To appear in the Journal of Symbolic Computation

Modular Termination Proofs for Rewriting Using Dependency Pairs

JÜRGEN GIESL[1], THOMAS ARTS[2] AND ENNO OHLEBUSCH[3]

[1]*LuFG Informatik II, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany,* giesl@informatik.rwth-aachen.de

[2]*Computer Science Laboratory, Ericsson, P.O. Box 1505, 125 25 Stockholm, Sweden,* thomas@cslab.ericsson.se

[3]*Faculty of Technology, University of Bielefeld, P.O. Box 10 01 31, 33501 Bielefeld, Germany,* enno@TechFak.Uni-Bielefeld.DE

Abstract

Recently, Arts and Giesl developed the dependency pair approach which allows automated termination and innermost termination proofs for many term rewriting systems for which such proofs were not possible before. The motivation for this approach was that virtually all previous techniques for automated termination proofs of term rewriting systems were based on simplification orderings. In practice, however, many rewrite systems are not simply terminating, i.e., their termination cannot be verified by any simplification ordering.

In this article we introduce a refinement of the dependency pair framework which further extends the class of term rewriting systems for which termination or innermost termination can be shown automatically. By means of this refinement, one can now prove termination in a modular way. Thus, this refinement is inevitable in order to verify the termination of large rewrite systems occurring in practice. To be more precise, one may use several different orderings in one termination proof.

Subsequently, we present several new modularity results based on dependency pairs. First, we show that the well-known modularity of simple termination for disjoint unions can be extended to DP quasi-simple termination, i.e., to the class of rewrite systems where termination can be shown automatically by the dependency pair technique in combination with quasi-simplification orderings. Under certain additional conditions, this new result also holds for constructor-sharing and composable systems. Second, the above-mentioned refinement of the dependency pair method yields new modularity criteria for innermost termination which extend previous results in this area considerably. In particular, existing results for modularity of innermost termination can easily be shown to be direct consequences of our new criteria.

1. Introduction

In many applications of term rewriting systems (TRSs), *termination* is an important property. A TRS is said to be terminating if it does not allow infinite reductions. Since termination is in general undecidable [Huet and Lankford, 1978], several methods for proving this property have been developed; for surveys see e.g. [Dershowitz, 1987, Steinbach, 1995, Dershowitz and Hoot, 1995]. Practically all known methods that are amenable to automation use *simplification orderings* [Dershowitz, 1979, 1987, Steinbach, 1995, Middeldorp and Zantema, 1997] and in fact, even *total* orderings [Ferreira and Zantema, 1994]. However, there exist numerous important TRSs for which termination cannot be proved by this kind of orderings. For that reason, Arts and Giesl [2000] developed the so-called *dependency pair* approach. Given a TRS, the dependency pair technique automatically generates a set of constraints and the existence of a well-founded (quasi-)ordering satisfying these constraints is sufficient for termination. The advantage is that standard (automatic) techniques can often synthesize such a well-founded ordering even if a direct termination proof with the same techniques fails. In this way, simplification orderings can now be used to prove termination of non-simply terminating TRSs. Several such systems from different areas of computer science (including many challenging problems from the literature) can for instance be found in [Arts and Giesl, 2001] and applications of dependency pairs for realistic industrial problems in the area of distributed telecommunication processes are discussed in [Giesl and Arts, 2001]. For an implementation of the dependency pair approach see [Arts, 2000] or [CiME 2, 1999]. Dependency pairs have also been successfully applied in automatic termination proofs of logic programs, see [Ohlebusch et al., 2000, Ohlebusch, 2001].

After introducing required preliminaries on orderings in Section 2, in Section 3 a refinement of the dependency pair technique is presented that allows *modular* termination proofs using dependency pairs. In other words, now several well-founded relations may be used in the termination proof of one TRS. Applying the dependency pair approach in the proposed modular way cannot complicate the proof, whereas it may allow a successful application where the original technique failed. Hence, it is always advantageous, and often more powerful, to take this modular approach into account.

The above-mentioned notion of modularity is expressed in terms of dependency pairs. Therefore, it differs slightly from the conventional notion, where a property φ of TRSs (like termination) is called *modular* if whenever $\mathcal{R}_1$ and $\mathcal{R}_2$ are TRSs both satisfying φ, then their combined system $\mathcal{R}_1 \cup \mathcal{R}_2$ also satisfies φ. The knowledge that (perhaps under certain conditions) a property φ is modular provides a divide and conquer approach to establish properties of TRSs. If one wants to know whether a large TRS has a certain modular property φ, then this system can be decomposed into small subsystems and one merely has to check whether each of these subsystems has property φ. This conventional notion of modularity is inspired by a well-known paradigm in computer science; programs are developed in small modules that together form the whole program. In practice it is an enormous benefit if it suffices to prove a property of a module just once, independent of the context in which the module is used afterwards.

Clearly, this conventional notion of modularity can also be applied successfully in combination with the original dependency pair approach. However, termination and innermost termination are not modular properties for arbitrary TRSs. The modular refinement of the dependency pair approach introduced in Section 3 is applicable to numerous TRSs that do not belong to one of the restricted classes where conventional modularity results are applicable.

Toyama [1987] showed that termination is not even modular for disjoint unions, i.e., combinations of TRSs without common function symbols. So the question is what restrictions have to be imposed on the constituent TRSs so that their disjoint union is again terminating. The first results were obtained by investigating the distribution of collapsing rules and duplicating rules among the TRSs; see [Rusinowitch, 1987, Middeldorp, 1989]. In [Toyama et al., 1995] it is shown that termination is modular for confluent and left-linear TRSs. Ever since an abundance of modularity results for disjoint unions, constructor-sharing systems, composable systems, and hierarchical combinations has been published; see [Middeldorp, 1990, Ohlebusch, 1994a, Gramlich, 1996b] for an overview.

Most of the modularity results are often not applicable in practice. For example, collapsing and duplicating rules occur naturally in most TRSs. In contrast to this, since most standard methods for *automated* termination proofs are based on synthesizing simplification orderings, the result of Kurihara and Ohuchi [1992] for constructor-sharing systems is of practical relevance. They showed that the constructor-sharing combination of finite simply terminating TRSs is again simply terminating. Their result was extended to composable systems [Ohlebusch, 1995] and to certain hierarchical combinations [Krishna Rao, 1994]. Moreover, all these results also hold for infinite TRSs; see [Middeldorp and Zantema, 1997].

Thus, if one has a method to prove simple termination of a TRS, then one can use this method in a modular way for the above-mentioned classes of TRSs, whereas an arbitrary method for proving termination cannot be used in this way. However, simple termination is a considerably restricted form of termination. As indicated above, the reason for the development of the dependency pair approach was that there are numerous relevant TRSs for which simplification orderings fail in proving termination. Thus, now TRSs for which automated termination proofs are (potentially) feasible are no longer just simply terminating systems, but *DP (quasi-)simply terminating* systems, i.e., systems whose termination can be verified by using (quasi-)simplification orderings in combination with dependency pairs. Hence, a natural question is whether the current state of the art of modularity can be refined as well by extending the conventional modularity results from simple to DP (quasi-)simple termination. In Section 4 we show that this is indeed possible. Thus, the number of TRSs for which termination can be proved in a modular way is extended significantly. The practical consequence of this result is that if one has proved termination of a TRS using the dependency pair approach, then adding a TRS and proving termination of the new combination reduces to no more than proving termination of the added TRS with the dependency pair technique.

Subsequently, we consider *innermost* termination, i.e., the requirement that all reductions where only *innermost* redexes are rewritten are finite. We develop a modular technique for innermost termination proofs using dependency pairs in Section 5.

The known modularity results for innermost termination are less restrictive than those for termination. Innermost termination is modular for disjoint unions and for TRSs with shared constructors [Gramlich, 1995], for composable constructor systems [Middeldorp and Toyama, 1993], for composable TRSs [Ohlebusch, 1995], and for proper extensions [Krishna Rao, 1995], which are special hierarchical combinations. As innermost termination implies termination for several classes of TRSs [Gramlich, 1995, 1996a], 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 Section 6 we show that the modular dependency pair approach leads to new modularity criteria for innermost termination (which can also be used independently of the dependency pair technique). Moreover, we demonstrate that in our framework the known modularity results for innermost termination of composable TRSs and proper extensions are obtained as easy consequences.

Preliminary versions of parts of this article appeared in [Arts and Giesl, 1998] and [Giesl and Ohlebusch, 2000].

2. Preliminaries on Orderings

We assume the reader to be familiar with the basic notions of term rewriting. For an introduction to term rewriting see e.g. [Dershowitz and Jouannaud, 1990, Klop, 1992, Baader and Nipkow, 1998]. We restrict ourselves to finite signatures containing at least one constant (i.e., we assume that there exist *ground terms*) and to TRSs with finitely many rules. In the following we introduce the background material on orderings which is relevant to this article. A *rewrite ordering* $\succ$ over a set of terms $\mathcal{T}(\mathcal{F},\mathcal{V})$ is an ordering (i.e., an irreflexive and transitive relation) that is (strongly) monotonic (i.e., $s \succ t$ implies $f(\dots s \dots) \succ f(\dots t \dots)$ for all function symbols $f \in \mathcal{F}$) and closed under substitutions (i.e., $s \succ t$ implies $s\sigma \succ t\sigma$ for all substitutions σ). A *simplification ordering* is a rewrite ordering having the subterm property (i.e., $f(\dots x \dots) \succ x$ for all $f \in \mathcal{F}$). It is a well-known consequence of Kruskal's theorem that every simplification ordering over $\mathcal{T}(\mathcal{F},\mathcal{V})$ is well founded provided that $\mathcal{F}$ is finite.* It is also well known that simplification orderings satisfy the following property.

*For details on infinite signatures see [Middeldorp and Zantema, 1997].

LEMMA 2.1 (VARIABLES AND SIMPLIFICATION ORDERINGS): *Let $\succ$ be a simplification ordering. If $s \succ t$, then $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(s)$ and $s \notin \mathcal{V}$.*

A TRS $\mathcal{R}$ over a finite signature $\mathcal{F}$ is called *simply terminating* if its termination can be proven by a simplification ordering. This is equivalent to the statement that the TRS $\mathcal{R} \cup \mathcal{E}mb(\mathcal{F})$ is terminating, where

$$\mathcal{E}mb(\mathcal{F}) = \{f(x_1, \ldots, x_n) \to x_i \mid f \in \mathcal{F},\ f \text{ is } n\text{-ary, and } 1 \leq i \leq n\}$$

is the set of *embedding rules.*

A *quasi-rewrite ordering* $\succsim$ over a set of terms $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is a quasi-ordering (i.e., a reflexive and transitive relation) that is (weakly) monotonic (i.e., $s \succsim t$ implies $f(\ldots s \ldots) \succsim f(\ldots t \ldots)$ for all $f \in \mathcal{F}$) and closed under substitutions.

In the dependency pair method a set of inequalities is generated from a TRS $\mathcal{R}$. To prove termination of $\mathcal{R}$, one has to show that these inequalities are satisfied by some pair $(\succsim, \succ)$ consisting of a quasi-rewrite ordering $\succsim$ and an ordering $\succ$ with the properties

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

(Note that $\succ$ need not be monotonic.) Such a pair is called a *reduction pair* [Kusakari et al., 1999]. Given a quasi-rewrite ordering $\succsim$, a natural candidate for the corresponding ordering $\succ$ is the *strict* relation $\succ^s$ defined by $t \succ^s u$ if and only if $t \succsim u$ and $u \not\succsim t$. Unfortunately, $\succ^s$ is in general not closed under substitutions (see below). Therefore, to determine suitable reduction pairs automatically, one usually chooses $\succ$ to be the so-called *stable-strict* relation $\succ^{ss}$ corresponding to the quasi-rewrite ordering $\succsim$. We have $t \succ^{ss} u$ if and only if $t\sigma \succ^s u\sigma$ holds for all ground substitutions σ, where a *ground* substitution is a substitution mapping all variables to ground terms. In other words, for all those substitutions σ we must have $t\sigma \succsim u\sigma$ and $u\sigma \not\succsim t\sigma$.

