading function symbol f. For a TRS $\mathcal{R}$ with the rules R over a signature $\mathcal{F}$, $D = \{\mathrm{root}(l) | l \to r \in R\}$ is the set of the *defined symbols* and $C = \mathcal{F} \setminus D$ is the set of *constructors* of $\mathcal{R}$. To stress the

splitting of the signature we denote a TRS by $\mathcal{R}(D, C, R)$. For example consider the following TRS with the constructors s and c and the defined symbol f.

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

Most methods for automated termination proofs are restricted to *simplification orderings* [Der87, Ste95b]. Hence, these methods cannot prove termination of TRSs like the one above, as $\mathsf{f}(x, \mathsf{c}(\mathsf{s}(x)))$ can be reduced to the term $\mathsf{f}(x, \mathsf{s}(\mathsf{f}(x, \mathsf{s}(\mathsf{c}(\mathsf{s}(x))))))$ where it is embedded in.

In contrast to previous approaches we do not compare left- and right-hand sides of rules, but we only compare left-hand sides with those *subterms* that may possibly start a new reduction. Hence, we only focus on those subterms of right-hand sides which have a defined root symbol.

More precisely, if $f(s_1, \ldots, s_n)$ rewrites to $C[g(t_1, \ldots, t_m)]$ (where g is a defined symbol and C is some context), then we only compare the argument tuples $s_1, \ldots, s_n$ and $t_1, \ldots, t_m$. To avoid the handling of *tuples*, a new *tuple symbol* $F \notin \mathcal{F}$ is introduced for every defined symbol f in D. Instead of comparing *tuples*, now the *terms* $F(s_1, \ldots, s_n)$ and $G(t_1, \ldots, t_m)$ are compared. To ease readability we assume that the signature $\mathcal{F}$ consists of lower case function symbols only and denote the tuple symbols by the corresponding upper case symbols.

Definition 1 (Dependency Pair). Let $\mathcal{R}(D, C, R)$ be a term rewriting system. If $f(s_1, \ldots, s_n) \to C[g(t_1, \ldots, t_m)]$ is a rewrite rule of R with $g \in D$, then $\langle F(s_1, \ldots, s_n), G(t_1, \ldots, t_m)\rangle$ is a *dependency pair* of $\mathcal{R}$.

In the above example we obtain the following dependency pairs:

$$\langle \mathsf{F}(x, \mathsf{c}(y)), \mathsf{F}(x, \mathsf{s}(\mathsf{f}(y, y)))\rangle \tag{1}$$

$$\langle \mathsf{F}(x, \mathsf{c}(y)), \mathsf{F}(y, y)\rangle \tag{2}$$

$$\langle \mathsf{F}(\mathsf{s}(x), y), \mathsf{F}(x, \mathsf{s}(\mathsf{c}(y)))\rangle. \tag{3}$$

To trace newly introduced redeces in a reduction, we consider special sequences of dependency pairs. Here, the right-hand side of every dependency pair corresponds to the redex traced. The reductions from instantiations of the right-hand sides to instantiations of left-hand sides of consecutive dependency pairs are used to contract the arguments of redeces.

Definition 2 (Chain). Let $\mathcal{R}$ be a TRS. A sequence of dependency pairs $\langle s_1, t_1\rangle$ $\langle s_2, t_2\rangle \ldots$ 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 $\langle s_j, t_j\rangle$ and $\langle s_{j+1}, t_{j+1}\rangle$ in the sequence.

We always assume that different (occurrences of) dependency pairs have disjoint sets of variables and we always regard substitutions whose domains may be infinite. Hence, in our example we have the chain

$$\langle \mathsf{F}(x_1, \mathsf{c}(y_1)), \mathsf{F}(y_1, y_1)\rangle\ \langle \mathsf{F}(x_2, \mathsf{c}(y_2)), \mathsf{F}(y_2, y_2)\rangle\ \langle \mathsf{F}(x_3, \mathsf{c}(y_3)), \mathsf{F}(y_3, y_3)\rangle,$$

as $\mathsf{F}(y_1, y_1)\sigma \to_{\mathcal{R}}^* \mathsf{F}(x_2, \mathsf{c}(y_2))\sigma$ and $\mathsf{F}(y_2, y_2)\sigma \to_{\mathcal{R}}^* \mathsf{F}(x_3, \mathsf{c}(y_3))\sigma$ hold for the substitution σ replacing y_1 and x_2 by $\mathsf{c}(\mathsf{c}(y_3))$ and both y_2 and x_3 by $\mathsf{c}(y_3)$. In fact

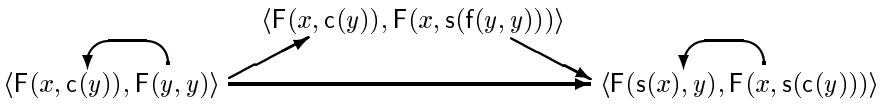

Fig. 1. The estimated dependency graph in our example.

any finite sequence of the dependency pair (2) is a chain. As proved in [AG97a], absence of infinite chains is a sufficient and necessary criterion for termination.

Theorem 3 (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. Recall that a dependency pair $\langle v, w\rangle$ may only follow $\langle s, t\rangle$ in a chain if $t\sigma$ reduces to $v\sigma$ for some substitution σ. For a term t with a constructor root symbol c, $t\sigma$ can only be reduced to terms which have the same root symbol c. If the root symbol of t is defined, then this does not give us any direct information about those terms $t\sigma$ can be reduced to. Let $\text{CAP}(t)$ result from replacing all subterms of t that have a defined root symbol by different new variables and let $\text{REN}(t)$ result from replacing all variables in t by different fresh variables. Then, to determine whether $\langle v, w\rangle$ can follow $\langle s, t\rangle$ in a chain, we check whether $\text{REN}(\text{CAP}(t))$ unifies with v. Here, the function REN is needed to rename multiple occurrences of the same variable x in t, because when instantiated with σ, two occurrences of $x\sigma$ could reduce to different terms.

So for instance we have $\text{REN}(\text{CAP}(\mathsf{F}(y, y))) = \text{REN}(\mathsf{F}(y, y)) = \mathsf{F}(y_1, y_2)$ and $\text{REN}(\text{CAP}(\mathsf{F}(x, \mathsf{s}(\mathsf{f}(y, y))))) = \text{REN}(\mathsf{F}(x, \mathsf{s}(z))) = \mathsf{F}(x_1, \mathsf{s}(z_1))$. Hence, (1) can never follow itself in a chain, because $\mathsf{F}(x_1, \mathsf{s}(z_1))$ does not unify with $\mathsf{F}(x, \mathsf{c}(y))$. To estimate which dependency pairs may occur consecutive, the *estimated dependency graph* has been introduced, cf. [AG97a].

Definition 4 (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 $\langle s, t\rangle$ to $\langle v, w\rangle$ if $\text{REN}(\text{CAP}(t))$ and v are unifiable.

In our example, we obtain the estimated dependency graph in Fig. 1. As usual, a subset $\mathcal{P}$ of dependency pairs is called a *cycle* if for any two dependency pairs $\langle s, t\rangle, \langle v, w\rangle \in \mathcal{P}$ there is a path from $\langle s, t\rangle$ to $\langle v, w\rangle$ and from $\langle v, w\rangle$ to $\langle s, t\rangle$ in the estimated dependency graph. (In particular, there must also be a path from $\langle s, t\rangle$ to itself for every $\langle s, t\rangle \in \mathcal{P}$). In our example we have two non-empty cycles, viz. $\{(2)\}$ and $\{(3)\}$.

Using the estimated dependency graph, we develop a new *modular* refinement of Thm. 3. In the following we always restrict ourselves to finite TRSs. Then any infinite chain corresponds to a cycle. Dependency pairs that do not occur on cycles (such as (1)) can be ignored. Hence, it suffices to prove that there is no infinite chain *from any cycle*.

Theorem 5 (Modular Termination Criterion). *A TRS $\mathcal{R}$ is terminating if and only if for each cycle $\mathcal{P}$ in the estimated dependency graph there exists no infinite $\mathcal{R}$-chain of dependency pairs from $\mathcal{P}$.*

Proof. The 'only if' direction is a direct consequence of Thm. 3. For the other direction, suppose that $\mathcal{R}$ is not terminating. Then by Thm. 3 there exists an infinite $\mathcal{R}$-chain. As $\mathcal{R}$ is finite, there are only finitely many dependency pairs and hence, one dependency pair occurs infinitely many times in the chain (up to renaming of the variables). Thus the infinite chain has the form $\dots \langle s\rho_1, t\rho_1\rangle \dots \langle s\rho_2, t\rho_2\rangle \dots \langle s\rho_3, t\rho_3\rangle \dots$, where $\rho_1, \rho_2, \rho_3, \dots$ are renamings. Hence, the tail $\langle s\rho_1, t\rho_1\rangle \dots \langle s\rho_2, t\rho_2\rangle \dots$ is an infinite $\mathcal{R}$-chain which consists of dependency pairs from one cycle in the estimated dependency graph only. □

By the above theorem we can prove termination of a TRS in a modular way, because the absence of infinite chains can be proved separately for every cycle.

