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.tu-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. 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, Ste95, 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* [Art96, AG96, AG97a, AG97b, Art97] which allows to apply 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.

Although in general, termination is not modular for the *direct sum* [Toy87, Dro89, TKB95], i.e. the partition of a TRS into subsystems with disjoint signatures, this modularity property holds for TRSs of a special form [Rus87, Mid89, Gra94, TKB95, SMP95]. For a survey see e.g. [Mid90, Ohl94, Gra96a].

* Appeared in the *Proceedings of the 9th International Conference on Rewriting Techniques and Applications (RTA-98)*, Tsukuba, Japan, LNCS 1379, 1998. This work was partially supported by the Deutsche Forschungsgemeinschaft under grants no. Wa 652/7-1,2 as part of the focus program 'Deduktion'.

However, a TRS often cannot be split into subsystems with disjoint signatures. Therefore, partitions into subsystems which may at least have constructors in common have also been considered [KO92, MT93, Gra95, MZ97]. Nevertheless, in practice these results often cannot be applied for automated termination proofs, either. For example, many systems are hierarchical combinations of TRSs that do not only share constructors, 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 special 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. For example, 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 composable TRSs and proper extensions are obtained as easy consequences.

In Sect. 2 we present the dependency pair approach and introduce a new termination criterion 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 independently from the dependency pair technique). See [AG97c] for a collection of examples to demonstrate the power of these results. In Sect. 5 we give a comparison with related work and we conclude in Sect. 6.

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, Ste95]. 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 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 redexes in a reduction, we consider special sequences of dependency pairs. Here, the right-hand side of every dependency pair corresponds to the redex being traced.

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 $\mathcal{R}$-chain (resp. 'chain' for short)

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

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 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 $\textsc{cap}(t)$ result from replacing all subterms of t

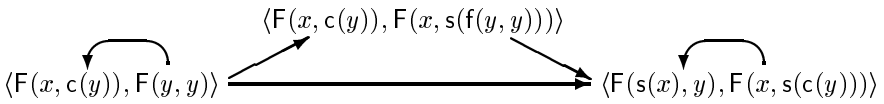

Fig. 1. The estimated dependency graph in our example.

that have a defined root symbol by different fresh 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 $\ldots \langle s\rho_1, t\rho_1\rangle \ldots \langle s\rho_2, t\rho_2\rangle \ldots \langle s\rho_3, t\rho_3\rangle \ldots$, where $\rho_1, \rho_2, \rho_3, \ldots$ are renamings. Hence, the tail $\langle s\rho_1, t\rho_1\rangle \ldots \langle s\rho_2, t\rho_2\rangle \ldots$ 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 $\langle s_1, t_1\rangle\langle s_2, t_2\rangle\langle s_3, t_3\rangle \ldots$ from $\mathcal{P}$ and for any substitution σ with $t_j\sigma \to_\mathcal{R}^* s_{j+1}\sigma$ (for all j) we demand

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

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 $f(\ldots s \ldots) \geq_\mathcal{P} f(\ldots t \ldots)$.) 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 $l \geq_\mathcal{P} r$ for all rules $l \to r$ of the TRS. Moreover, $s_j \geq_\mathcal{P} t_j$ and $s_j >_\mathcal{P} t_j$ ensure $s_j\sigma \geq_\mathcal{P} t_j\sigma$ and $s_j\sigma >_\mathcal{P} t_j\sigma$, respectively, 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 $\mathcal{P}'$.

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

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

where $\rho_1, \rho_2, \ldots$ 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

[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.

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

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, cf. [AG97a]. 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, 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 with the dependency pair approach 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. 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:

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

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 $\mathcal{P}$ of dependency pairs 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 analogously to Thm. 6.

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*

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

- $l \geq_{\mathcal{P}} r$ *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 from 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 '$\bullet$' (i.e. after replacing each term $\mathsf{sum}(s, t)$ and $s\bullet t$ by $\mathsf{sum}'(t)$ and $\bullet'(t)$, respectively), 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 the 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 a TRS into subsystems and to prove their (innermost) termination 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 a TRS into modular subsystems. In the following, let $\mathcal{O}(\mathcal{P})$ denote the *origin* of the dependency pairs in $\mathcal{P}$, i.e. $\mathcal{O}(\mathcal{P})$ is a set of those rules where the dependency pairs of $\mathcal{P}$ stem from[3]. So for the 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 $\mathcal{R}_j$.) Thus, every innermost $\mathcal{R}$-chain of dependency pairs from $\mathcal{P}_j$ is also an innermost $\mathcal{R}_j$-chain. Now the corollary is a direct consequence of Thm. 10. □

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 with the precedence $\mathsf{f} \rhd \mathsf{s}$ and $\mathsf{f} \rhd \mathsf{c}$.