For instance, many useful quasi-orderings are constructed by using mappings $|.|$ from the set of ground terms to a well-founded set like the natural numbers $\mathbb{N}$, cf. e.g. [Lankford, 1979, "polynomial orderings"]. Then $\succsim$ is defined as $t \succsim u$ if and only if $|t\sigma| \geq_{\mathbb{N}} |u\sigma|$ holds for all ground substitutions σ. A natural way to define a corresponding irreflexive ordering $\succ$ is to let $t \succ u$ hold if and only if $|t\sigma| >_{\mathbb{N}} |u\sigma|$ for all ground substitutions σ. However, now $\succ$ is not the corresponding strict relation, but the stable-strict relation corresponding to $\succsim$. Thus, the irreflexive relation intuitively associated with a quasi-ordering is often the stable-strict one instead of the strict one. In particular, if the quasi-ordering $\succsim$ is stable (i.e., closed under substitutions), then the corresponding stable-strict relation $\succ^{ss}$ is closed under substitutions too, whereas this is not necessarily true for the strict relation $\succ^s$.

For example, if $|\mathsf{0}| = 0$, $|\mathsf{s}(t)| = |t| + 1$, and $|\mathsf{f}(t)| = 2|t|$ for all ground terms t, then we have $\mathsf{f}(x) \succsim x$ and $x \not\succsim \mathsf{f}(x)$. Hence, this implies $\mathsf{f}(x) \succ^s x$. However, $\succ^s$

is not closed under substitutions because $\mathsf{f}(\mathsf{0}) \succ^s \mathsf{0}$ does not hold. This example also demonstrates that in general $\succ^s \subseteq \succ^{ss}$ is not true because for the stable-strict relation $\succ^{ss}$ we have $\mathsf{f}(x) \not\succ^{ss} x$.

Moreover, in general $\succ^{ss} \subseteq \succsim$ does not hold either (hence, $\succ^{ss} \subseteq \succ^s$ is false, too). If $\mathcal{R}$ is the TRS containing only the rule $\mathsf{f}(\mathsf{0}) \to \mathsf{0}$ and $\succsim$ is defined as $\to_{\mathcal{R}}^*$, then we have $\mathsf{f}(x) \succ^{ss} x$, but $\mathsf{f}(x) \not\succsim x$.

The following lemma states some straightforward properties of stable-strict relations.

Lemma 2.2 (Properties of Stable-Strict Relations):
Let $\succsim$ be a quasi-ordering that is closed under substitutions. Then we have

(i) $\succ^{ss}$ *is irreflexive*
(ii) $\succ^{ss}$ *is transitive*
(iii) $\succ^{ss}$ *is closed under substitutions*
(iv) if $\succsim$ *is total, then* $\succ^{ss} \subseteq \succ^s$
(v) if $\succ^s$ *is closed under substitutions, then* $\succ^s \subseteq \succ^{ss}$
(vi) if $\succ^s$ *is well founded, then* $\succ^{ss}$ *is well founded, too*
(vii) $s \succsim t \succ^{ss} u$ *implies* $s \succ^{ss} u$
(viii) $s \succ^{ss} t \succsim u$ *implies* $s \succ^{ss} u$
(ix) if $\succsim$ *is a quasi-rewrite ordering and* $\succ^s$ *is well founded, then* $(\succsim, \succ^{ss})$ *is a reduction pair*

Proof: The statements (i) and (ii) follow from the reflexivity and the transitivity of $\succsim$. Statements (iii), (iv), and (v) are direct consequences of the definition. For (vi), every potential infinite descending sequence $t_0 \succ^{ss} t_1 \succ^{ss} \ldots$ would result in an infinite descending sequence $t_0\sigma \succ^s t_1\sigma \succ^s \ldots$ Statements (vii) and (viii) follow from the transitivity and stability of $\succsim$. Statement (ix) follows from (i), (ii), (iii), (vi) and (vii) (or (viii)). □

In this article, $\succ$ always denotes an arbitrary ordering such that $(\succsim, \succ)$ forms a reduction pair. As shown in Lemma 2.2 (ix), one possibility is to choose $\succ$ to be the stable-strict relation corresponding to the quasi-rewrite relation $\succsim$ (provided that it is well founded). Lemma 2.2 (v) indicates that this choice is at least as powerful as choosing $\succ$ to be the strict relation corresponding to $\succsim$.

A *quasi-simplification ordering* (QSO) is a quasi-rewrite ordering $\succsim$ which has the (weak) subterm property (i.e., $f(\ldots x \ldots) \succsim x$ for all $f \in \mathcal{F}$). Kruskal's theorem implies that every quasi-simplification ordering over $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is well founded (more precisely, the corresponding (stable-)strict relation is well founded) provided that $\mathcal{F}$ is finite. Reduction pairs with quasi-simplification orderings satisfy a property analogous to Lemma 2.1.

Lemma 2.3 (Variables in Strict Inequalities): *Let $\succsim$ be a QSO and let $(\succsim, \succ)$ be a reduction pair. If $s \succ t$, then $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(s)$ and $s \notin \mathcal{V}$.*

Proof: Assume that there is a variable $x \in \mathcal{V}ar(t) \setminus \mathcal{V}ar(s)$. Then $t = C[x]$ for some context C. With $\sigma = \{x \mapsto s\}$ it follows that $s = s\sigma \succ t\sigma = C[s]$. Since $C[s] \succsim s$ according to the subterm property, we obtain $s \succ C[s] \succsim s$. This is a contradiction to the well-foundedness of $\succ$. Thus $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(s)$ holds. The proof of $s \notin \mathcal{V}$ is just as straightforward. □

A similar property even holds for non-strict inequalities.

Lemma 2.4 (Variables in Non-Strict Inequalities): *Let $\succsim$ be a QSO and let $(\succsim, \succ)$ be a reduction pair such that $s' \succ t'$ for some terms s', t' where $\mathcal{V}ar(t') \neq \emptyset$. If $s \succsim t$, then $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(s)$.*

Proof: First of all, $s' \succ t'$ implies $\mathcal{V}ar(t') \subseteq \mathcal{V}ar(s')$ according to Lemma 2.3. Without loss of generality, we assume that s and t are renamed such that they have no variables in common with s' or t'. We show $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(s)$ indirectly. Suppose that there is a variable $y \in \mathcal{V}ar(t) \setminus \mathcal{V}ar(s)$. Since $\mathcal{V}ar(t') \neq \emptyset$, there is a variable $x \in \mathcal{V}ar(t') \subseteq \mathcal{V}ar(s')$. Let $\sigma = \{x \mapsto s\}$ and $\sigma' = \{x \mapsto t\{y \mapsto s'\sigma\}\}$. We have (a) $s'\sigma \succ t'\sigma$ because $s' \succ t'$ and $\succ$ is closed under substitutions, (b) $t'\sigma \succsim t'\sigma'$ because $s \succsim t$ and $\succsim$ is weakly monotonic, and (c) $t'\sigma' \succsim x\sigma' \succsim s'\sigma$ because $\succsim$ has the weak subterm property and $\succsim$ is closed under substitutions. In summary, $s'\sigma \succ t'\sigma \succsim t'\sigma' \succsim s'\sigma$ is a contradiction to the well-foundedness of $\succ$. □

Examples of simplification orderings and QSOs include path orderings like the lexicographic path ordering (LPO) [Kamin and Lévy, 1980], the recursive path ordering (RPO) [Dershowitz, 1987, Steinbach, 1995, Ferreira, 1995], the Knuth-Bendix ordering (KBO) [Knuth and Bendix, 1970, Dick et al., 1990, Korovin and Voronkov, 2001], etc. Polynomial orderings, however, are not QSOs in general. For instance, if the constant $\mathsf{0}$ is associated with the number 0, $\mathsf{s}(x)$ is associated with $x + 1$, and $\mathsf{f}(x, y)$ is associated with the multiplication of x and y, then this polynomial ordering does not satisfy the subterm property (for example, $\mathsf{f}(\mathsf{s}(\mathsf{0}), \mathsf{0}) \succsim \mathsf{s}(\mathsf{0})$ does not hold). However, the following lemma shows that if the polynomial ordering respects some restrictions, then it is indeed a QSO.

Lemma 2.5 (Polynomial Orderings as QSOs): *Let $\succsim$ be a polynomial ordering where every function symbol is associated with a polynomial containing only non-negative coefficients.*

- *If every function symbol $f(x_1, \ldots, x_n)$ is associated with a polynomial which contains a (non-mixed) monomial of the form $m_i\, x_i^{k_i}$ (with $m_i, k_i \geq 1$) for every $i = 1, \ldots, n$, then $\succsim$ is a QSO.*
- *If every function symbol $f(x_1, \ldots, x_n)$ is associated with a polynomial containing all variables $x_1, \ldots, x_n$ and if every constant is associated with a number > 0, then $\succsim$ is a QSO.*

Proof: Straightforward. □

In fact, the conditions in Lemma 2.5 also entail (strong) monotonicity of the strict and stable-strict relations corresponding to the polynomial ordering.

3. Modular Termination Proofs With Dependency Pairs

Arts and Giesl [2000] introduced the dependency pair technique to prove the termination of term rewriting systems automatically. In this section we briefly recapitulate its basic concepts and present a new modular approach for automated termination proofs. We first introduce a modular termination criterion in Section 3.1 and develop an approach to check this criterion automatically in Section 3.2.

3.1. A Modular Termination Criterion

In the following we describe the notions relevant to the dependency pair method. For motivations and further refinements see [Arts and Giesl, 2000]. We adopt the notation of [Giesl and Middeldorp, 2000] and [Kusakari et al., 1999]. The *root* of a term $f(\ldots)$ is the leading function symbol f. For a TRS $\mathcal{R}$ over a signature $\mathcal{F}$, $\mathcal{D} = \{\text{root}(l) | l \to r \in \mathcal{R}\}$ is the set of the *defined symbols* and $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$ is the set of *constructors* of $\mathcal{R}$. Let $\mathcal{F}^\sharp$ denote the union of the signature $\mathcal{F}$ and $\{f^\sharp \mid f$ is a defined symbol of $\mathcal{R}\}$, where $f^\sharp$ has the same arity as f. The functions $f^\sharp$ are called *tuple symbols*. Given a term $t = f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ with f defined, we write $t^\sharp$ for the term $t = f^\sharp(t_1, \ldots, t_n)$. If $l \to r \in \mathcal{R}$ and t is a subterm of r with defined root symbol, then the rewrite rule $l^\sharp \to t^\sharp$ is called a *dependency pair* of $\mathcal{R}$. The set of all dependency pairs of $\mathcal{R}$ is denoted by $\text{DP}(\mathcal{R})$. We often write F for $f^\sharp$, etc.