For each cycle $\mathcal{P}$, we generate a set of inequalities such that the existence of well-founded quasi-orderings[1] $\geq_\mathcal{P}$ satisfying these inequalities is sufficient for the absence of infinite chains. For that purpose we have to ensure that the dependency pairs from $\mathcal{P}$ are decreasing w.r.t. $\geq_\mathcal{P}$. More precisely, for any sequence of dependency pairs $\langle s_1, t_1\rangle\langle s_2, t_2\rangle\langle s_3, t_3\rangle \dots$ from $\mathcal{P}$ and for any substitution σ with $t_j\sigma \to_\mathcal{R}^* s_{j+1}\sigma$ (for all j) we demand

$$s_1\sigma \geq_\mathcal{P} t_1\sigma \geq_\mathcal{P} s_2\sigma \geq_\mathcal{P} t_2\sigma \geq_\mathcal{P} s_3\sigma \geq_\mathcal{P} t_3\sigma \geq_\mathcal{P} \dots,$$

and for at least one $\langle s, t\rangle \in \mathcal{P}$ we demand the *strict* inequality $s\sigma >_\mathcal{P} t\sigma$. Then there exists no chain of dependency pairs from $\mathcal{P}$ which traverses all dependency pairs in $\mathcal{P}$ infinitely many times.

In the following we restrict ourselves to *weakly monotonic* quasi-orderings $\geq_\mathcal{P}$ where both $\geq_\mathcal{P}$ and its strict part $>_\mathcal{P}$ are *closed under substitution*. (A quasi-ordering $\geq_\mathcal{P}$ is *weakly monotonic* if $s \geq_\mathcal{P} t$ implies $f(\dots s \dots) \geq_\mathcal{P} f(\dots t \dots)$.) Then, to guarantee $t_j\sigma \geq_\mathcal{P} s_{j+1}\sigma$ whenever $t_j\sigma \to_\mathcal{R}^* s_{j+1}\sigma$ holds, it is sufficient to demand $l \geq r$ for all rewrite rules $l \to r$ of the TRS. Moreover, $s_j \geq_\mathcal{P} t_j$ resp. $s_j >_\mathcal{P} t_j$ ensures $s_j\sigma \geq_\mathcal{P} t_j\sigma$ resp. $s_j\sigma >_\mathcal{P} t_j\sigma$ for all substitutions σ.

Theorem 6 (Modular Termination Proofs). *A TRS $\mathcal{R}(D, C, R)$ is terminating if for each cycle $\mathcal{P}$ in the estimated dependency graph there exists a well-founded weakly monotonic quasi-ordering $\geq_\mathcal{P}$ where both $\geq_\mathcal{P}$ and $>_\mathcal{P}$ are closed under substitution, such that*

- *$l \geq_\mathcal{P} r$ for all rules $l \to r$ in R,*
- *$s \geq_\mathcal{P} t$ for all dependency pairs from $\mathcal{P}$, and*
- *$s >_\mathcal{P} t$ for at least one dependency pair from $\mathcal{P}$.*

Proof. Suppose there exists an infinite $\mathcal{R}$-chain of dependency pairs from a cycle $\mathcal{P}$. Without loss of generality let $\mathcal{P}$ be *minimal*, i.e. if $\mathcal{P}$ contains a cycle $\mathcal{P}'$ as proper subset, then there is no infinite chain of dependency pairs from $\mathcal{P}'$.

[1] A quasi-ordering $\geq_\mathcal{P}$ is a reflexive and transitive relation and $\geq_\mathcal{P}$ is called *well-founded* if its strict part $>_\mathcal{P}$ is well founded.

For one dependency pair $\langle s, t\rangle \in \mathcal{P}$ we have the strict inequality $s >_\mathcal{P} t$. Due to the minimality of $\mathcal{P}$, $\langle s, t\rangle$ occurs infinitely many times in the chain (up to variable renaming), i.e. the chain has the form

$$\langle v_{1,1} w_{1,1}\rangle \dots \langle v_{1,n_1} w_{1,n_1}\rangle \;\; \langle s\rho_1, t\rho_1\rangle \;\; \langle v_{2,1} w_{2,1}\rangle \dots \langle v_{2,n_2} w_{2,n_2}\rangle \;\; \langle s\rho_2, t\rho_2\rangle \dots,$$

where $\rho_1, \rho_2, \dots$ are renamings. Hence, there exists a substitution σ such that $w_{i,j}\sigma \to_\mathcal{R}^* v_{i,j+1}\sigma$, $w_{i,n_i}\sigma \to_\mathcal{R}^* s\rho_i\sigma$, and $t\rho_i\sigma \to_\mathcal{R}^* v_{i+1,1}\sigma$. As $l \geq_\mathcal{P} r$ holds for all rules of $\mathcal{R}$ and as $\geq_\mathcal{P}$ is weakly monotonic, we have $\to_\mathcal{R}^* \subseteq \geq_\mathcal{P}$. Moreover, all dependency pairs from $\mathcal{P}$ are weakly decreasing. Thus, we obtain

$$v_{1,1}\sigma \geq_\mathcal{P} w_{1,1}\sigma \geq_\mathcal{P} \dots v_{1,n_1}\sigma \geq_\mathcal{P} w_{1,n_1}\sigma \geq_\mathcal{P} s\rho_1\sigma >_\mathcal{P} t\rho_1\sigma \geq_\mathcal{P}$$
$$v_{2,1}\sigma \geq_\mathcal{P} w_{2,1}\sigma \geq_\mathcal{P} \dots v_{2,n_2}\sigma \geq_\mathcal{P} w_{2,n_2}\sigma \geq_\mathcal{P} s\rho_2\sigma >_\mathcal{P} t\rho_2\sigma \geq_\mathcal{P} \dots$$

But this is a contradiction to the well-foundedness of $>_\mathcal{P}$. Hence, no infinite chain of dependency pairs from $\mathcal{P}$ exists and by Thm. 5, $\mathcal{R}$ is terminating. □

With this theorem, termination of our example can easily be proved automatically. After computing the estimated dependency graph in Fig. 1, two quasi-orderings $\geq_1, \geq_2$ have to be generated which satisfy

$$\mathsf{f}(x, \mathsf{c}(y)) \geq_1 \mathsf{f}(x, \mathsf{s}(\mathsf{f}(y, y))) \quad (4) \qquad \mathsf{f}(x, \mathsf{c}(y)) \geq_2 \mathsf{f}(x, \mathsf{s}(\mathsf{f}(y, y))) \quad (7)$$
$$\mathsf{f}(\mathsf{s}(x), y) \geq_1 \mathsf{f}(x, \mathsf{s}(\mathsf{c}(y))) \quad (5) \qquad \mathsf{f}(\mathsf{s}(x), y) \geq_2 \mathsf{f}(x, \mathsf{s}(\mathsf{c}(y))) \quad (8)$$
$$\mathsf{F}(x, \mathsf{c}(y)) >_1 \mathsf{F}(y, y) \quad (6) \qquad \mathsf{F}(\mathsf{s}(x), y) >_2 \mathsf{F}(x, \mathsf{s}(\mathsf{c}(y))). \quad (9)$$

Note that in contrast to *direct* termination proofs, here we only need *weakly* monotonic quasi-orderings $\geq_1, \geq_2$. Hence, before synthesizing a suitable ordering some of the arguments of function symbols may be eliminated. For instance, in the inequalities (4) - (6) one may eliminate the second argument of the function symbol f. Then every term $\mathsf{f}(s, t)$ in the inequalities is replaced by $\mathsf{f}'(s)$ (where f' is a new unary function symbol). So instead of (4) we obtain the inequality $\mathsf{f}'(x) \geq_1 \mathsf{f}'(x)$. By comparing the terms resulting from this replacement (instead of the original terms) we can take advantage of the fact that f does not have to be strongly monotonic in its second argument. Now the inequalities resulting from (4) - (6) are satisfied by the lexicographic path ordering (lpo) where subterms are compared right-to-left [KL80]. For the inequalities (7) - (9) we again delete the second argument of f. Then these inequalities are also satisfied by the lpo (with the precedence $\mathsf{F} \rhd \mathsf{s}, \mathsf{F} \rhd \mathsf{c}$), but this time subterms are compared left-to-right. Note that there exist only finitely many (and only few) possibilities to eliminate arguments of function symbols. Therefore all these possibilities can be checked automatically. As path orderings like the lpo can also be generated automatically, this enables a fully automatic termination proof of our TRS, whereas a direct termination proof with simplification orderings was not possible.

So Thm. 6 allows us to use *different* quasi-orderings to prove the absence of chains for different cycles. In our example, this is essential for the termination proof, because there exists no quasi-simplification ordering satisfying *all* inequalities (4) - (9) (not even after elimination of arguments). Hence, without our modularity result, an automated termination proof for this example fails.