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. Even better, 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. Innermost termination of the original example resp. termination of the above modified example 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.

5 Comparison with Related Work

Now we show that in the case of finite TRSs, existing modularity results for innermost termination are obtained as easy consequences of our criteria and that our criteria extend 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 immediately implies[5] 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 not 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 a cycle $\mathcal{P}$ 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 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, $\mathcal{R}_0 \cup \mathcal{R}_1$ is innermost terminating if $\mathcal{R}_0$ and $\mathcal{R}_1$ are innermost terminating. □

[5] A direct proof of Thm. 15 is not too difficult either, but our alternative proof serves to illustrate the connections between our criteria and existing modularity results.

Note that 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 for finite TRSs 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.

Let $\mathcal{R}_0(D_0, C_0, R_0)$ and $\mathcal{R}_1(D_1, C_1, R_1)$ form a hierarchical combination. 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 $\mathcal{R}_0$, i.e. $D_1^0 = \{f | f \in D_1, f \succeq_d g \text{ for some } g \in D_0\}$ 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 $D_0 \cup D_1^0$ except at the root position, cf. [KR95].

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 $D_1^1 = \emptyset$. This example is not a 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 16 (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. As in the proof of Cor. 13, since $\mathcal{R}_0$ 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 in $\mathcal{P}$, 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 $\mathcal{R}_0 \cup \mathcal{R}_1$. □

However, apart from proper extensions, we can also handle certain hierarchical combinations where $\mathcal{R}_1$ contains defined symbols of $\mathcal{R}_0$ in the arguments of its recursive calls, cf. the sum and weight-example. Such systems occur frequently in practice. Hence, our results significantly extend the class of TRSs where innermost termination can be proved in a modular way.

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. [AG97c]. 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 modularity theorems.

References

[AG96] T. Arts & J. Giesl, Termination of constructor systems. In *Proc. RTA-96*, LNCS 1103, pp. 63–77, New Brunswick, NJ, 1996.

[AG97a] T. Arts & J. Giesl, Automatically proving termination where simplification orderings fail. *TAPSOFT '97*, LNCS 1214, pp. 261–273, Lille, France, 1997.

[AG97b] T. Arts & J. Giesl, Proving innermost normalisation automatically. In *Proc. RTA-97*, LNCS 1232, pp. 157–172, Sitges, Spain, 1997.

[AG97c] T. Arts & J. Giesl, Modularity of termination using dependency pairs. Tech. Rep. IBN 97/45, TU Darmstadt, 1997. `http://www.inferenzsysteme.informatik.tu-darmstadt.de/~reports/notes/ibn-97-45.ps`

[Art96] T. Arts, Termination by absence of infinite chains of dependency pairs. In *Proc. CAAP '96*, LNCS 1059, pp. 196–210, 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. *JSC*, 3:69–116, 1987.

[Der94] N. Dershowitz, *Hierarchical Termination.* In *Proc. CTRS-94*, LNCS 968, pp. 89–105, 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, pp. 255–273, S. Margherita, Italy, 1995.

[Gie95] J. Giesl, Generating polynomial orderings for termination proofs. In *Proc. RTA-95*, LNCS 914, pp. 426–431, 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, pp. 93–107, New Brunswick, NJ, 1996.

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

[KL80] S. Kamin & 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. *TCS*, 151:487–512, 1995.

[KO92] M. Kurihara & A. Ohuchi, Modularity of simple termination of term rewriting systems with shared constructors. *TCS*, 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.

[Mid89] A. Middeldorp, A sufficient condition for the termination of the direct sum of term rewriting systems. *LICS '89*, pp. 396–401, 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. *JSC*, 15:331–348, 1993.

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

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

[Ohl95] E. Ohlebusch, Modular properties of composable term rewriting systems. *JSC*, 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.

[Ste95] 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 complete term rewriting systems. *J. ACM*, 42:1275–1304, 1995.