For example, consider the following TRS with the constructors s and c and the defined symbol f:

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

Note that this TRS is not simply terminating as $\mathsf{f}(x, \mathsf{c}(\mathsf{s}(x)))$ can be reduced to the term $\mathsf{f}(x, \mathsf{s}(\mathsf{f}(x, \mathsf{s}(\mathsf{c}(\mathsf{s}(x))))))$ in which it is embedded. The TRS has the following dependency pairs:

$$\mathsf{F}(x, \mathsf{c}(y)) \to \mathsf{F}(x, \mathsf{s}(\mathsf{f}(y, y))) \quad (1)$$
$$\mathsf{F}(x, \mathsf{c}(y)) \to \mathsf{F}(y, y) \quad (2)$$
$$\mathsf{F}(\mathsf{s}(x), y) \to \mathsf{F}(x, \mathsf{s}(\mathsf{c}(y))) \quad (3)$$

A sequence of dependency pairs $s_1 \to t_1,\ s_2 \to t_2, \ldots$ is an $\mathcal{R}$*-chain* if there exists a substitution σ such that $t_j\sigma \to_{\mathcal{R}}^* s_{j+1}\sigma$ holds for every two consecutive pairs $s_j \to t_j$ and $s_{j+1} \to t_{j+1}$ in the sequence. We always assume that different (occurrences of) dependency pairs have disjoint sets of variables and we always consider substitutions whose domains may be infinite. In case $\mathcal{R}$ is clear from the context we often write *chain* instead of $\mathcal{R}$-chain. Hence, in our example we have the chain

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

as $\mathsf{F}(y_1, y_1)\sigma \rightarrow^*_{\mathcal{R}} \mathsf{F}(x_2, \mathsf{c}(y_2))\sigma$ and $\mathsf{F}(y_2, y_2)\sigma \rightarrow^*_{\mathcal{R}} \mathsf{F}(x_3, \mathsf{c}(y_3))\sigma$ hold for the substitution $\sigma = \{y_1 \mapsto \mathsf{c}(\mathsf{c}(y_3)), x_2 \mapsto \mathsf{c}(\mathsf{c}(y_3)), y_2 \mapsto \mathsf{c}(y_3), x_3 \mapsto \mathsf{c}(y_3)\}$. In fact any finite sequence of the dependency pair (2) is a chain. As proved by Arts and Giesl [2000], the absence of infinite chains is a sufficient and necessary criterion for termination.

Theorem 3.1 (Termination Criterion): *A TRS $\mathcal{R}$ is terminating if and only if there exists no infinite $\mathcal{R}$-chain.*

Some dependency pairs can never occur twice in any chain and hence they need not be considered when proving that no infinite chain exists. For identifying these insignificant dependency pairs, the notion of *dependency graph* has been introduced by Arts and Giesl [2000].

Definition 3.2 (Dependency graph): *The* dependency graph *of a TRS $\mathcal{R}$ is the directed graph whose nodes are the dependency pairs and there is an arc from $s \rightarrow t$ to $v \rightarrow w$ iff $s \rightarrow t$, $v \rightarrow w$ is a chain.*

The dependency graph for our example is given in Figure 1.

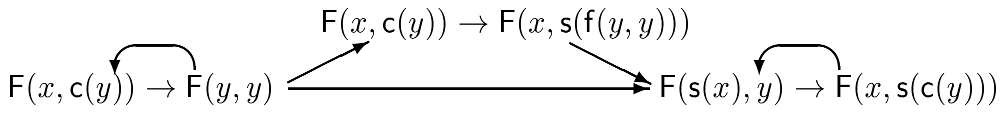
$$\begin{tikzcd}
 & \mathsf{F}(x,\mathsf{c}(y)) \rightarrow \mathsf{F}(x,\mathsf{s}(\mathsf{f}(y,y))) \arrow[dr] & \\
\mathsf{F}(x,\mathsf{c}(y)) \rightarrow \mathsf{F}(y,y) \arrow[loop above] \arrow[ur] \arrow[rr] & & \mathsf{F}(\mathsf{s}(x),y) \rightarrow \mathsf{F}(x,\mathsf{s}(\mathsf{c}(y))) \arrow[loop above]
\end{tikzcd}$$

Figure 1: Dependency graph.

A non-empty set $\mathcal{P}$ of dependency pairs is called a *cycle* if for any two pairs $s \rightarrow t$ and $v \rightarrow w$ in $\mathcal{P}$ there is a non-empty path from $s \rightarrow t$ to $v \rightarrow w$ which only traverses pairs from $\mathcal{P}$. Thus, in the example above there are two cycles, viz. $\{(2)\}$ and $\{(3)\}$. Since we restrict ourselves to finite TRSs, obviously any infinite chain corresponds to a cycle. Hence, the dependency pairs that are not on a cycle in the dependency graph are insignificant for the termination proof. In other words, in our example we may disregard the dependency pair (1).

Now we come to our first modularity result, stating that one can prove termination of a TRS in a modular way, because absence of infinite chains can be proved separately for every cycle.

Theorem 3.3 (Modular Termination Criterion): *A TRS $\mathcal{R}$ is terminating if and only if for each cycle $\mathcal{P}$ in the dependency graph there exists no infinite $\mathcal{R}$-chain of dependency pairs from $\mathcal{P}$.*

Proof: The only-if direction is a direct consequence of Theorem 3.1. For the other direction, suppose that $\mathcal{R}$ is not terminating. Then by Theorem 3.1 there exists an infinite $\mathcal{R}$-chain. As we only regard finite TRSs $\mathcal{R}$, 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

$$\ldots, s\rho_1 \to t\rho_1, \ldots, s\rho_2 \to t\rho_2, \ldots, s\rho_3 \to t\rho_3, \ldots,$$

where $\rho_1, \rho_2, \rho_3, \ldots$ are renamings. Hence, the tail $s\rho_1 \to t\rho_1, \ldots, s\rho_2 \to t\rho_2, \ldots$ is an infinite $\mathcal{R}$-chain which consists of dependency pairs from one cycle in the dependency graph only. □

According to the above theorem, in our example we can separate the proof that there is no infinite chain consisting of the dependency pair $\{(2)\}$ from the corresponding proof for the dependency pair $\{(3)\}$.

One should remark that for the soundness of this theorem one indeed has to regard *all* cycles, not just the minimal ones (i.e., not just those cycles which contain no other cycles as proper subsets). For a counterexample to illustrate this fact see [Giesl and Arts, 2001, p. 50].

Note that in standard graph terminology, a path $v_0 \Rightarrow v_1 \Rightarrow \ldots \Rightarrow v_k$ in a directed graph forms a cycle if $v_0 = v_k$ and $k \geq 1$. In our context we identify cycles with the *set* of elements that occur in it, i.e., we call $\{v_0, v_1, \ldots, v_{k-1}\}$ a cycle. Since a set never contains multiple occurrences of an element, this results in several cycling paths being identified with the same set. Moreover, for a finite TRS we only have finitely many cycles, since the number of dependency pairs is finite, too.

3.2. Checking the Modular Termination Criterion Automatically

For an automatic approach the definition of a dependency graph is impractical, since it is in general undecidable whether two dependency pairs form a chain. However, in order to obtain a sound technique for termination proofs, we can safely use any approximation of the dependency graph that preserves all its cycles. To estimate which dependency pairs may occur consecutive, the *estimated dependency graph* has been introduced, cf. [Arts and Giesl, 2000]. Let $\textsc{cap}(t)$ result from replacing all subterms of t that have a defined root symbol by different fresh variables and let $\textsc{ren}(t)$ result from replacing all variables in t by different fresh variables. Then, to determine whether $v \to w$ can follow $s \to t$ in a chain, we check whether $\textsc{ren}(\textsc{cap}(t))$ unifies with v. So we have $\textsc{ren}(\textsc{cap}(\mathsf{F}(y,y))) = \textsc{ren}(\mathsf{F}(y,y)) = \mathsf{F}(y_1,y_2)$ and $\textsc{ren}(\textsc{cap}(\mathsf{F}(x,\mathsf{s}(\mathsf{f}(y,y))))) = \textsc{ren}(\mathsf{F}(x,\mathsf{s}(z))) = \mathsf{F}(x_1,\mathsf{s}(z_1))$. Hence, (1) can never follow itself in a chain, because $\mathsf{F}(x_1,\mathsf{s}(z_1))$ does not unify with $\mathsf{F}(x,\mathsf{c}(y))$.

Definition 3.4 (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 $s \to t$ to $v \to w$ iff* $\textsc{ren}(\textsc{cap}(t))$ *and v are unifiable.*

In our example, the estimated dependency graph is the same as the dependency graph given in Figure 1. For an automation of the modular criterion of Theorem 3.3, we use this estimated dependency graph. Indeed, Theorem 3.3 also holds for the estimated dependency graph instead of the dependency graph, because all dependency pairs on a cycle in the dependency graph are also on a cycle in its estimation. The only-if direction of Theorem 3.3 holds anyway regardless of the estimation used, since whenever a TRS is terminating, then there is no infinite chain (Theorem 3.1).

To check the criterion of Theorem 3.3 automatically, for each cycle $\mathcal{P}$, we generate a set of inequalities such that the existence of reduction pairs $(\gtrsim_{\mathcal{P}},\succ_{\mathcal{P}})$ 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. $\gtrsim_{\mathcal{P}}$. More precisely, for any sequence of dependency pairs $s_1 \to t_1, s_2 \to t_2, s_3 \to t_3, \ldots$ from $\mathcal{P}$ and for any substitution σ with $t_j\sigma \to_{\mathcal{R}}^{*} s_{j+1}\sigma$ (for all j) we demand

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

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

Since $\gtrsim_{\mathcal{P}}$ is closed under substitutions and weakly monotonic, to guarantee $t_j\sigma \gtrsim_{\mathcal{P}} s_{j+1}\sigma$ whenever $t_j\sigma \to_{\mathcal{R}}^{*} s_{j+1}\sigma$ holds, it is sufficient to demand $l \gtrsim_{\mathcal{P}} r$ for all rules $l \to r$ of the TRS. Moreover, $s_j \gtrsim_{\mathcal{P}} t_j$ and $s_j \succ_{\mathcal{P}} t_j$ ensure $s_j\sigma \gtrsim_{\mathcal{P}} t_j\sigma$ and $s_j\sigma \succ_{\mathcal{P}} t_j\sigma$, respectively, for all substitutions σ.

Because rewrite rules and dependency pairs are just pairs of terms, we write $\mathcal{R} \cup \mathcal{P} \subseteq \gtrsim_{\mathcal{P}}$ as a shorthand for $l \gtrsim_{\mathcal{P}} r$ for every rewrite rule $l \to r$ in $\mathcal{R}$ and every dependency pair $l \to r$ from $\mathcal{P}$. Moreover, $\mathcal{P} \cap \succ_{\mathcal{P}} \neq \emptyset$ denotes that $l \succ_{\mathcal{P}} r$ holds for at least one dependency pair $l \to r$ from $\mathcal{P}$.

THEOREM 3.5 (MODULAR TERMINATION PROOFS): *A TRS $\mathcal{R}$ is terminating if and only if for each cycle $\mathcal{P}$ in the (estimated) dependency graph there is a reduction pair $(\gtrsim_{\mathcal{P}}, \succ_{\mathcal{P}})$ such that*

(a) $\mathcal{R} \cup \mathcal{P} \subseteq \gtrsim_{\mathcal{P}}$ and
(b) $\mathcal{P} \cap \succ_{\mathcal{P}} \neq \emptyset$.

Proof: For the if direction, suppose that there exists an infinite $\mathcal{R}$-chain of dependency pairs from a cycle $\mathcal{P}$. Without loss of generality let $\mathcal{P}$ be such that for all proper subcycles $\mathcal{P}'$ of $\mathcal{P}$, there is no infinite chain of dependency pairs from $\mathcal{P}'$.