3 Modular Innermost Termination with Dependency Pairs

In [AG97b] we showed that the dependency pair approach can also be modified in order to verify *innermost* termination automatically. Unlike previous methods, this technique can also prove innermost termination of non-terminating systems automatically. Similar to the preceding section, our technique for innermost termination proofs can also be used in a modular way. As an example consider the following TRS (inspired by [Toy87]):

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

By applying the first f-rule to $\mathsf{f}(x,\mathsf{c}(x),\mathsf{c}(\mathsf{g}(x,\mathsf{c}(x))))$, we obtain an infinite (cycling) reduction. However, it is not an innermost reduction, because this term contains a redex $\mathsf{g}(\ldots)$ as a proper subterm. It turns out that the TRS is not terminating, but it is innermost terminating.

To develop a criterion for innermost termination similar to the termination criterion of Sect. 2, we have to restrict the notion of chains. Since we now consider innermost reductions, arguments of a redex must be in normal form before the redex is contracted. Therefore we demand that all instantiated left-hand sides $s_j\sigma$ of dependency pairs have to be normal. Moreover, the reductions of the arguments to normal forms must be innermost reductions (denoted by '$\stackrel{i}{\to}$').

Definition 7 (Innermost Chain). Let $\mathcal{R}$ be a TRS. A sequence of dependency pairs $\langle s_1,t_1\rangle\,\langle s_2,t_2\rangle\ldots$ is an *innermost $\mathcal{R}$-chain* if there exists a substitution σ, such that all $s_j\sigma$ are in normal form and $t_j\sigma \stackrel{i}{\to}_{\mathcal{R}}^{*} s_{j+1}\sigma$ holds for every two consecutive pairs $\langle s_j,t_j\rangle$ and $\langle s_{j+1},t_{j+1}\rangle$ in the sequence.

Of course, every innermost chain is also a chain, but not vice versa. In our example, we have the following dependency pairs.

$$\langle \mathsf{F}(x,\mathsf{c}(x),\mathsf{c}(y)),\mathsf{F}(y,y,\mathsf{f}(y,x,y))\rangle \tag{10}$$

$$\langle \mathsf{F}(x,\mathsf{c}(x),\mathsf{c}(y)),\mathsf{F}(y,x,y)\rangle \tag{11}$$

$$\langle \mathsf{F}(\mathsf{s}(x),y,z),\mathsf{F}(x,\mathsf{s}(\mathsf{c}(y)),\mathsf{c}(z))\rangle \tag{12}$$

The infinite sequence consisting of the dependency pair (10) is an infinite chain, but no *innermost* chain, because $\mathsf{F}(y_1,y_1,\mathsf{f}(y_1,x_1,y_1))\sigma$ can only reduce to $\mathsf{F}(x_2,\mathsf{c}(x_2),\mathsf{c}(y_2))\sigma$ for substitutions σ where $y_1\sigma$ is not a normal form. In [AG97b], we proved that 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 Sect. 2, we introduce the estimated *innermost* dependency graph to approximate whether a dependency pair $\langle v,w\rangle$ can follow $\langle s,t\rangle$ in an innermost chain. Again we 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 Sect. 2 we do not have to rename multiple occurrences of the same variable. The reason is that we restrict ourselves to normal substitutions σ, i.e. all variables x are instantiated with normal forms and therefore, occurrences of $x\sigma$ cannot be reduced. Hence, there is no arc from (10) to itself, because $\text{CAP}(\mathsf{F}(y_1, y_1, \mathsf{f}(y_1, x_1, y_1))) = \mathsf{F}(y_1, y_1, z)$ does not unify with $\mathsf{F}(x_2, \mathsf{c}(x_2), \mathsf{c}(y_2))$. Furthermore, we also demand that the most general unifier of $\text{CAP}(t)$ and v instantiates the left-hand sides s and v to normal forms.

Definition 9 (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 $\langle s, t\rangle$ to $\langle v, w\rangle$ if $\text{CAP}(t)$ and v are unifiable by a most general unifier μ such that $s\mu$ and $v\mu$ are normal forms.

In the estimated innermost dependency graph of our example, there are arcs from (11) to each dependency pair and there are arcs from (10) to (12) and from (12) to itself. Hence, the only non-empty cycles are $\{(11)\}$ and $\{(12)\}$. Analogous to Thm. 5 one can show that it suffices to prove the absence of infinite innermost chains separately for every cycle.

Theorem 10 (Modular Innermost Termination Criterion). *A TRS $\mathcal{R}$ is innermost terminating iff for each cycle $\mathcal{P}$ in the estimated innermost dependency graph there exists no infinite innermost $\mathcal{R}$-chain of dependency pairs from $\mathcal{P}$.*

To prove innermost termination in a modular way, we again generate a set of inequalities for every cycle $\mathcal{P}$ and search for a well-founded quasi-ordering $\geq_{\mathcal{P}}$ satisfying them. However, to ensure $t\sigma \geq_{\mathcal{P}} v\sigma$ whenever $t\sigma$ reduces to $v\sigma$, we do not have to demand $l \geq_{\mathcal{P}} r$ for *all* rules of the TRS any more. As we restrict ourselves to *normal* substitutions σ, not all rules are *usable* in a reduction of $t\sigma$. For example, *no* rule can be used to reduce a normal instantiation of $\mathsf{F}(y, x, y)$, because F is no defined symbol. In general, if t contains a defined symbol f, then all f-rules are *usable* and moreover, all rules that are *usable* for right-hand sides of f-rules are also *usable* for t.

Definition 11 (Usable Rules). Let $\mathcal{R}(D, C, R)$ be a TRS. For any symbol f let $\text{Rls}_R(f) = \{l \to r \in R \mid \text{root}(l) = f\}$. For any term we define the *usable rules*:

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

where $R' = R \setminus \text{Rls}_R(f)$. Moreover, for any set of dependency pairs $\mathcal{P}$ we define $\mathcal{U}_R(\mathcal{P}) = \bigcup_{\langle s,t\rangle\in\mathcal{P}} \mathcal{U}_R(t)$.

So we have $\mathcal{U}_R(\mathsf{F}(y, y, \mathsf{f}(y, x, y))) = \text{Rls}_R(\mathsf{f})$ and $\mathcal{U}_R(\{(11)\}) = \mathcal{U}_R(\{(12)\}) = \emptyset$, i.e. there are no usable rules for the cycles. Note that $\text{Rls}_R(f) = \emptyset$ for any constructor f. Now our theorem for automatic[2] modular verification of innermost termination can be proved analogous to Thm. 6.

[2] Additional refinements for the automated checking of our innermost termination criterion can be found in [AG97b].

Theorem 12 (Modular Innermost Termination Proofs). *A TRS $\mathcal{R}(D, C, R)$ is innermost terminating if for each cycle $\mathcal{P}$ in the estimated innermost dependency graph there exists a well-founded weakly monotonic quasi-ordering $\geq_\mathcal{P}$ where both $\geq_\mathcal{P}$ and $>_\mathcal{P}$ are closed under substitution, such that*

- *$l \geq_\mathcal{P} r$ for all rules $l \to r$ in $\mathcal{U}_R(\mathcal{P})$,*
- *$s \geq_\mathcal{P} t$ for all dependency pairs from $\mathcal{P}$, and*
- *$s >_\mathcal{P} t$ for at least one dependency pair from $\mathcal{P}$.*

In this way, we obtain the following constraints for our example:

$$\mathsf{F}(x, \mathsf{c}(x), \mathsf{c}(y)) >_1 \mathsf{F}(y, x, y) \qquad \mathsf{F}(\mathsf{s}(x), y, z) >_2 \mathsf{F}(x, \mathsf{s}(\mathsf{c}(y)), \mathsf{c}(z)).$$

For $>_1$ we may use the lpo comparing subterms right-to-left and for $>_2$ we may use the lpo comparing subterms left-to-right. Hence, innermost termination of this example can easily be proved automatically. Without our modularity result, this proof would not be possible, because there exists no simplification ordering satisfying *both* inequalities (not even after elimination of arguments).

4 Modularity Criteria

In this section we present two corollaries of our results from the preceding sections which are particularly useful in practice. Moreover, these corollaries also allow a comparison with existing modularity results, as will be shown in Sect. 5.

4.1 Hierarchical Combinations

A straightforward corollary of Thm. 10 and 12 can be obtained for *hierarchical combinations*. Two term rewriting systems $\mathcal{R}_0(D_0, C_0, R_0)$ and $\mathcal{R}_1(D_1, C_1, R_1)$ form a *hierarchical combination* if $D_0 \cap D_1 = C_0 \cap D_1 = \emptyset$, i.e. defined symbols of $\mathcal{R}_0$ may occur as constructors in $\mathcal{R}_1$, but not vice versa. As an example consider the following TRS. Here, nil denotes the empty list and $n{\bullet}x$ represents the insertion of a number n into a list x, where '$n{\bullet}m{\bullet}x$' abbreviates '$n{\bullet}(m{\bullet}x)$'. The function $\mathsf{sum}(x, y)$ adds all elements of x to the first element of y, i.e. $\mathsf{sum}(n_0{\bullet}n_1{\bullet}\ldots{\bullet}n_k{\bullet}\mathsf{nil}, m{\bullet}y) = (m + \sum_{i=0}^{k} n_i){\bullet}y$. The function weight computes the weighted sum, i.e. $\mathsf{weight}(n_0{\bullet}n_1{\bullet}\ldots{\bullet}n_k{\bullet}\mathsf{nil}) = n_0 + \sum_{i=1}^{k} i\, n_i$.

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

Let $\mathcal{R}_0$ consist of the three sum-rules and let $\mathcal{R}_1$ be the system consisting of the two weight-rules. Then these two systems form a hierarchical combination, where sum is a defined symbol of $\mathcal{R}_0$ and a constructor of $\mathcal{R}_1$.

Note that tuple symbols in dependency pairs of $\mathcal{R}_0$ do not occur in left-hand sides of $\mathcal{R}_1$-dependency pairs. Hence, a cycle in the estimated innermost

dependency graph either consists of $\mathcal{R}_0$-dependency pairs or of $\mathcal{R}_1$-dependency pairs only. So in our example, every cycle either contains just SUM- or just WEIGHT-dependency pairs. Thus, we obtain the following corollary.

Corollary 13 (Innermost Termination for Hierarchical Combinations). *Let $\mathcal{R}$ be the hierarchical combination of $\mathcal{R}_0(D_0, C_0, R_0)$ and $\mathcal{R}_1(D_1, C_1, R_1)$.*

(a) $\mathcal{R}$ is innermost terminating iff $\mathcal{R}_0$ is innermost terminating and there exists no infinite innermost $\mathcal{R}$-chain of $\mathcal{R}_1$-dependency pairs.

(b) $\mathcal{R}$ is innermost terminating if $\mathcal{R}_0$ is innermost terminating and if there exists a well-founded weakly monotonic quasi-ordering $\geq$ where both $\geq$ and $>$ are closed under substitution, such that for all dependency pairs $\langle s, t\rangle$ of $\mathcal{R}_1$
- *$l \geq r$ for all rules $l \to r$ in $\mathcal{U}_{R_0 \cup R_1}(t)$ and*
- *$s > t$.*

Proof. The corollary is a direct consequence of Thm. 10 and 12, since for any dependency pair $\langle s, t\rangle$ of $\mathcal{R}_0$ the only rules that can be used to reduce a normal instantiation of t are the rules from $\mathcal{R}_0$ (i.e. $\mathcal{U}_{R_0 \cup R_1}(t) \subseteq R_0$). □

(Innermost) termination of the sum-system ($\mathcal{R}_0$) is easily proved (e.g. by the lpo with the precedence $\mathsf{sum} \rhd \bullet$ and $\mathsf{sum} \rhd \mathsf{s}$). For the weight-subsystem ($\mathcal{R}_1$) we obtain the following constraints. (Note that $\langle \mathsf{WEIGHT}(\ldots), \mathsf{SUM}(\ldots)\rangle$ is no dependency pair of $\mathcal{R}_1$, since $\mathsf{sum} \notin D_1$.)

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

After eliminating the first arguments of sum and '$\bullet$', the inequalities are also satisfied by the lpo, but now we have to use the precedence $\bullet \rhd \mathsf{sum}$.

In this way, innermost termination of this example can be proved automatically. Moreover, as the system is non-overlapping, this also proves its termination. Note that this system is not simply terminating and without modularity, no quasi-simplification ordering would have satisfied the constraints resulting from the dependency pair approach (even when using elimination of arguments).

A corollary like Cor. 13 can also be formulated for termination instead of innermost termination, because in the termination case there cannot be a cycle consisting of dependency pairs from both $\mathcal{R}_0$ and $\mathcal{R}_1$ either. But in contrast to the innermost termination case, rules of $\mathcal{R}_1$ can be used to reduce instantiated right-hand sides of $\mathcal{R}_0$-dependency pairs (as we cannot restrict ourselves to normal substitutions then). Hence, to prove the absence of infinite $\mathcal{R}_0$-chains we have to use a quasi-ordering where the rules of $\mathcal{R}_1$ are also weakly decreasing. Therefore, the constraints for the termination proof of the sum and weight-example (according to Sect. 2) are not satisfied by any quasi-simplification ordering amenable to automation, whereas the constraints for *innermost* termination are fulfilled by such an ordering. Hence, for non-overlapping systems, it is always advantageous to verify termination by proving *innermost* termination only.

4.2 Splitting into Subsystems

The modularity results presented so far were all used in the context of dependency pairs. However, the classical approach to modularity is to split the TRS under consideration into subsystems and to prove (innermost) termination of these subsystems separately. The following corollary of Thm. 10 shows that the consideration of cycles in the estimated innermost dependency graph can also be used to decompose the original 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[3]. So for the innermost terminating example of Sect. 3 we have $\mathcal{O}(\{(11)\}) = \{\mathsf{f}(x, \mathsf{c}(x), \mathsf{c}(y)) \to \mathsf{f}(y, y, \mathsf{f}(y, x, y))\}$ and $\mathcal{O}(\{(12)\}) = \{\mathsf{f}(\mathsf{s}(x), y, z) \to \mathsf{f}(x, \mathsf{s}(\mathsf{c}(y)), \mathsf{c}(z))\}$.

Corollary 14 (Modularity for Subsystems). *Let $\mathcal{R}(D, C, R)$ be a TRS, let $\mathcal{P}_1, \ldots, \mathcal{P}_n$ be the cycles in its estimated innermost dependency graph, and let $\mathcal{R}_j(D_j, C_j, R_j)$ be subsystems of $\mathcal{R}$ such that $\mathcal{U}_R(\mathcal{P}_j) \cup \mathcal{O}(\mathcal{P}_j) \subseteq R_j$ (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.*

Proof. As $\mathcal{P}_j$ is a cycle, every dependency pair from $\mathcal{P}_j$ is an $\mathcal{R}_j$-dependency pair. (The reason is that for every[4] $\langle F(\mathbf{s}), G(\mathbf{t})\rangle \in \mathcal{P}_j$ there is also a dependency pair $\langle G(\mathbf{v}), H(\mathbf{w})\rangle \in \mathcal{P}_j$. Hence, g must be a defined symbol of $\mathcal{R}_j$.) Now the corollary is a direct consequence of Thm. 10, because every innermost $\mathcal{R}$-chain of dependency pairs from $\mathcal{P}_j$ is also an innermost $\mathcal{R}_j$-chain. □

For instance, in the example of Sect. 3 we only have two non-empty cycles, viz. $\{(11)\}$ and $\{(12)\}$. As these dependency pairs have no defined symbols on their right-hand sides, their sets of usable rules are empty. Hence, to prove innermost termination of the whole system, by Cor. 14 it suffices to prove innermost termination of the two one-rule subsystems $\mathsf{f}(x, \mathsf{c}(x), \mathsf{c}(y)) \to \mathsf{f}(y, y, \mathsf{f}(y, x, y))$ and $\mathsf{f}(\mathsf{s}(x), y, z) \to \mathsf{f}(x, \mathsf{s}(\mathsf{c}(y)), \mathsf{c}(z))$.

In fact, both subsystems are even terminating as can easily be proved automatically. For the first system one can use a polynomial interpretation mapping $\mathsf{f}(x, y, z)$ to $x+y+z$ and $\mathsf{c}(x)$ to $5x+1$ [Lan79]. Methods for the automated generation of polynomial orderings have for instance been developed in [Ste94, Gie95]. For the second system one can use the lpo.

Hence, the modularity criterion of Cor. 14 allows the use of well-known simplification orderings for innermost termination proofs of non-terminating systems, because it guarantees that innermost termination of the two simply terminating subsystems is sufficient for innermost termination of the original TRS.

A similar splitting is also possible for the example in Sect. 2. In particular, if we modify the TRS into a non-overlapping one

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

[3] 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.

[4] Here, $\mathbf{s}$ and $\mathbf{t}$ denote *tuples* of terms $s_1, \ldots, s_n$ and $t_1, \ldots, t_m$ respectively.

then Cor. 14 allows to conclude termination of the whole system from termination of the two one-rule subsystems. Their termination can be proved by the lpo, but for the first rule one needs the precedence $\mathsf{c} \rhd \mathsf{s}$ and $\mathsf{c} \rhd \mathsf{f}$, whereas for the second rule the precedence $\mathsf{f} \rhd \mathsf{s}$ and $\mathsf{f} \rhd \mathsf{c}$ is required. Hence, termination of this non-simply terminating example is implied by termination of its two simply terminating subsystems.

5 Comparison with Related Work

In this section we show that existing modularity results for innermost termination can be obtained as easy consequences of our criteria and that our criteria extend these previously developed results. Sect. 5.1 focuses on composable TRSs and Sect. 5.2 gives a comparison with results on hierarchical combinations.

5.1 Shared Constructors and Composable TRSs

By the framework of the previous sections we can easily prove that innermost termination is modular for composable TRSs [Ohl95] and hence also for TRSs with disjoint sets of defined symbols and shared constructors [Gra95]. Two TRSs $\mathcal{R}_0(D_0, C_0, R_0)$ and $\mathcal{R}_1(D_1, C_1, R_1)$ are *composable* if $C_0 \cap D_1 = C_1 \cap D_0 = \emptyset$ and if both systems contain all rewrite rules that define a defined symbol whenever that symbol is shared, i.e. $\{l \to r \mid \mathrm{root}(l) \in D_0 \cap D_1\} \subseteq R_0 \cap R_1$. Now Cor. 14 implies the following result of Ohlebusch [Ohl95].