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

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

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 $l \succsim_{\mathcal{P}} r$ holds for all rules of $\mathcal{R}$ and as $\succsim_{\mathcal{P}}$ is weakly monotonic and closed under substitutions, we have $\to_{\mathcal{R}}^* \subseteq \succsim_{\mathcal{P}}$. Moreover, all dependency pairs from $\mathcal{P}$ are weakly decreasing. Thus, we obtain

$$v_{1,1}\sigma \succsim_{\mathcal{P}} w_{1,1}\sigma \succsim_{\mathcal{P}} \ldots v_{1,n_1}\sigma \succsim_{\mathcal{P}} w_{1,n_1}\sigma \succsim_{\mathcal{P}} s\rho_1\sigma \succ_{\mathcal{P}} t\rho_1\sigma \succsim_{\mathcal{P}}$$
$$v_{2,1}\sigma \succsim_{\mathcal{P}} w_{2,1}\sigma \succsim_{\mathcal{P}} \ldots v_{2,n_2}\sigma \succsim_{\mathcal{P}} w_{2,n_2}\sigma \succsim_{\mathcal{P}} s\rho_2\sigma \succ_{\mathcal{P}} t\rho_2\sigma \succsim_{\mathcal{P}} \ldots$$

But this is a contradiction to the well-foundedness of $\succ_{\mathcal{P}}$. Hence, no infinite chain of dependency pairs from $\mathcal{P}$ exists and by Theorem 3.3, $\mathcal{R}$ is terminating.

For the only-if direction we refer to [Arts and Giesl, 2000, Theorem 7], where it is shown that termination of $\mathcal{R}$ even implies termination of $\mathcal{R} \cup \mathrm{DP}(\mathcal{R})$. A simple alternative proof for this statement using typing can be found in [Middeldorp and Ohsaki, 2000]. □

We already mentioned that for Theorem 3.3 (and hence, also for the above theorem) considering just the minimal cycles would be unsound. In fact, for Theorem 3.5 it would also be unsound just to consider *maximal* cycles (i.e., those cycles which are not contained in any other cycle). The problem is that it is not sufficient if just one dependency pair of each maximal cycle is strictly decreasing. For a counterexample to illustrate this fact see [Giesl and Arts, 2001, p. 51]. Thus, it is crucial to consider *all* cycles $\mathcal{P}$ for Theorem 3.5.

With the above theorem, termination of our example can easily be proved automatically (where for an automation of Theorem 3.5 we again use the estimated dependency graph instead of the (real) dependency graph). After computing the graph in Figure 1, two reduction pairs $(\succsim_1, \succ_1)$, $(\succsim_2, \succ_2)$ have to be generated which satisfy

$$\begin{array}{rcllrcll}
\mathsf{f}(x, \mathsf{c}(y)) & \succsim_1 & \mathsf{f}(x, \mathsf{s}(\mathsf{f}(y, y))) & (4) & \mathsf{f}(x, \mathsf{c}(y)) & \succsim_2 & \mathsf{f}(x, \mathsf{s}(\mathsf{f}(y, y))) & (7)\\
\mathsf{f}(\mathsf{s}(x), y) & \succsim_1 & \mathsf{f}(x, \mathsf{s}(\mathsf{c}(y))) & (5) & \mathsf{f}(\mathsf{s}(x), y) & \succsim_2 & \mathsf{f}(x, \mathsf{s}(\mathsf{c}(y))) & (8)\\
\mathsf{F}(x, \mathsf{c}(y)) & \succ_1 & \mathsf{F}(y, y) & (6) & \mathsf{F}(\mathsf{s}(x), y) & \succ_2 & \mathsf{F}(x, \mathsf{s}(\mathsf{c}(y))). & (9)
\end{array}$$

Of course, our aim is to use standard techniques to obtain suitable reduction pairs satisfying the constraints of Theorem 3.5. However, most existing methods generate orderings which are *strongly* monotonic, whereas for the dependency pair approach we only need a *weakly* monotonic quasi-ordering. For that reason, before synthesizing a suitable ordering, some of the arguments of the function symbols can be eliminated, cf. [Arts and Giesl, 2000]. 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 $\mathsf{f}(x) \succsim_1 \mathsf{f}(x)$. 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 (i.e., $\succsim_1$ is chosen to be $\succsim_{LPO}$ and $\succ_1$ is chosen to be the (stable-)strict relation $\succ_{LPO}$). For the inequalities (7) - (9) we again delete the second argument of f. Then these inequalities are also satisfied by LPO (with the precedence $\mathsf{F} > \mathsf{s}$, $\mathsf{F} > \mathsf{c}$), but this time subterms are compared left-to-right. Hence, termination of the TRS under consideration is proved. Note that this TRS is not simply terminating. So in the dependency pair approach, simplification orderings like LPO can be used to prove termination of TRSs for which their direct application would fail.

Apart from eliminating arguments of function symbols, another possibility is to replace functions by one of their arguments. So instead of deleting the second argument of f, one could also replace all terms $\mathsf{f}(s,t)$ by f's first argument s. Then the resulting inequalities are again satisfied by LPO. To perform this elimination of arguments resp. of function symbols the concept of argument filtering was introduced by Arts and Giesl [2000] (here we use the notation of [Kusakari et al., 1999]).

Definition 3.6 (Argument filtering): *An* argument filtering *for a signature $\mathcal{F}$ is a mapping π that associates with every n-ary function symbol an argument position $i \in \{1,\dots,n\}$ or a (possibly empty) list $[i_1,\dots,i_m]$ of argument positions with $1 \leq i_1 < \dots < i_m \leq n$. The signature $\mathcal{F}_\pi$ consists of all function symbols f such that $\pi(f) = [i_1,\dots,i_m]$, where in $\mathcal{F}_\pi$ the arity of f is m. Every argument filtering π induces a mapping from $\mathcal{T}(\mathcal{F},\mathcal{V})$ to $\mathcal{T}(\mathcal{F}_\pi,\mathcal{V})$, also denoted by π, which is defined as:*

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

As proved by Arts and Giesl [2000], in order to find a reduction pair satisfying a particular set of inequalities, one may first apply an argument filtering for the signature $\mathcal{F}^\sharp$ to the terms in the inequalities. Subsequently, one only has to find a reduction pair that satisfies these modified inequalities. In the following, for any set of rules or pairs $\mathcal{R}$ and any argument filtering π let

$$\pi(\mathcal{R}) = \{\pi(l) \to \pi(r) | l \to r \in \mathcal{R} \text{ and } \pi(l) \neq \pi(r)\}.$$

Criterion 3.7 (Modular Automated Termination Criterion):
A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is terminating if and only if for each cycle $\mathcal{P}$ in the (estimated) dependency graph there is an argument filtering $\pi_\mathcal{P}$ for $\mathcal{F}^\sharp$ and a reduction pair $(\succsim_\mathcal{P}, \succ_\mathcal{P})$ such that

(a) $\pi_\mathcal{P}(\mathcal{R} \cup \mathcal{P}) \subseteq \succsim_\mathcal{P}$ *and*
(b) $\pi_\mathcal{P}(\mathcal{P}) \cap \succ_\mathcal{P} \neq \emptyset$.

Note that there exist only finitely many possibilities for the choice of such argument filterings. Therefore in principle, all these possibilities can be checked automatically. Hence, by combining the generation of a suitable argument filtering with well-known automatic techniques for the synthesis of (*strongly* monotonic) simplification orderings, now the search for a *weakly* monotonic ordering satisfying the constraints can be automated. As mentioned before, in a reduction pair $(\succsim, \succ)$ one usually chooses $\succ$ to be the stable-strict relation corresponding to the quasi-ordering $\succsim$. By using the estimated dependency graph, this results in a fully automatic termination proof of our TRS, whereas a direct termination proof with simplification orderings was not possible. So Criterion 3.7 allows us to use *different* quasi-orderings resp. reduction pairs to prove the absence of chains for different cycles. In our example this is essential, because there exists no reduction pair with a quasi-simplification ordering satisfying *all* inequalities (4) - (9) (not even after elimination of arguments). The reason is that (9) and (6) entail

$$\mathsf{F}(\mathsf{s}(x),\mathsf{s}(x)) \succ_2 \mathsf{F}(x,\mathsf{s}(\mathsf{c}(\mathsf{s}(x)))) \to_{\mathcal{E}mb(\mathcal{F}^\sharp)} \mathsf{F}(x,\mathsf{c}(\mathsf{s}(x))) \succ_1 \mathsf{F}(\mathsf{s}(x),\mathsf{s}(x)).$$

Hence, without our modularity result, an automated termination proof with the dependency pair approach fails.

In order to synthesize suitable reduction pairs, the argument filterings should be chosen in a way such that for all resulting inequalities the variables in the right-hand side also occur in the left-hand side. Then the resulting inequalities could be transformed into a TRS as well and for proving termination of the original TRS it would be sufficient to prove termination of the transformed TRSs for all cycles.

Criterion 3.8 (Termination Criterion by Transformation): *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is terminating if and only if for each cycle $\mathcal{P}$ in the (estimated) dependency graph there is an argument filtering $\pi_\mathcal{P}$ for $\mathcal{F}^\sharp$ such that $\pi_\mathcal{P}(\mathcal{R} \cup \mathcal{P})$ is a terminating TRS and such that $\pi_\mathcal{P}(\mathcal{P}) \neq \emptyset$.*

This criterion is sufficient for termination, since one may choose $(\to^*_{\pi_\mathcal{P}(\mathcal{R}\cup\mathcal{P})}, \to^+_{\pi_\mathcal{P}(\mathcal{P})})$ as the reduction pairs in Criterion 3.7. It is also necessary for termination, because due to [Arts and Giesl, 2000, Theorem 7], termination of $\mathcal{R}$ implies termination of all $\mathcal{R}\cup\mathcal{P}$ (and hence, of $\pi_\mathcal{P}(\mathcal{R}\cup\mathcal{P})$, if $\pi_\mathcal{P}(f) = [1,\ldots,n]$ for every $f \in \mathcal{F}$ with arity n, i.e., if $\pi_\mathcal{P}$ does not filter any arguments).

4. Modularity Results for DP (Quasi-)Simple Termination

The modularity as proposed in Criteria 3.7 and 3.8 could be seen as rather method-specific. The more conventional approach of dividing the termination proof into parts is to split the TRS into subsystems and to prove termination of the subsystems separately. This, however, only works for very specific classes of TRSs. The two-rule TRS of our example can only be split in one way and no conventional modularity result exists that justifies this partitioning.

The advantage of this conventional notion of modularity is that TRSs that have been proved terminating do not have to be reconsidered after combining them with other TRSs of this kind. Thus, termination proofs never have to be redone for these combinations. Therefore, results which guarantee that termination of subsystems suffices for termination of the whole TRS are of practical interest. Based on the approach of the previous section, in this section we develop such results for the case where we use the dependency pair approach for proving termination.

More precisely, we extend the existing modularity results for simple termination to DP (quasi-)simple termination. The latter notion is formally defined in Section 4.2. Basically, a TRS is DP (quasi-)simply terminating if the constraints of Criterion 3.7 are satisfied by a suitable (quasi-)simplification ordering or if simple termination can be proved for all TRSs constructed by the transformation of Criterion 3.8, respectively.

First we briefly recall the basic notions and notations for the combination of TRSs in Section 4.1. In Section 4.3 we show that DP quasi-simple termination is modular for disjoint unions. Section 4.4 contains similar results about constructor-sharing and composable TRSs.

4.1. Basic Notions of the Union of Term Rewriting Systems

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be TRSs over the signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. Their *combined system* is the union $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ over the signature $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$. Its set of defined symbols is $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ and its set of constructors is $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$, where $\mathcal{D}_i$ ($\mathcal{C}_i$) denotes the defined symbols (constructors) in $\mathcal{R}_i$.

(1) $\mathcal{R}_1$ and $\mathcal{R}_2$ are *disjoint* if $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$.