Theorem 15 (Modularity for Composable TRSs). *Let $\mathcal{R}_0(D_0, C_0, R_0)$ and $\mathcal{R}_1(D_1, C_1, R_1)$ be composable TRSs. If $\mathcal{R}_0$ and $\mathcal{R}_1$ are innermost terminating, then $\mathcal{R}_0 \cup \mathcal{R}_1$ is also innermost terminating.*

Proof. Let $\langle F(\mathbf{s}), G(\mathbf{t})\rangle$ be a dependency pair of $\mathcal{R}_0 \cup \mathcal{R}_1$. If $f \in D_0$, then there exists a rule $f(\mathbf{t}) \to C[g(\mathbf{t})]$ in R_0. (This rule cannot be from $R_1 \setminus R_0$, because $\mathcal{R}_0$ and $\mathcal{R}_1$ are composable.) Hence, $g \in D_0$, because constructors of $\mathcal{R}_0$ are no defined symbols of $\mathcal{R}_1$. Similarly, $f \in D_1$ implies $g \in D_1$. So any dependency pair of $\mathcal{R}_0 \cup \mathcal{R}_1$ is an $\mathcal{R}_0$-dependency pair or an $\mathcal{R}_1$-dependency pair.

Moreover, there can only be an arc from $\langle F(\mathbf{s}), G(\mathbf{t})\rangle$ to a dependency pair of the form $\langle G(\mathbf{v}), H(\mathbf{w})\rangle$. Hence, if $\langle F(\mathbf{s}), G(\mathbf{t})\rangle$ is an $\mathcal{R}_j$-dependency pair, then $g \in D_j$ and therefore, $\langle G(\mathbf{v}), H(\mathbf{w})\rangle$ is also an $\mathcal{R}_j$-dependency pair (for $j \in \{0, 1\}$). So every cycle $\mathcal{P}$ in the estimated innermost dependency graph of $\mathcal{R}_0 \cup \mathcal{R}_1$ either consists of $\mathcal{R}_0$-dependency pairs or of $\mathcal{R}_1$-dependency pairs only.

If $\mathcal{P}$ only contains $\mathcal{R}_0$-dependency pairs, then R_0 is a superset of $\mathcal{U}_{R_0 \cup R_1}(\mathcal{P}) \cup \mathcal{O}(\mathcal{P})$, as the defined symbols of $\mathcal{R}_1 \setminus \mathcal{R}_0$ do not occur as constructors in $\mathcal{R}_0$. Similarly, for a set $\mathcal{P}$ of $\mathcal{R}_1$-dependency pairs, we have $\mathcal{U}_{R_0 \cup R_1}(\mathcal{P}) \cup \mathcal{O}(\mathcal{P}) \subseteq R_1$. Hence by Cor. 14, $\mathcal{R}_0 \cup \mathcal{R}_1$ is innermost terminating if $\mathcal{R}_0$ and $\mathcal{R}_1$ are innermost terminating. □

So this modularity criterion for composable TRSs is a direct consequence of Cor. 14. However, our results extend modularity to a much larger class of TRSs, e.g. they also allow a splitting into non-composable subsystems which share defined symbols as demonstrated in Sect. 4.2.

5.2 Proper Extensions

Krishna Rao [KR95] proved that innermost termination is modular for a certain form of hierarchical combinations, viz. so-called *proper* extensions. In this section we show that this is also a direct consequence of our results.

For a TRS $\mathcal{R}(D, C, R)$, the dependency relation $\succeq_d$ is the smallest quasi-ordering satisfying the condition $f \succeq_d g$ whenever there is a rewrite rule $f(\ldots) \to C[g(\ldots)] \in R$. So $f \succeq_d g$ holds if the function f depends on the definition of g.

Now the defined symbols D_1 of $\mathcal{R}_1$ are split in two sets D_1^0 and D_1^1, where D_1^0 contains all defined symbols which depend on a defined symbol of $\mathcal{R}_0$.

Definition 16 (Proper Extension, [KR95]). Let $\mathcal{R}_0(D_0, C_0, R_0)$ and $\mathcal{R}_1(D_1, C_1, R_1)$ form a hierarchical combination. The set D_1 is split into two sets

- $D_1^0 = \{f | f \in D_1, f \succeq_d g \text{ for some } g \in D_0\}$ and
- $D_1^1 = D_1 \setminus D_1^0$.

Then $\mathcal{R}_1$ is a *proper extension* of $\mathcal{R}_0$ if each rewrite rule $l \to r \in R_1$ satisfies the following condition: For every subterm t of r, if $root(t) \in D_1^0$ and $root(t) \succeq_d root(l)$, then t contains no symbols from $D_0 \cup D_1^0$ below its root position.

For instance, in the sum and weight-example from Sect. 4.1 we have $D_0 = \{\mathsf{sum}\}$, $D_1^0 = \{\mathsf{weight}\}$ (because weight depends on the definition of sum), and $D_1^1 = \emptyset$. Note that this example is no proper extension, because there is a weight-rule where the D_0-symbol sum occurs below the D_1^0-symbol weight. Thus, in a proper extension functions depending on $\mathcal{R}_0$ are never called within a recursive call of $\mathcal{R}_1$-functions. Cor. 13 and 14 imply the following result of [KR95].

Theorem 17 (Modularity for Proper Extensions). *Let $\mathcal{R}_1(D_1, C_1, R_1)$ be a proper extension of $\mathcal{R}_0(D_0, C_0, R_0)$. The TRS $\mathcal{R}_0 \cup \mathcal{R}_1$ is innermost terminating if $\mathcal{R}_0$ and $\mathcal{R}_1$ are innermost terminating.*

Proof. Similar to Cor. 13, as $\mathcal{R}_0$ and $\mathcal{R}_1$ form a hierarchical combination, every cycle in the innermost dependency graph of $\mathcal{R}_0 \cup \mathcal{R}_1$ consists solely of $\mathcal{R}_0$-dependency pairs or of $\mathcal{R}_1$-dependency pairs. If a cycle $\mathcal{P}$ consists of dependency pairs of $\mathcal{R}_0$, we have $\mathcal{U}_{R_0 \cup R_1}(\mathcal{P}) \cup \mathcal{O}(\mathcal{P}) \subseteq R_0$, because dependency pairs of $\mathcal{R}_0$ do not contain any defined symbols of $\mathcal{R}_1$.

Otherwise, the cycle $\mathcal{P}$ consists of $\mathcal{R}_1$-dependency pairs. If $\langle F(\mathbf{s}), G(\mathbf{t})\rangle$ is an $\mathcal{R}_1$-dependency pair, then there exists a rule $f(\mathbf{s}) \to C[g(\mathbf{t})]$ in R_1 and $f, g \in D_1$. In addition, we have $f \succeq_d g$ and $g \succeq_d f$ (as $\mathcal{P}$ is a cycle).

If $g \in D_1^1$, then f also belongs to D_1^1, hence no defined symbol of $D_0 \cup D_1^0$ occurs in $\mathbf{t}$. Otherwise, if $g \in D_1^0$, then by definition of a proper extension again all defined symbols in $\mathbf{t}$ are from D_1^1. Thus, in both cases, all defined symbols of $\mathcal{U}_{R_0 \cup R_1}(G(\mathbf{t}))$ belong to D_1^1. Hence, $\mathcal{U}_{R_0 \cup R_1}(G(\mathbf{t}))$ is a subsystem of $\mathcal{R}_1$.

So for any cycle $\mathcal{P}$ of $\mathcal{R}_1$-dependency pairs, we have $\mathcal{U}_{R_0 \cup R_1}(\mathcal{P}) \cup \mathcal{O}(\mathcal{P}) \subseteq R_1$. Hence, by Cor. 14 innermost termination of $\mathcal{R}_0$ and $\mathcal{R}_1$ implies innermost termination of $\mathcal{R}_0 \cup \mathcal{R}_1$. □

Thus, modularity of innermost termination for proper extensions is a consequence of Cor. 13 and 14. On the other hand, as demonstrated by the sum and weight-example, our results significantly extend the class of TRSs where innermost termination can be proved in a modular way. In particular, we can also handle hierarchical combinations where $\mathcal{R}_1$ contains defined symbols of $\mathcal{R}_0$ in the arguments of its recursive calls. Such systems occur frequently in practice.

Another modularity criterion for hierarchical combinations is due to Dershowitz [Der94]. Here, occurrences of D_0-symbols in recursive calls of D_1-symbols are allowed, but only if $\mathcal{R}_1$ is *oblivious* of the $\mathcal{R}_0$-rules, i.e. termination of $\mathcal{R}_1$ must not depend on the $\mathcal{R}_0$-rules. However, this criterion is not applicable for the sum and weight-example, because termination of the weight-rules in fact depends on the result of $\mathsf{sum}(n{\bullet}m{\bullet}x, \mathsf{0}{\bullet}x)$.

An alternative modularity result for hierarchical combinations was presented by Fernandez and Jouannaud [FJ95]. However, their result is restricted to systems where the arguments of recursive calls in $\mathcal{R}_1$ decrease w.r.t. the subterm relation (compared as multisets or lexicographically). Hence, their result is not applicable to the sum and weight-example either.

6 Conclusion

In this paper we introduced a refinement of the dependency pair approach in order to perform termination and innermost termination proofs in a modular way. This refinement allows automated termination and innermost termination proofs for many TRSs where such proofs were not possible before, cf. Sect. 7. We showed that our new modularity results extend previous results for modularity of innermost termination. Due to the framework of dependency pairs, we also obtain easy proofs for existing non-straightforward modularity theorems.

7 Examples

This section contains a collection of examples to illustrate the power of our modularity results. The following examples are TRSs, where an (innermost) termination proof without modularity is *provably 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.

But in addition, there exist numerous examples where the (innermost) termination proof with the dependency pair approach may also succeed without modularity, but where our modularity results can be used to *ease* the search for a quasi-ordering satisfying the resulting constraints. For example, to prove termination of the non-simply terminating TRS

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

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

instead of using the dependency pair approach directly, one could apply Cor. 13. The subsystem consisting of the two minus-rules is simply terminating (this can for instance be proved by the recursive path ordering (rpo)). For the quot-system we only obtain the constraints

$$\begin{aligned}\mathsf{minus}(x,\mathsf{0}) &\geq x\\ \mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) &\geq \mathsf{minus}(x,y)\\ \mathsf{QUOT}(\mathsf{s}(x),\mathsf{s}(y)) &> \mathsf{QUOT}(\mathsf{minus}(x,y),\mathsf{s}(y)).\end{aligned}$$

After eliminating the second argument of minus they are satisfied by the rpo again. A collection of more than 75 such examples where (innermost) termination can be proved automatically by the dependency pair approach can be found in [AG97c]. Most of these examples are hierarchical combinations that are no proper extensions, i.e. our modularity results can be used to ease their (innermost) termination proofs, whereas previous modularity results cannot be applied. This collection includes TRSs from different areas of computer science (e.g. arithmetical operations such as mod, gcd, logarithm, average, sorting algorithms such as minimum sort and quicksort, algorithms on graphs and trees, etc.) and several other well-known non-simply terminating TRSs from the literature (e.g. from [Der87, DH95, Ste95a]).

7.1 An Overlapping System

The following TRS is the leading example of Sect. 2.

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

The system is not simply terminating, as we have the following reduction:

$$\mathsf{f}(x,\mathsf{c}(\mathsf{s}(x))) \to_{\mathcal{R}} \mathsf{f}(x,\mathsf{s}(\mathsf{f}(\mathsf{s}(x),\mathsf{s}(x)))) \to_{\mathcal{R}} \mathsf{f}(x,\mathsf{s}(\mathsf{f}(x,\mathsf{s}(\mathsf{c}(\mathsf{s}(x)))))).$$

However, $\mathsf{f}(x,\mathsf{c}(\mathsf{s}(x)))$ is embedded in $\mathsf{f}(x,\mathsf{s}(\mathsf{f}(x,\mathsf{s}(\mathsf{c}(\mathsf{s}(x)))))).$ So termination cannot be proved directly by any simplification ordering. Hence, in order to use a quasi-simplification ordering for the (automated) termination proof, one has to apply the dependency pair approach.

However, the constraints obtained without using our modularity results include $\mathsf{F}(x,\mathsf{c}(y)) > \mathsf{F}(y,y)$ and $\mathsf{F}(\mathsf{s}(x),y) > \mathsf{F}(x,\mathsf{s}(\mathsf{c}(y)))$. Before applying techniques for the synthesis of quasi-simplification orderings, we may first eliminate arguments of function symbols. This is due to the fact that we only need a *weakly* monotonic ordering satisfying the constraints generated. However, in this example we cannot eliminate the arguments of s or c. Then no simplification ordering satisfies the above constraints, as they imply

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

But by using our modularity result of Thm. 6 we can search for two different orderings satisfying the constraints resulting from the two different (non-empty) cycles of the estimated dependency graph, cf. Fig. 1.

$$\begin{array}{ll} \mathsf{f}(x,\mathsf{c}(y)) \geq_1 \mathsf{f}(x,\mathsf{s}(\mathsf{f}(y,y))) & \mathsf{f}(x,\mathsf{c}(y)) \geq_2 \mathsf{f}(x,\mathsf{s}(\mathsf{f}(y,y))) \\ \mathsf{f}(\mathsf{s}(x),y) \geq_1 \mathsf{f}(x,\mathsf{s}(\mathsf{c}(y))) & \mathsf{f}(\mathsf{s}(x),y) \geq_2 \mathsf{f}(x,\mathsf{s}(\mathsf{c}(y))) \\ \mathsf{F}(x,\mathsf{c}(y)) >_1 \mathsf{F}(y,y) & \mathsf{F}(\mathsf{s}(x),y) >_2 \mathsf{F}(x,\mathsf{s}(\mathsf{c}(y))). \end{array}$$

After deleting the second argument of f, for $\geq_1$ we can use the lpo comparing subterms right-to-left and for $\geq_2$ we can use the lpo comparing subterms left-to-right.

Note 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. 6 for the termination proof instead.

7.2 A Non-Overlapping System

The following system is a non-overlapping variant of the preceding one, which can be obtained by replacing y in the second rule by $\mathsf{s}(y)$ (cf. Sect. 4.2).

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

Again the system is not simply terminating (we have the same reduction as in Ex. 7.1). Similar to the preceding example, an automatic termination or innermost termination proof without modularity fails, because the resulting constraints imply $\mathsf{F}(x,\mathsf{c}(\mathsf{s}(x))) > \mathsf{F}(x,\mathsf{s}(\mathsf{c}(\mathsf{s}(x))))$, which is not satisfied by any simplification ordering.

In this example, we obtain the estimated dependency graph in Fig. 2 (which is identical to the estimated innermost dependency graph).

This example is non-overlapping and hence, we can prove termination by verifying innermost termination. For that purpose we may use Cor. 14. As the sets of usable rules are empty for both dependency pairs $\langle \mathsf{F}(x,\mathsf{c}(y)), \mathsf{F}(y,y)\rangle$ and $\langle \mathsf{F}(\mathsf{s}(x),\mathsf{s}(y)), \mathsf{F}(x,\mathsf{s}(\mathsf{c}(\mathsf{s}(y))))\rangle$, 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 $\mathsf{c} \rhd \mathsf{s}$ and $\mathsf{c} \rhd \mathsf{f}$. 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 $\mathsf{f} \rhd \mathsf{s}$ and $\mathsf{f} \rhd \mathsf{c}$. In this way, the two simply terminating subsystems imply termination of the whole (non-simply terminating) TRS.

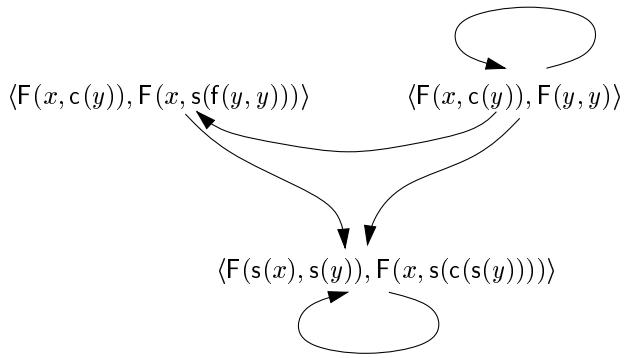

Fig. 2. The estimated (innermost) dependency graph in Ex. 7.2.

7.3 An Innermost Terminating System

The following system combines the preceding examples with the well-known example of Toyama [Toy87], cf. Sect. 3.

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

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

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

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

Similar to the preceding example, we can use Cor. 14 for the innermost termination proof. The estimated innermost dependency graph only contains two non-empty cycles consisting of $\langle \mathsf{F}(x, \mathsf{c}(x), \mathsf{c}(y)), \mathsf{F}(y, x, y)\rangle$ and $\langle \mathsf{F}(\mathsf{s}(x), y, z), \mathsf{F}(x, \mathsf{s}(\mathsf{c}(y)), \mathsf{c}(z))\rangle$ 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 $\langle \mathsf{F}(x, \mathsf{c}(x), \mathsf{c}(y)), \mathsf{F}(y, y, \mathsf{f}(y, x, y))\rangle$ 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 the lpo with the precedence $\mathsf{f} \rhd \mathsf{s}$ and $\mathsf{f} \rhd \mathsf{c}$. Hence, Cor. 14 allows us to split a non-terminating, but innermost terminating system into two simply terminating subsystems.

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

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

7.4 Sum and Weight

The following TRS computes the weighted sum of a list (see Sect. 4.1 for a detailed description).

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

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. 3.