(2) $\mathcal{R}_1$ and $\mathcal{R}_2$ are *constructor-sharing* if $\mathcal{F}_1 \cap \mathcal{F}_2 = \mathcal{C}_1 \cap \mathcal{C}_2$ ($\subseteq \mathcal{C}$).

(3) $\mathcal{R}_1$ and $\mathcal{R}_2$ are *composable* if $\mathcal{C}_1 \cap \mathcal{D}_2 = \mathcal{D}_1 \cap \mathcal{C}_2 = \emptyset$ and both systems contain all rewrite rules that define a defined symbol whenever that symbol is shared: $\{l \to r \in \mathcal{R} \mid \text{root}(l) \in \mathcal{D}_1 \cap \mathcal{D}_2\} \subseteq \mathcal{R}_1 \cap \mathcal{R}_2$.

(4) $\mathcal{R}_1$ and $\mathcal{R}_2$ form a *hierarchical combination* if $\mathcal{D}_1 \cap \mathcal{D}_2 = \mathcal{C}_1 \cap \mathcal{D}_2 = \emptyset$. So defined symbols of $\mathcal{R}_1$ may occur as constructors in $\mathcal{R}_2$, but not vice versa.

We introduce some basic notions that are helpful when reasoning about disjoint unions. Let $\Box \notin \mathcal{F}_1 \cup \mathcal{F}_2$ be a special constant. A *context* C is a term in $\mathcal{T}(\mathcal{F}_1 \cup \mathcal{F}_2 \cup \{\Box\}, \mathcal{V})$ and $C[t_1, \ldots, t_n]$ is the result of replacing from left to right the $n \geq 0$ occurrences of $\Box$ with $t_1, \ldots, t_n$. We write $t = C[\![t_1, \ldots, t_n]\!]$ if $C \in \mathcal{T}(\mathcal{F}_i \cup \{\Box\}, \mathcal{V})$, $C \neq \Box$, and $\text{root}(t_1), \ldots \text{root}(t_n) \in \mathcal{F}_{3-i}$ for some $i \in \{1, 2\}$. In this case, the t_j are the *aliens* of t and C is the topmost $\mathcal{F}_i$-homogeneous part of t, denoted by $\text{top}_i(t)$ (whereas $\text{top}_{3-i}(t)$ is $\Box$). This definition is similar to the definition of CAP where the roles of the defined symbols and the constructors are replaced by $\mathcal{F}_1$ and $\mathcal{F}_2$. Note, however, that we now use the more standard $\Box$ symbol instead of a fresh variable to replace the subterms. So for example, if $\mathcal{R}_1$ consists of the following two rules

$$\mathsf{f}(\mathsf{0},\mathsf{1},x) \to \mathsf{f}(\mathsf{s}(x),x,x) \tag{10}$$

$$\mathsf{f}(x,y,\mathsf{s}(z)) \to \mathsf{s}(\mathsf{f}(\mathsf{0},\mathsf{1},z)), \tag{11}$$

and $\mathcal{R}_2$ contains the rules

$$\mathsf{g}(x,y) \to x \tag{12}$$

$$\mathsf{g}(x,y) \to y, \tag{13}$$

then $\mathcal{R}_1$ and $\mathcal{R}_2$ are disjoint and a term like $\mathsf{f}(\mathsf{g}(\mathsf{0},\mathsf{0}),x,\mathsf{g}(y,y))$ can be written as $C[\![\mathsf{g}(\mathsf{0},\mathsf{0}),\mathsf{g}(y,y)]\!]$, where C is $\mathsf{f}(\Box,x,\Box)$. Thus $\mathrm{top}_1(\mathsf{f}(\mathsf{g}(\mathsf{0},\mathsf{0}),x,\mathsf{g}(y,y))) = \mathsf{f}(\Box,x,\Box)$ and $\mathrm{top}_2(\mathsf{f}(\mathsf{g}(\mathsf{0},\mathsf{0}),x,\mathsf{g}(y,y))) = \Box$.

Moreover, for any term t its *rank* is the maximal number of alternating function symbols (from $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively) in any path through the term, i.e.,

$$\mathrm{rank}(t) = 1 + \max\{\,\mathrm{rank}(t_j) \mid 1 \le j \le n\} \text{ where } t = C[\![t_1,\ldots,t_n]\!]$$

and $\max\emptyset = 0$. So for example we have $\mathrm{rank}(\mathsf{f}(\mathsf{g}(\mathsf{0},\mathsf{0}),x,\mathsf{g}(y,y))) = 3$. Our modularity results crucially depend on the well-known fact that $s \to_{\mathcal{R}_1\cup\mathcal{R}_2} t$ implies $\mathrm{rank}(s) \ge \mathrm{rank}(t)$.

A rewrite step $s \to_{\mathcal{R}_1\cup\mathcal{R}_2} t$ is *destructive at level 1* if $\mathrm{root}(s) \in \mathcal{F}_i$ and $\mathrm{root}(t) \in \mathcal{F}_{3-i}$ for some $i \in \{1,2\}$. A reduction step $s \to_{\mathcal{R}_1\cup\mathcal{R}_2} t$ is *destructive at level* $m+1$ (for some $m \ge 1$) if $s = C[\![s_1,\ldots,s_j,\ldots,s_n]\!] \to_{\mathcal{R}_1\cup\mathcal{R}_2} C[s_1,\ldots,t_j,\ldots,s_n] = t$ with $s_j \to_{\mathcal{R}_1\cup\mathcal{R}_2} t_j$ destructive at level m. Obviously, if a rewrite step is destructive, then the rewrite rule applied is collapsing, i.e., the right-hand side of the rule is a variable. For example, the rewrite step $\mathsf{f}(\mathsf{g}(\mathsf{0},\mathsf{0}),x,\mathsf{g}(y,y)) \to \mathsf{f}(\mathsf{0},x,\mathsf{g}(y,y))$ is destructive at level 2.

4.2. DP (Quasi-)Simple Termination

Most methods for finding well-founded orderings search for total orderings. However, we concentrate on simplification orderings or quasi-simplification orderings [Dershowitz, 1987, Steinbach, 1995, Middeldorp and Zantema, 1997] because all TRSs that are totally terminating have been shown to be simply terminating [Zantema, 1994] and because simple termination has a nice modular behaviour, whereas modularity of total termination is still an open problem.

Now we formally define the notion of DP quasi-simple termination which results from restricting ourselves to QSOs when using the dependency pair approach (i.e., when using Criterion 3.7). The motivation for this notion is that it contains all TRSs where termination can be proved *automatically* in the following way: First, the constraints described in Theorem 3.5 are generated using the estimated dependency graph, which can be determined mechanically. Then an argument filtering is applied to eliminate arguments of function symbols (or to replace functions by their arguments) as in Criterion 3.7, and finally a standard technique is used to generate a QSO $\succsim$ such that a reduction pair $(\succsim, \succ)$ satisfies the resulting constraints. For example, $\succ$ can be chosen to be the stable-strict relation corresponding to $\succsim$.

Definition 4.1 (DP quasi-simple termination): *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is called* DP quasi-simply terminating *if and only if for each cycle $\mathcal{P}$ in the estimated dependency graph there exists an argument filtering $\pi_\mathcal{P}$ for $\mathcal{F}^\sharp$ and a reduction pair $(\succsim_\mathcal{P}, \succ_\mathcal{P})$ with a QSO $\succsim_\mathcal{P}$ such that*

(a) $\pi_\mathcal{P}(\mathcal{R} \cup \mathcal{P}) \subseteq \succsim_\mathcal{P}$ *and*
(b) $\pi_\mathcal{P}(\mathcal{P}) \cap \succ_\mathcal{P} \neq \emptyset$.

Definition 4.1 captures the TRSs for which an automated termination proof using dependency pairs with the estimated dependency graph[†] is potentially feasible (since virtually all quasi-orderings that can be generated are QSOs). In fact, there are numerous DP quasi-simply terminating TRSs which are not simply terminating; cf. e.g. the collection by Arts and Giesl [2001]. This observation motivated the development of the dependency pair approach and it also motivated the work of the present section, as our aim is to extend well-known modularity results for simple termination to DP quasi-simple termination. For instance, the TRS from Section 3 is obviously DP quasi-simply terminating, because the resulting constraints are satisfied by LPO (which is a quasi-simplification ordering). Similarly, for the TRS $\mathcal{R}_1 = \{(10), (11)\}$ from Section 4.1 we obtain the following dependency pairs

$$\begin{aligned} \mathsf{F}(0,1,x) &\rightarrow \mathsf{F}(\mathsf{s}(x),x,x) \qquad (14)\\ \mathsf{F}(x,y,\mathsf{s}(z)) &\rightarrow \mathsf{F}(0,1,z). \qquad (15)\end{aligned}$$

Our estimation technique determines that the first dependency pair (14) can never follow itself in a chain, because $\mathsf{F}(\mathsf{s}(x_1),x_2,x_3)\sigma \rightarrow^*_{\mathcal{R}_1} \mathsf{F}(0,1,x_4)\sigma$ does not hold for any substitution σ. So in our example, the estimated dependency graph contains an arc from (14) to (15) and arcs from (15) to (14) and to itself. Thus, the only cycles in our example are $\{(15)\}$ and $\{(14),(15)\}$. Hence, according to Theorem 3.5, to prove the absence of infinite chains from the cycle $\{(15)\}$ we have to find a reduction pair satisfying

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

[†]Note that the notion of DP quasi-simple termination and therefore also our modularity results depend on the estimation of the dependency graph. Thus, for other approximation techniques one would have to investigate the resulting modularity properties separately.

By using the argument filtering that maps f to its third argument, these constraints are satisfied by RPO with the precedence $\mathsf{s} > \mathsf{0}$ and $\mathsf{s} > \mathsf{1}$. Similarly, (by eliminating the first two arguments of F) one can also prove the absence of infinite chains from the cycle $\{(14), (15)\}$. Hence, termination of the TRS consisting of the rules (10) and (11) is proved and (as RPO is a quasi-simplification ordering), it is DP quasi-simply terminating.

In this article, we impose a minor restriction on the argument filterings used, viz. for all cycles $\mathcal{P}$ we restrict ourselves to argument filterings $\pi_\mathcal{P}$ such that for all rules $s \to t$ in $\pi_\mathcal{P}(\mathcal{R} \cup \mathcal{P})$ both $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(s)$ and $s \notin \mathcal{V}$. This restriction ensures that the rules $\pi_\mathcal{P}(\mathcal{R} \cup \mathcal{P})$ from Criterion 3.8 indeed form a term rewriting system. According to Lemma 2.4, if there is a quasi-simplification ordering satisfying the constraints in Criterion 3.7 (i.e., in Definition 4.1) and if these constraints include at least one strict inequality with variables in its right-hand side, then $\mathcal{V}ar(\pi(r)) \subseteq \mathcal{V}ar(\pi(l))$ is always satisfied for all $l \to r$ in $\mathcal{R} \cup \mathcal{P}$. In other words, the restriction is not very severe.

In fact, in the proof of modularity of DP quasi-simple termination it is sufficient to know that for every cycle of a DP quasi-simply terminating TRS there is at least *one* argument filtering satisfying the minor restriction and a suitable QSO that prove termination. However, it is an open problem whether for every DP quasi-simply terminating TRS such an argument filtering and a suitable QSO always exist. Nevertheless, even if there were a counterexample, then the QSO satisfying the constraints must fulfill $s \gtrsim C[y] \gtrsim y$ for some term s with $y \notin \mathcal{V}ar(s)$ or $x \gtrsim t$ for a term $t \neq x$. Clearly, this is impossible for path orderings like LPO or RPO. Hence, whenever the constraints of Definition 4.1 are satisfied by such a path ordering, then the restriction on the argument filterings is fulfilled anyway. A constraint of the form $s \gtrsim y$ with $y \notin \mathcal{V}ar(s)$ cannot be satisfied by polynomial orderings either unless terms are only mapped to finitely many different numbers. Thus, the question whether DP quasi-simple termination would also be modular without the above restriction is not so important for practical termination proofs.