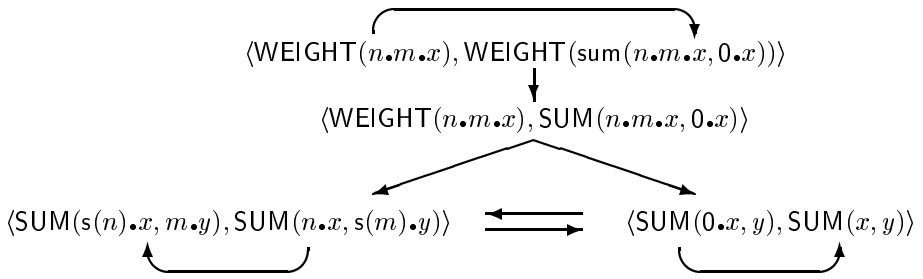

Fig. 3. The estimated (innermost) dependency graph in Ex. 7.4.

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 $\mathsf{SUM}(\mathsf{s}(n){\bullet}x, m{\bullet}y) > \mathsf{SUM}(n{\bullet}x, \mathsf{s}(m){\bullet}y)$, 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 $\mathsf{sum}(\mathsf{nil}, y) \geq y$), the constraints $\mathsf{sum}(\mathsf{s}(n){\bullet}x, m{\bullet}y) \geq \mathsf{sum}(n{\bullet}x, \mathsf{s}(m){\bullet}y)$ enforces that the first argument of sum may not be deleted either. But $\mathsf{WEIGHT}(n{\bullet}m{\bullet}x) > \mathsf{WEIGHT}(\mathsf{sum}(n{\bullet}m{\bullet}x, \ldots))$ does not hold for any quasi-simplification ordering.

Termination of the sum and weight-example can be proved by Cor. 13. The sum-subsystem ($\mathcal{R}_0$) is terminating (this can be proved by the lpo with the precedence $\mathsf{sum} \rhd \bullet$ and $\mathsf{sum} \rhd \mathsf{s}$). For the weight-subsystem ($\mathcal{R}_1$) we obtain the constraints

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

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

Note that the constraints for termination (according to Sect. 2) are not satisfied by any quasi-simplification ordering amenable to automation, i.e. this example shows that proving *innermost* termination is essentially easier than proving termination. The reason is that for the cycle consisting of $\langle \mathsf{SUM}(\mathsf{s}(n){\bullet}x, m{\bullet}y), \mathsf{SUM}(n{\bullet}x, \mathsf{s}(m){\bullet}y)\rangle$, we would obtain constraints which also demand that the weight-rules are weakly decreasing. Similar to the above argumentation (when we showed why modularity was necessary in this example), sum's first argument cannot be deleted. Due to $\mathsf{weight}(n{\bullet}\mathsf{nil}) \geq n$, the argument of weight cannot be deleted either. But then $\mathsf{weight}(n{\bullet}m{\bullet}x) \geq \mathsf{weight}(\mathsf{sum}(n{\bullet}m{\bullet}x, \mathsf{0}{\bullet}x))$ is not satisfied by any quasi-simplification ordering amenable to automation. (It is obviously not satisfied by any path ordering amenable to automation and it is not satisfied by any polynomial ordering with natural coefficients either. The reason for the latter is that sum depends on both arguments. Hence, the polynomial corresponding to $\mathsf{sum}(x, y)$ has monomials where x and y occur and therefore (for $x, y \geq 1$), $\mathsf{sum}(x, y)$ is mapped to a number which is at least as large as the *sum* of the numbers corresponding to x and to y. In particular, then the number corresponding to $\mathsf{sum}(x, y)$ is *strictly* greater than the number corresponding to x.)

7.5 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. 7.6.

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

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

$$\mathsf{G}(x, \mathsf{c}(\mathsf{s}(x))) > \mathsf{G}(x, \mathsf{g}(\mathsf{s}(\mathsf{c}(\mathsf{s}(x))), \mathsf{s}(x))) \geq \mathsf{G}(x, \mathsf{if}(\ldots, \mathsf{s}(\mathsf{c}(\mathsf{s}(x))), \ldots))$$

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. 13. Termination of $\mathcal{R}_0$ (the f- and if-rules) can for instance be verified by the rpo. For $\mathcal{R}_1$ (the g-rules) we obtain the following constraints after eliminating the arguments of s and f:

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

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

7.6 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 $\mathsf{lambda}(z,t)$.

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

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. 13 can be used for the termination proof.

Termination of $\mathcal{R}_0$ can for instance be proved by the 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:

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

A well-founded ordering satisfying these constraints can easily 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. 13 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 $\mathsf{EQ}(\mathsf{var}(l),\mathsf{var}(l')) > \mathsf{EQ}(l,l')$ the argument of var cannot be deleted. Hence, (as if's second argument cannot be deleted either), $\mathsf{ren}(\mathsf{var}(l),\mathsf{var}(k),\mathsf{var}(l')) \geq \mathsf{if}(\mathsf{eq}(l,l'),\mathsf{var}(k),\mathsf{var}(l'))$ enforces that ren must depend on its second argument. Moreover, due to $\mathsf{EQ}(t{\bullet}l,t'{\bullet}l') > \mathsf{EQ}(t,t')$, the first argument of '$\bullet$' cannot be eliminated. But the inequality

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

is not satisfied by any quasi-simplification ordering.

The simplified system of Ex. 7.5 is obtained from the subsystem

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

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).

7.7 Overlapping Variant

The following system is an overlapping TRS which is also inspired by the preceding example.

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

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 $\mathsf{G}(x, \mathsf{c}(y)) > \mathsf{G}(x, \mathsf{if}(\ldots, \ldots, \mathsf{c}(y)))$.

There is just one cycle in the estimated dependency graph which contains an F-dependency pair, viz. $\{\langle \mathsf{F}(\mathsf{s}(x)), \mathsf{F}(x)\rangle\}$. Absence of infinite chains of this dependency pair can be proved by the rpo, if the arguments of c and g are deleted. Then all rules are weakly decreasing (using the precedence $\mathsf{f} \triangleright \mathsf{true}$, $\mathsf{f} \triangleright \mathsf{false}$, $\mathsf{g}' \triangleright \mathsf{c}'$). For all other cycles one can eliminate the arguments of s, f, and if before using the rpo.

7.8 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. Functions like $\mathsf{if}_{\mathsf{min}}$ are used to encode conditions. They ensure that conditions are evaluated first (to true or to false) and that the corresponding result is evaluated afterwards. Hence, the first argument of $\mathsf{if}_{\mathsf{min}}$ is the condition that has to be tested and the other argument is the original argument of min. Further evaluation is only possible after the condition has been reduced to true or to false.

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

$$\begin{aligned}\mathsf{sort}(\mathsf{nil}) &\to \mathsf{nil}\\ \mathsf{sort}(n{\bullet}x) &\to \mathsf{min}(n{\bullet}x){\bullet}\mathsf{sort}(\mathsf{replace}(\mathsf{min}(n{\bullet}x), n, x))\end{aligned}$$

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. 13.

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 $\mathcal{R}_1$:

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

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

As we only need *weakly* monotonic orderings, before synthesizing a suitable ordering, we may first eliminate arguments of function symbols. But apart from eliminating arguments of function symbols, another possibility is to replace functions by one of their arguments. For example, instead of deleting arguments of $\mathsf{replace}$, one could substitute all terms $\mathsf{replace}(t_1, t_2, t_3)$ by the third argument t_3. In our example, a suitable elimination is given by

$$\begin{aligned}n{\bullet}x &\mapsto {\bullet}'(x)\\ \mathsf{s}(n) &\mapsto \mathsf{s}'\\ \mathsf{eq}(x, y) &\mapsto \mathsf{eq}'\\ \mathsf{le}(x, y) &\mapsto \mathsf{le}'\\ \mathsf{replace}(x, y, z) &\mapsto z\\ \mathsf{if}_{\mathsf{replace}}(b, x, y, z) &\mapsto z.\end{aligned}$$

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 the lpo or the rpo which is amenable to automation would satisfy the resulting constraints. The reason is that due to $\mathsf{EQ}(\mathsf{s}(n), \mathsf{s}(m)) > \mathsf{EQ}(n, m)$, the argument of s cannot be eliminated and hence, $\mathsf{min}(\mathsf{s}(n){\bullet}\mathsf{nil}) \geq \mathsf{s}(n)$ implies that the first argument of '$\bullet$' cannot be deleted either. Now due to $\mathsf{if}_{\mathsf{replace}}(\mathsf{true}, n, m, k{\bullet}x) \geq m{\bullet}x$, the third argument of $\mathsf{if}_{\mathsf{replace}}$ cannot be removed. Then $\mathsf{replace}(n, m, k{\bullet}x) \geq \mathsf{if}_{\mathsf{replace}}(\ldots, n, m, k{\bullet}x)$ 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. $\mathsf{replace} \trianglerighteq \mathsf{if}_{\mathsf{replace}}$. As $\mathsf{replace}$ depends on its second argument, $\mathsf{if}_{\mathsf{replace}}(\mathsf{false}, n, m, k{\bullet}x) \geq k{\bullet}\mathsf{replace}(n, m, x)$ implies $\mathsf{if}_{\mathsf{replace}} \trianglerighteq \bullet$. Hence, we have $\mathsf{replace} \trianglerighteq \bullet$. But then $\mathsf{SORT}(n{\bullet}x) > \mathsf{SORT}(\mathsf{replace}(\ldots, n, x))$ does not hold.