A straightforward way to generate a QSO $\succeq$ from a simplification ordering $\succ$ is to define $t \succeq u$ if and only if $t \succ u$ or $t = u$, where $=$ is syntactic equality. In the following, we denote the reflexive closure of a relation by underlining, i.e., $\succeq$ denotes the reflexive closure of $\succ$. By restricting ourselves to this class of QSOs, we obtain the notion of DP simple termination.

Definition 4.2 (DP simple termination): *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is called* DP simply terminating *if and only if for each cycle $\mathcal{P}$ in the estimated dependency graph there is an argument filtering $\pi_\mathcal{P}$ for $\mathcal{F}^\sharp$ and a simplification ordering $\succ_\mathcal{P}$ such that*

(a) $\pi_\mathcal{P}(\mathcal{R} \cup \mathcal{P}) \subseteq \succeq_\mathcal{P}$ *and*
(b) $\pi_\mathcal{P}(\mathcal{P}) \cap \succ_\mathcal{P} \neq \emptyset$.

Note that whenever there exist argument filterings and simplification orderings satisfying the constraints (a) and (b) of Definition 4.2, then the minor restriction on the argument filterings is satisfied according to Lemma 2.1. Due to that lemma, there is the following alternative characterization for DP simple termination (which uses Criterion 3.8 instead of Criterion 3.7).

Corollary 4.3 (Alternative Charact. of DP simple termination): *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is DP simply terminating if and only if for each cycle $\mathcal{P}$ in the estimated dependency graph there is an argument filtering $\pi_{\mathcal{P}}$ for $\mathcal{F}^\sharp$ such that $\pi_{\mathcal{P}}(\mathcal{R} \cup \mathcal{P})$ is a simply terminating TRS.*

For instance, both the TRS from Section 3 and $\mathcal{R}_1 = \{(10), (11)\}$ from Section 4.1 are already DP simply terminating, because for their termination proofs we may use quasi-simplification orderings in which only syntactically identical terms are considered to be equivalent. Moreover, it also turns out that most of the examples in [Arts and Giesl, 2001] are not only DP quasi-simply terminating but even DP simply terminating. The following lemma illustrates the connections between the different notions.

Lemma 4.4 (Characterizing DP (quasi-)simple termination): *The following implications hold: simple termination $\Rightarrow$ DP simple termination $\Rightarrow$ DP quasi-simple termination $\Rightarrow$ termination.*

Proof: The second implication holds as $\succ$ is closed under substitutions and therefore $(\succeq, \succ)$ is a reduction pair. The last implication follows from Criterion 3.7.

It remains to show the first implication. Let $\mathcal{R}$ be a simply terminating TRS over the signature $\mathcal{F}$. If $\mathcal{R}$ is simply terminating, then there exists a simplification ordering $\succ$ such that $l \succ r$ holds for all rules $l \to r$ of $\mathcal{R}$.

Let Ω be the function which replaces every tuple symbol $f^\sharp$ in a term $s \in \mathcal{T}(\mathcal{F}^\sharp, \mathcal{V})$ by its corresponding function symbol $f \in \mathcal{F}$. Then $\succ$ can be extended to a simplification ordering $\succ'$ on $\mathcal{T}(\mathcal{F}^\sharp, \mathcal{V})$ by defining $t \succ' u$ if and only if $\Omega(t) \succ \Omega(u)$ holds. We claim that the simplification ordering $\succ'$ satisfies the constraints (a) and (b) of Definition 4.2 without applying an argument filtering.

Obviously, $l \succ' r$ holds for all rules $l \to r$ of $\mathcal{R}$. Moreover, for every dependency pair $s \to t$ we have $s \succ' t$. The reason is that each dependency pair $f^\sharp(s_1, \ldots, s_n) \to g^\sharp(t_1, \ldots, t_m)$ originates from a rule $f(s_1, \ldots, s_n) \to C[g(t_1, \ldots, t_m)]$ in $\mathcal{R}$. Thus, $f(\ldots) \succ C[g(\ldots)]$ implies $f(\ldots) \succ g(\ldots)$ by the subterm property which in turn implies $f^\sharp(\ldots) \succ' g^\sharp(\ldots)$. Hence, $\succ'$ satisfies both constraints (a) and (b) of Definition 4.2. □

The following examples show that none of the converse implications of Lemma 4.4 holds.

Example 4.5: *The system $\{\mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{c}(\mathsf{f}(x)))\}$ is DP simply terminating as the only dependency pair on a cycle is $\mathsf{F}(\mathsf{f}(x)) \to \mathsf{F}(x)$. Hence, the resulting* *constraints are satisfied by RPO if one uses the argument filtering that maps* $\mathsf{c}(x)$ *to its argument. However, this TRS is not simply terminating. The TRS*

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

is DP quasi-simply terminating as can be proved in a similar way using the argument filtering which maps c *and* d *to their arguments, and RPO where* 0 *and* 1 *are equal in the precedence. However, it is not DP simply terminating, because due to the first four rules, the argument filtering must reduce* $\mathsf{c}(x)$ *and* $\mathsf{d}(x)$ *to their arguments. But then* $\mathsf{g}(\mathsf{0}) \succeq \mathsf{g}(\mathsf{1})$ *and* $\mathsf{g}(\mathsf{1}) \succeq \mathsf{g}(\mathsf{0})$ *lead to a contradiction.*

Finally, the system $\{\mathsf{f}(\mathsf{0},\mathsf{1},x) \to \mathsf{f}(x,x,x)\}$ *is terminating but not DP quasi-simply terminating. The reason is that* $\{\mathsf{F}(\mathsf{0},\mathsf{1},x) \to \mathsf{F}(x,x,x)\}$ *is a cycle in the estimated dependency graph, but there is no argument filtering* π *and no reduction pair* $(\succsim,\succ)$ *with a QSO* $\succsim$ *that satisfies* $\pi(\mathsf{F}(\mathsf{0},\mathsf{1},x)) \succ \pi(\mathsf{F}(x,x,x))$.

One might remark that the definition of argument filtering could be modified by not only eliminating arguments but by also identifying different function symbols. This would change the notion of DP simple termination, but DP simple termination and DP quasi-simple termination would still not coincide. To see this, one can replace the last two rules in the second system of Example 4.5.

$$\begin{array}{rclcrclcrcl}
\mathsf{f}(\mathsf{f}(x)) & \to & \mathsf{f}(\mathsf{c}(\mathsf{f}(x))) & \quad & \mathsf{g}(\mathsf{c}(x)) & \to & x & \quad & \mathsf{g}(\mathsf{c}(\mathsf{h}(\mathsf{0}))) & \to & \mathsf{g}(\mathsf{d}(\mathsf{1})) \\
\mathsf{f}(\mathsf{f}(x)) & \to & \mathsf{f}(\mathsf{d}(\mathsf{f}(x))) & & \mathsf{g}(\mathsf{d}(x)) & \to & x & & \mathsf{g}(\mathsf{c}(\mathsf{1})) & \to & \mathsf{g}(\mathsf{d}(\mathsf{h}(\mathsf{0}))) \\
 & & & & & & & & \mathsf{g}(\mathsf{h}(x)) & \to & \mathsf{g}(x)
\end{array}$$

The system is still DP quasi-simply terminating as can be shown by a polynomial ordering with $|\mathsf{h}(t)| = |t| + 1$, $|\mathsf{0}| = 0$, $|\mathsf{1}| = 1$, $|\mathsf{f}(t)| = |t| + 1$, where all other function symbols are mapped to the identity. However, even with the new definition of argument filtering, the system is still not DP simply terminating. The reason is that again, the argument filtering π must map c and d to their arguments. Then the third and fourth g-rule imply $\pi(\mathsf{g}(\mathsf{h}(\mathsf{0}))) = \pi(\mathsf{g}(\mathsf{1}))$. Since $\pi(\mathsf{g}) \neq [\,]$ due to the first g-rule, this implies $\pi(\mathsf{h}(\mathsf{0})) = \pi(\mathsf{1})$. Due to the dependency pair $\mathsf{G}(\mathsf{h}(x)) \to \mathsf{G}(x)$, π may neither map h to its argument nor to any constant like 1. Hence, even with this alternative definition of argument filtering, these constraints are not satisfiable.

4.3. Combining Disjoint Systems

In this section we show that DP quasi-simple termination is modular for disjoint TRSs. For the proof, we need the following lemma.

LEMMA 4.6 (TRANSFORMING REDUCTION SEQUENCES): *Let* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *be two TRSs over disjoint signatures* $\mathcal{F}_1$ *and* $\mathcal{F}_2$*, respectively. Furthermore, let* $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ *be their union. If* u, v *are terms over the signature* $\mathcal{F}_1$ *such that* $u \to_{\mathcal{R}_1} v$ *and* $v\sigma \to^*_{\mathcal{R}} u\sigma$ *hold for a ground substitution* $\sigma : \mathcal{V}ar(u) \to \mathcal{T}(\mathcal{F}_1 \cup \mathcal{F}_2)$*, then there is also a ground substitution* $\tau : \mathcal{V}ar(u) \to \mathcal{T}(\mathcal{F}_1)$ *such that* $u\tau \to_{\mathcal{R}_1} v\tau \to^*_{\mathcal{R}_1 \cup \mathcal{E}mb(\mathcal{F}_1)} u\tau$.

Proof: Clearly, all terms in the cyclic derivation

$$D: \quad u\sigma \to_{\mathcal{R}_1} v\sigma \to_{\mathcal{R}}^* u\sigma$$

have the same rank. Since the root symbol of u is in $\mathcal{F}_1$, the root symbol of every term in the reduction sequence D is also in $\mathcal{F}_1$ (reduction steps which are destructive at level 1 would decrease the rank).

Suppose first that every function symbol in $\mathcal{F}_1$ has arity ≤ 1. Then every reduction step in D which is destructive at level 2 strictly decreases the rank. Consequently, there is no reduction step of this kind in D. Hence

$$\mathrm{top}_1(u\sigma) \to_{\mathcal{R}_1} \mathrm{top}_1(v\sigma) \to_{\mathcal{R}_1}^* \mathrm{top}_1(u\sigma)$$

is an $\mathcal{R}_1$-reduction sequence of ground terms over $\mathcal{F}_1 \cup \{\Box\}$. Let $\mathcal{V}ar(u) = \{x_1, \ldots, x_n\}$ and recall $\mathcal{V}ar(v) \subseteq \mathcal{V}ar(u)$. In this case, we define the substitution τ by $\tau = \{x_i \mapsto \mathrm{top}_1(x_i\sigma)' \mid 1 \leq i \leq n\}$, where $\mathrm{top}_1(t)'$ results from $\mathrm{top}_1(t)$ by replacing all holes $\Box$ by an arbitrary constant from $\mathcal{F}_1$ (note that we restricted ourselves to signatures containing at least one constant). Then

$$u\tau = \mathrm{top}_1(u\sigma)' \to_{\mathcal{R}_1} \mathrm{top}_1(v\sigma)' = v\tau \to_{\mathcal{R}_1}^* \mathrm{top}_1(u\sigma)' = u\tau$$

is the reduction sequence we are looking for.

Suppose otherwise that there is a function symbol f in $\mathcal{F}_1$ with arity $m > 1$. Let Cons be a binary function symbol which neither occurs in $\mathcal{F}_1$ nor in $\mathcal{F}_2$ and let $\mathcal{C}_{\mathcal{E}} = \{\mathrm{Cons}(x_1, x_2) \to x_1, \mathrm{Cons}(x_1, x_2) \to x_2\}$. By [Gramlich, 1994, Lemma 3.8] or [Ohlebusch, 1994b, Theorem 3.13], the reduction sequence D can be transformed by a transformation function[‡] Φ into a reduction sequence

$$\Phi(u\sigma) \to_{\mathcal{R}_1} \Phi(v\sigma) \to_{\mathcal{R}_1 \cup \mathcal{C}_{\mathcal{E}}}^* \Phi(u\sigma)$$

of terms over $\mathcal{F}_1 \cup \{\mathrm{Cons}\}$. The transformation function Φ satisfies $\Phi(t) = C[\Phi(t_1), \ldots, \Phi(t_n)]$ for every term t with $\mathrm{root}(t) \in \mathcal{F}_1$ and $t = C[\![t_1, \ldots, t_n]\!]$, cf. [Ohlebusch, 1994b]. In this case, we first define $\sigma' = \{x_i \mapsto \Phi(x_i\sigma) \mid 1 \leq i \leq n\}$ and obtain

$$u\sigma' = \Phi(u\sigma) \to_{\mathcal{R}_1} \Phi(v\sigma) = v\sigma' \to_{\mathcal{R}_1 \cup \mathcal{C}_{\mathcal{E}}}^* \Phi(u\sigma) = u\sigma' \ .$$

Let $u\sigma' = u_0, u_1, \ldots, u_k = u\sigma'$ be the sequence of terms occurring in the above reduction sequence. Now in each term u_i replace every $\mathrm{Cons}(t_1, t_2)$ with $f(t_1, t_2, z, \ldots, z)$, where z is a variable or a constant from $\mathcal{F}_1$, and denote the resulting term by $\Psi(u_i)$. The definition $\tau = \{x_i \mapsto \Psi(x_i\sigma') \mid 1 \leq i \leq n\}$ yields the desired reduction sequence

$$u\tau = \Psi(u\sigma') = \Psi(u_0) \to_{\mathcal{R}_1} \Psi(u_1) = \Psi(v\sigma') = v\tau \to_{\mathcal{R}_1 \cup \mathcal{E}mb(\mathcal{F}_1)}^* \Psi(u_k) = u\tau$$

in which $\Psi(u_i) \to_{\mathcal{R}_1 \cup \mathcal{E}mb(\mathcal{F}_1)} \Psi(u_{i+1})$ by the rule $f(x_1, \ldots, x_m) \to x_j$, $j \in \{1, 2\}$, if $u_i \to_{\mathcal{R}_1 \cup \mathcal{C}_{\mathcal{E}}} u_{i+1}$ by the rule $\mathrm{Cons}(x_1, x_2) \to x_j$. $\Box$

[‡]More precisely, Φ is the transformation $\Phi_1^{u\sigma}$ defined in [Ohlebusch, 1994b, Definition 3.10].

Now we are in a position to prove our modularity theorem for DP quasi-simple termination.

Theorem 4.7 (Modularity of DP quasi-simple termination): *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two TRSs over disjoint signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. Then their union $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ is DP quasi-simply terminating if and only if both $\mathcal{R}_1$ and $\mathcal{R}_2$ are DP quasi-simply terminating.*

Proof: The only-if direction is trivial. For the if direction, let $\mathcal{P}$ be a cycle in the estimated dependency graph of $\mathcal{R}$. Since $\mathcal{R}_1$ and $\mathcal{R}_2$ are disjoint, $\mathcal{P}$ is a cycle in the estimated dependency graph of $\mathcal{R}_1$ or of $\mathcal{R}_2$. Without loss of generality, let $\mathcal{P}$ be a cycle in the estimated dependency graph of $\mathcal{R}_1$.

As $\mathcal{R}_1$ is DP quasi-simply terminating, there is an argument filtering π for $\mathcal{F}_1^\sharp$ such that the constraints (a) and (b) of Definition 4.1 are satisfied for $\mathcal{R}_1$, $\mathcal{P}$, and some reduction pair $(\succsim, \succ)$, where $\succsim$ is a QSO. Now let

$$\begin{aligned} \mathcal{S}_1 &= \pi(\mathcal{R}_1 \cup \mathcal{P}) \cup \mathcal{E}mb(\mathcal{F}^\sharp_{1_\pi}) \\ \mathcal{S}_2 &= \mathcal{R}_2 \cup \mathcal{E}mb(\mathcal{F}_2). \end{aligned}$$

Due to our minor restriction on the argument filterings, $\mathcal{S}_1$ is a TRS over the signature $\mathcal{F}^\sharp_{1_\pi}$. Hence $\mathcal{R}' = \mathcal{S}_1 \cup \mathcal{S}_2$ is a TRS over $\mathcal{F}^\sharp_{1_\pi} \cup \mathcal{F}_2$. It is clear that $\to^*_{\mathcal{R}'}$ is a QSO.[§] Note however, that the strict part of $\to^*_{\mathcal{R}'}$ is not necessarily closed under substitutions. Instead we prove that the reduction pair consisting of $\to^*_{\mathcal{R}'}$ and its stable-strict relation satisfies the constraints of Definition 4.1, if π is extended to $\mathcal{F}_1^\sharp \cup \mathcal{F}_2$ by not filtering any arguments for function symbols from $\mathcal{F}_2$. As the cycle $\mathcal{P}$ was chosen arbitrarily, to prove DP quasi-simple termination of $\mathcal{R}$, we only have to show

(a) $\pi(\mathcal{R} \cup \mathcal{P}) \subseteq \to^*_{\mathcal{R}'}$ and
(b) there exists a dependency pair $s \to t$ from $\mathcal{P}$ such that
$\pi(t)\sigma \not\to^*_{\mathcal{R}'} \pi(s)\sigma$ holds for all ground substitutions σ.

Condition (a) is obviously satisfied, since for all $l \to r \in \mathcal{R}_2$ we have $\pi(l) = l$ and $\pi(r) = r$ and for all $l \to r$ in $\mathcal{R}_1 \cup \mathcal{P}$ either $\pi(l) = \pi(r)$ or $\pi(l) \to \pi(r)$ is a rule of $\mathcal{S}_1$. Hence, we only have to show conjecture (b). Since $\succsim$ is the QSO used for the DP quasi-simple termination proof of $\mathcal{R}_1$, we have $\to^*_{\mathcal{S}_1} \subseteq \succsim$. Let $s \to t$ be a dependency pair from $\mathcal{P}$ such that $\pi(s) \succ \pi(t)$. Suppose that there exists a ground substitution $\sigma : \mathcal{V}ar(\pi(s)) \to \mathcal{T}(\mathcal{F}^\sharp_{1_\pi} \cup \mathcal{F}_2)$ such that $\pi(t)\sigma \to^*_{\mathcal{R}'} \pi(s)\sigma$. By Lemma 4.6, this implies the existence of a ground substitution $\tau : \mathcal{V}ar(\pi(s)) \to \mathcal{T}(\mathcal{F}^\sharp_{1_\pi})$ such that $\pi(t)\tau \to^*_{\mathcal{S}_1} \pi(s)\tau$, since $\mathcal{E}mb(\mathcal{F}^\sharp_{1_\pi}) \subseteq \mathcal{S}_1$. (Here, $\mathcal{F}^\sharp_{1_\pi}$ corresponds to $\mathcal{F}_1$ in Lemma 4.6, $\pi(s)$ and $\pi(t)$ correspond to u and v, respectively, and $\mathcal{S}_1$ and $\mathcal{S}_2$ correspond to $\mathcal{R}_1$ and $\mathcal{R}_2$ in Lemma 4.6.) This would imply $\pi(t)\tau \succsim \pi(s)\tau$. Since $\succ$ is closed under substitutions, we therefore

[§]If $\mathcal{R}$ is a TRS over the signature $\mathcal{F}$ then $\to^*_{\mathcal{R}\cup\mathcal{E}mb(\mathcal{F})}$ is the smallest QSO containing $\to_\mathcal{R}$ (that is, if $\succsim$ is a QSO with $\to_\mathcal{R} \subseteq \succsim$, then $\to^*_{\mathcal{R}\cup\mathcal{E}mb(\mathcal{F})} \subseteq \succsim$).

would have $\pi(s)\tau \succ \pi(t)\tau \succsim \pi(s)\tau \succ \dots$ which contradicts the well-foundedness of $\succ$. Thus, $\pi(t)\sigma \not\to^*_{\mathcal{R}'} \pi(s)\sigma$ holds for all ground substitutions σ. This proves conjecture (b). Finally, note that, since $\pi(\mathcal{R} \cup \mathcal{P})$ is a TRS, the minor restriction on the argument filterings holds for this π. $\square$

Thus, if $\mathcal{R}_1$ is the TRS consisting of the rules (10) and (11) and $\mathcal{R}_2$ contains the rules (12) and (13), then this theorem allows us to conclude termination of their combination because both systems are DP quasi-simply terminating. This example cannot be handled by any of the previous modularity results. Note also that in this example, modularity of termination is far from being trivial because if $\mathcal{R}_1$'s rule $\mathsf{f}(\mathsf{0},\mathsf{1},x) \to \mathsf{f}(\mathsf{s}(x),x,x)$ would be just slightly changed to $\mathsf{f}(\mathsf{0},\mathsf{1},x) \to \mathsf{f}(x,x,x)$, then $\mathcal{R}_1$ would still be terminating, but the union with $\mathcal{R}_2$ would not terminate any more, cf. [Toyama, 1987]. It is interesting to note that Theorem 4.7 provides an elegant proof of the fact that $\mathsf{f}(\mathsf{0},\mathsf{1},x) \to \mathsf{f}(x,x,x)$ is not DP quasi-simply terminating because $\mathcal{R}_2$ is DP quasi-simply terminating but its union with $\mathsf{f}(\mathsf{0},\mathsf{1},x) \to \mathsf{f}(x,x,x)$ is non-terminating.

From the proof it is clear that the modularity result of Theorem 4.7 also holds if in the definition of DP quasi-simple termination we fix the ordering $\succ_{\mathcal{P}}$ to be the stable-strict relation corresponding to the QSO $\succsim_{\mathcal{P}}$. In other words, the termination proof of $\mathcal{R}_1 \cup \mathcal{R}_2$ also succeeds with reduction pairs consisting of a QSO and its associated stable-strict relation.

One should remark that a further extension of the modularity result in Theorem 4.7 beyond the class of DP quasi-simply terminating systems is not straightforward. For example, if one would define DP quasi-simple termination by using the real dependency graph instead of the estimated graph, then this notion of termination would no longer be modular for disjoint systems. The previous system would serve as a counterexample, since in the real dependency graph of $\mathsf{f}(\mathsf{0},\mathsf{1},x) \to \mathsf{f}(x,x,x)$ there is no cycle. Hence, it would depend on the rules of $\mathcal{R}_2$ whether dependency pairs of $\mathcal{R}_1$ form a cycle. The same problem occurs with the recent technique of [Middeldorp, 2001] where dependency graphs are approximated using tree automata techniques.