7.9 Different Termination Arguments, Version 1

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

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

Without modularity results, termination of this system cannot be proved by path orderings like the lpo or the 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. 14 however, it suffices to prove termination of the two one-rule subsystems. Their termination can easily be verified (e.g. by using the lpo and comparing subterms left-to-right for the first rule, whereas for the second rule they are compared right-to-left).

7.10 Different Termination Arguments, Version 2

While termination of the TRS in the preceding example 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.

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

Using Cor. 13 for the termination proof, termination of $\mathcal{R}_0$ (the f-rule) is proved with the lpo (comparing subterms left-to-right). For the $\mathcal{R}_1$-constraints we eliminate the arguments of f and use the lpo comparing subterms right-to-left.

7.11 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 modification of Ex. 7.9:

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

We obtain the following dependency pairs:

$$\langle \mathsf{F}(\mathsf{c}(\mathsf{s}(x), y), \mathsf{F}(\mathsf{c}(x, \mathsf{s}(y)))\rangle \tag{13}$$
$$\langle \mathsf{F}(\mathsf{c}(\mathsf{s}(x), \mathsf{s}(y))), \mathsf{G}(\mathsf{c}(x, y))\rangle \tag{14}$$
$$\langle \mathsf{G}(\mathsf{c}(x, \mathsf{s}(y))), \mathsf{G}(\mathsf{c}(\mathsf{s}(x), y))\rangle \tag{15}$$
$$\langle \mathsf{G}(\mathsf{c}(\mathsf{s}(x), \mathsf{s}(y))), \mathsf{F}(\mathsf{c}(x, y))\rangle \tag{16}$$

The cycles of the estimated dependency graph are $\emptyset, \{(13)\}, \{(15)\}, \{(14), (16)\}$, $\{(13), (14), (16)\}, \{(14), (15), (16)\}$, and $\{(13), (14), (15), (16)\}$. So the only maximal cycle in this example is $\{(13), (14), (15), (16)\}$. 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. 5 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 $\geq_{\mathcal{P}}$ 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 $\{(13)\}$, $\mathsf{c}(x, y)$ is mapped to x, whereas for the cycle $\{(15)\}$ 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.

7.12 Different Eliminations, Version 1

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

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

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

But termination can easily be proved using Cor. 13. 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 the rpo (using the precedence $0 \rhd \mathsf{f}$).

A similar termination proof is also possible for the TRS

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

7.13 Different Eliminations, Version 2

By adding two symmetrical rules, the TRS of Ex. 7.12 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

$$\langle \mathsf{F}(\mathsf{1}), \mathsf{F}(\mathsf{g}(\mathsf{1})) \rangle \tag{17}$$
$$\langle \mathsf{F}(\mathsf{1}), \mathsf{G}(\mathsf{1}) \rangle \tag{18}$$
$$\langle \mathsf{F}(\mathsf{f}(x)), \mathsf{F}(x) \rangle \tag{19}$$
$$\langle \mathsf{G}(\mathsf{0}), \mathsf{G}(\mathsf{f}(\mathsf{0})) \rangle \tag{20}$$
$$\langle \mathsf{G}(\mathsf{0}), \mathsf{F}(\mathsf{0}) \rangle \tag{21}$$
$$\langle \mathsf{G}(\mathsf{g}(x)), \mathsf{G}(x) \rangle. \tag{22}$$

The non-empty cycles are $\{(17)\}, \{(19)\}, \{(17), (19)\}, \{(20)\}, \{(22)\}, \{(20), (22)\}$. For the constraints resulting from the first three cycles (according to Thm. 6 or 12) we eliminate the arguments of g, whereas for the last three cycles we eliminate the arguments of f. Then the constraints are satisfied by the rpo.

References

[AG96] T. Arts & J. Giesl, Termination of constructor systems. In *Proc. RTA-96*, LNCS 1103, New Brunswick, NJ, 1996.

[AG97a] T. Arts & J. Giesl, Automatically proving termination where simplification orderings fail. In *Proc. TAPSOFT '97*, LNCS 1214, Lille, France, 1997.

[AG97b] T. Arts & J. Giesl, Proving innermost normalisation automatically. In *Proc. RTA-97*, LNCS 1232, Sitges, Spain, 1997.

[AG97c] T. Arts and J. Giesl, Termination of term rewriting using dependency pairs. Technical Report IBN 97/46, TU Darmstadt, Germany, 1997. `http://kirmes.inferenzsysteme.informatik.th-darmstadt.de/~reports/notes/ibn-97-46.ps.gz`

[Art96] T. Arts, Termination by absence of infinite chains of dependency pairs. In *Proc. CAAP '96*, LNCS 1059, Linköping, Sweden, 1996.

[Art97] T. Arts, *Automatically proving termination and innermost normalisation of term rewriting systems.* PhD Thesis, Utrecht Univ., The Netherlands, 1997.

[Der87] N. Dershowitz, Termination of rewriting. *J. Symb. Comp.*, 3:69–116, 1987.

[Der94] N. Dershowitz, *Hierarchical Termination.* In *Proc. CTRS-94*, LNCS 968, Jerusalem, Israel, 1994.

[DH95] N. Dershowitz & C. Hoot, Natural termination. *TCS*, 142(2):179–207, 1995.

[Dro89] K. Drosten, *Termersetzungssysteme.* Springer, Berlin, 1989.

[FJ95] M. Fernández & J.-P. Jouannaud, Modular termination of term rewriting systems revisited. In *Proc. 10th Workshop on Specification of Abstract Data Types*, LNCS 906, S. Margherita, Italy, 1995.

[Gie95] J. Giesl, Generating polynomial orderings for termination proofs. In *Proc. RTA-95*, LNCS 914, Kaiserslautern, Germany, 1995.

[Gra94] B. Gramlich, Generalized sufficient conditions for modular termination of rewriting. *Appl. Algebra in Engineering, Comm. & Comp.*, 5:131–158, 1994.

[Gra95] B. Gramlich, Abstract relations between restricted termination and confluence properties of rewrite systems. *Fundamenta Informaticae*, 24:3–23, 1995.

[Gra96a] B. Gramlich, *Termination and confluence properties of structured rewrite systems.* PhD Thesis, Universität Kaiserslautern, Germany, 1996.

[Gra96b] B. Gramlich, On proving termination by innermost termination. In *Proc. RTA-96*, LNCS 1103, New Brunswick, NJ, 1996.

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

[KL80] S. Kamin and J.-J. Levy, Two generalizations of the recursive path ordering. Department of Computer Science, University of Illinois, IL, 1980.

[KR95] M. R. K. Krishna Rao, Modular proofs for completeness of hierarchical term rewriting systems. *Theoretical Computer Science*, 151:487–512, 1995.

[KO92] M. Kurihara & A. Ohuchi, Modularity of simple termination of term rewriting systems with shared constructors. *Theor. Comp. Sc.*, 103:273–282, 1992.

[Lan79] D. S. Lankford, On proving term rewriting systems are noetherian. Technical Report Memo MTP-3, Louisiana Tech. University, Ruston, LA, 1979.

[MA96] D. McAllester and K. Arkoudas, Walther recursion. In *Proc. CADE-13*, LNCS 1104, New Brunswick, NJ, 1996.

[Mid89] A. Middeldorp, A sufficient condition for the termination of the direct sum of term rewriting systems. In *Proc. LICS '89*, Pacific Grove, CA, 1989.

[Mid90] A. Middeldorp, *Modular properties of term rewriting systems.* PhD Thesis, Free University Amsterdam, The Netherlands, 1990.

[MT93] A. Middeldorp & Y. Toyama, Completeness of combinations of constructor systems. *Journal of Symbolic Computation*, 15:331–348, 1993.

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

[Ohl94] E. Ohlebusch, On the modularity of termination of term rewriting systems. *Theoretical Computer Science*, 136:333–360, 1994.

[Ohl95] E. Ohlebusch, Modular properties of composable term rewriting systems. *Journal of Symbolic Computation*, 1:1–42, 1995.

[Rus87] M. Rusinowitch, On termination of the direct sum of term-rewriting systems. *Information Processing Letters*, 26:65–70, 1987.

[SMP95] M. Schmidt–Schauß, M. Marchiori, & S. E. Panitz, Modular termination of r-consistent and left-linear term rewriting systems. *TCS*, 149:361–374, 1995.

[Ste94] J. Steinbach, Generating polynomial orderings. *Information Processing Letters*, 49:85–93, 1994.

[Ste95a] J. Steinbach, Automatic termination proofs with transformation orderings. In *Proc. RTA-95*, LNCS 914, Kaiserslautern, Germany. Full version appeared as Technical Report SR-92-93, 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.

[TKB95] Y. Toyama, J. W. Klop, & H. P. Barendregt, Termination for direct sums of left-linear term rewriting systems. *Journal of the ACM*, 42:1275–1304, 1995.

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