DP quasi-simply terminating systems occur frequently in practice. Consider the following two TRSs where nil denotes the empty list and $x : l$ represents the insertion of a number x into a list l. Here $\mathsf{sum}(l)$ computes a singleton list containing the sum of all elements in the list l.

$$\begin{array}{rrcl}
\mathcal{R}_1: & x - \mathsf{0} & \to & x \\
& \mathsf{s}(x) - \mathsf{s}(y) & \to & x - y \\
& \mathsf{quot}(\mathsf{0},\mathsf{s}(y)) & \to & \mathsf{0} \\
& \mathsf{quot}(\mathsf{s}(x),\mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{quot}(x-y,\mathsf{s}(y)))
\end{array}
\qquad
\begin{array}{rrcl}
\mathcal{R}_2: & \mathsf{app}(\mathsf{nil},k) & \to & k \\
& \mathsf{app}(l,\mathsf{nil}) & \to & l \\
& \mathsf{app}(x:l,k) & \to & x:\mathsf{app}(l,k) \\
& \mathsf{sum}(x:\mathsf{nil}) & \to & x:\mathsf{nil} \\
& \mathsf{sum}(x:(y:l)) & \to & \mathsf{sum}((x+y):l) \\
& \mathsf{sum}(\mathsf{app}(l,x:(y:k))) & \to & \mathsf{sum}(\mathsf{app}(l,\mathsf{sum}(x:(y:k))))
\end{array}$$

Both TRSs above are not simply terminating, but they are both DP quasi-simply terminating, cf. [Arts and Giesl, 2000]. Hence, Theorem 4.7 now also allows to conclude DP quasi-simple termination of their union.

4.4. Combining Constructor-Sharing and Composable Systems

It may be a bit surprising that Theorem 4.7 cannot be directly extended to constructor-sharing TRSs; even if we disallow the use of argument filterings. In other words, there are constructor-sharing TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ which are both DP quasi-simply terminating, but their union $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ is not DP quasi-simply terminating.

Example 4.8: *Consider the following TRSs:*

$$\begin{array}{llcl} \mathcal{R}_1: & \mathsf{f}(\mathsf{c}(x)) & \to & \mathsf{f}(x) \\ & \mathsf{f}(\mathsf{b}(x)) & \to & x \end{array} \qquad \begin{array}{llcl} \mathcal{R}_2: & \mathsf{g}(\mathsf{d}(x)) & \to & \mathsf{g}(x) \\ & \mathsf{g}(\mathsf{c}(x)) & \to & \mathsf{c}(\mathsf{g}(\mathsf{b}(\mathsf{c}(x)))) \end{array}$$

$\mathcal{R}_1$ and $\mathcal{R}_2$ are DP quasi-simply terminating. ($\mathcal{R}_1$ is even simply terminating and $\mathcal{R}_2$ is already DP simply terminating as can be shown using the argument filtering $\pi(\mathsf{b}) = [\,]$ and RPO. Alternatively, DP quasi-simple termination of $\mathcal{R}_2$ can even be shown without any argument filtering by using a polynomial ordering which maps c, b, g, and G to the identity and which maps $\mathsf{d}(x)$ to $x+1$.) However, the union of $\mathcal{R}_1$ and $\mathcal{R}_2$ is not DP quasi-simply terminating. As $\mathsf{F}(\mathsf{c}(x)) \to \mathsf{F}(x)$ represents a cycle in the estimated dependency graph one would have to find a QSO satisfying

$$\begin{array}{rcl} \mathsf{f}(\mathsf{c}(x)) & \succsim & \mathsf{f}(x) \qquad (16) \\ \mathsf{f}(\mathsf{b}(x)) & \succsim & x \qquad (17) \\ \mathsf{g}(\mathsf{d}(x)) & \succsim & \mathsf{g}(x) \qquad (18) \\ \mathsf{g}(\mathsf{c}(x)) & \succsim & \mathsf{c}(\mathsf{g}(\mathsf{b}(\mathsf{c}(x)))) \qquad (19) \\ \mathsf{F}(\mathsf{c}(x)) & \succ & \mathsf{F}(x). \qquad (20) \end{array}$$

Without argument filtering, no QSO satisfies (16) - (20), since otherwise we would have

$$\begin{array}{rcll} \mathsf{F}(\mathsf{c}(\mathsf{g}(\mathsf{c}(x)))) & \succ & \mathsf{F}(\mathsf{g}(\mathsf{c}(x))) & \textit{due to (20)} \\ & \succsim & \mathsf{F}(\mathsf{c}(\mathsf{g}(\mathsf{b}(\mathsf{c}(x))))) & \textit{due to (19)} \\ & \succsim & \mathsf{F}(\mathsf{c}(\mathsf{g}(\mathsf{c}(x)))) & \textit{due to the subterm property.} \end{array}$$

By (20), the argument filtering can only map c to $[1]$, i.e., $\pi(\mathsf{c}(x)) = \mathsf{c}(x)$. If $\pi(\mathsf{b}) = [\,]$ then (17) would be transformed into $\mathsf{f}(\mathsf{b}) \succsim x$. But as there exists the strict inequality (20) with a variable in its right-hand side, this results in the contradiction $\mathsf{F}(\mathsf{c}(\mathsf{f}(\mathsf{b}))) \succ \mathsf{F}(\mathsf{f}(\mathsf{b})) \succsim \mathsf{F}(x)$. Similarly, the argument of g cannot be eliminated either, since $\mathsf{g} \succsim \mathsf{c}(\mathsf{g})$ would be a contradiction to (20).

Thus, the only possible argument filtering maps b or g to its argument. But then *we would again obtain* $\mathsf{F}(\mathsf{c}(\mathsf{g}(\mathsf{c}(x)))) \succ \circ \succsim \mathsf{F}(\mathsf{c}(\mathsf{g}(\mathsf{c}(x))))$ *or* $\mathsf{F}(\mathsf{c}(\mathsf{c}(x))) \succ \circ \succsim \mathsf{F}(\mathsf{c}(\mathsf{c}(x)))$ *as above. Hence, the TRS indeed is not DP quasi-simply terminating.*

Thus, in order to obtain a modularity result for constructor-sharing combinations we have to exclude TRSs like $\mathcal{R}_2$. Note that without applying an argument filtering, DP simple termination of the TRS $\mathcal{R}_2$ cannot be proved (while DP quasi-simple termination can be shown without using any argument filtering at all). Thus, we will impose two restrictions: (a) In the remainder of the section we will restrict ourselves to DP simple termination instead of DP quasi-simple termination and (b) we have to restrict ourselves to systems where the argument filtering does not eliminate arguments for shared symbols like b.

But we need another requirement to ensure modularity. For example, let us remove the first rule $\mathsf{g}(\mathsf{d}(x)) \to \mathsf{g}(x)$ from $\mathcal{R}_2$. Now there is no cycle in the estimated dependency graph of $\mathcal{R}_2$ any more and hence we obtain no constraints at all for $\mathcal{R}_2$. Thus, DP simple termination of $\mathcal{R}_2$ can now even be proved without using argument filterings, but the combined system $\mathcal{R}_1 \cup \mathcal{R}_2$ is still not DP simply terminating. Here, the problem is due to the fact that TRSs without cycles are DP simply terminating, even if there is no simplification ordering $\succ$ such that $l \succeq r$ holds for their rules. To exclude such TRSs we will demand that the constraint (a) of Definition 4.2 (i.e., $\pi(l) \succeq \pi(r)$ for all rules) should also be satisfied even if there does not exist any cycle $\mathcal{P}$. Thus, in the following we also take the *empty* cycle $\mathcal{P}$ into account.

With this additional requirement, DP simple termination is at least modular for *disjoint* combinations¶, whereas without this requirement, Theorem 4.7 would not hold for DP simple termination instead of DP quasi-simple termination. As a counterexample consider the TRS $\mathcal{R}_1$ with the rule $\mathsf{f}(\mathsf{s}(x)) \to \mathsf{f}(x)$ and the TRS $\mathcal{R}_2$ with the rules

$$\begin{array}{lllllllll}
\mathsf{g}(\mathsf{0}) & \to & \mathsf{g}(\mathsf{c}(\mathsf{0})) & \quad \mathsf{g}(\mathsf{c}(x)) & \to & x & \quad \mathsf{g}(\mathsf{c}(\mathsf{0})) & \to & \mathsf{g}(\mathsf{d}(\mathsf{1})) \\
\mathsf{g}(\mathsf{0}) & \to & \mathsf{g}(\mathsf{d}(\mathsf{0})) & \quad \mathsf{g}(\mathsf{d}(x)) & \to & x & \quad \mathsf{g}(\mathsf{c}(\mathsf{1})) & \to & \mathsf{g}(\mathsf{d}(\mathsf{0})).
\end{array}$$

$\mathcal{R}_1$ is even simply terminating. $\mathcal{R}_2$ is DP simply terminating, but the reason is just that there does not exist any cycle in its estimated dependency graph. However, when combining $\mathcal{R}_1$ and $\mathcal{R}_2$, their union has a cycle and hence, one now also has to demand $\pi(l) \succeq \pi(r)$ for the rules of $\mathcal{R}_2$. However, for all argument filterings π, this is not fulfilled by any QSO whose equivalence relation is just syntactic equality. So their union is not DP simply terminating, but of course due to Theorem 4.7 it is DP quasi-simply terminating.

Nonetheless, the following example shows that this restriction is not yet sufficient for obtaining a modularity result for DP simple termination of *constructor-sharing* systems.

¶This can be proved similar to Theorem 4.7 using the simplification ordering $\to^+_{\mathcal{R}'}$ where instead of condition (b) in this proof one only has to show that $\pi(s) \neq \pi(t)$ holds for some dependency pair $s \to t$ from $\mathcal{P}$ (this follows immediately from DP simple termination of $\mathcal{R}_1$).

EXAMPLE 4.9: *Let $\mathcal{R}_1$ consist of the rules*

$$\begin{array}{rclcrcl} \mathsf{g}(\mathsf{s}(x)) & \to & \mathsf{g}(x) & \qquad & \mathsf{g}(\mathsf{0}) & \to & \mathsf{g}(\mathsf{1}) \\ \mathsf{g}(\mathsf{s}(x)) & \to & x & & \mathsf{f}(\mathsf{0}) & \to & \mathsf{g}(\mathsf{f}(\mathsf{s}(\mathsf{0}))) \end{array}$$

and let $\mathcal{R}_2$ consist of the rule $\mathsf{h}(\mathsf{1}) \to \mathsf{h}(\mathsf{0})$. To prove DP simple termination of $\mathcal{R}_1$ we have to use an argument filtering mapping f *to* [ ] *and* g *to* 1*. This, however, would imply $\mathsf{0} \succ \mathsf{1}$ which is a contradiction to $\mathsf{h}(\mathsf{1}) \succsim \mathsf{h}(\mathsf{0})$. Thus, the combination of both systems is not DP simply terminating.*

So we also have to ensure that an application of the argument filtering to the resulting inequalities does not transform left-hand sides which had a non-shared root symbol like g into terms with a shared root symbol (like the former constructor 0).[‖] For that reason we have to demand the following compatibility requirement for all argument filterings used, where $\mathcal{G}$ must contain all shared function symbols.

DEFINITION 4.10 ($\mathcal{G}$-COMPATIBILITY): *Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$ and let $\mathcal{G}$ be a signature. An argument filtering π for $\mathcal{F}$ is $\mathcal{G}$-*compatible *for $\mathcal{R}$ if and only if*

(a) $\pi(f) = [1, \ldots, n]$ for every $f \in \mathcal{F} \cap \mathcal{G}$, where n is the arity of f (i.e., π does not filter arguments for function symbols from $\mathcal{G}$).

(b) For every rule $l \to r \in \mathcal{R}$: if $\mathrm{root}(l) \notin \mathcal{G}$, then $\mathrm{root}(\pi(l)) \notin \mathcal{G}$.

The restriction to $\mathcal{G}$-compatible argument filterings ensures that symbols from $\mathcal{F} \cap \mathcal{G}$ are not changed and furthermore constructors from $\mathcal{F} \cap \mathcal{G}$ are not turned into defined symbols after application of the argument filtering. In the following, for any TRS $\mathcal{R}$ over the signature $\mathcal{F}$ let $\mathcal{C}_\pi$ be the set of constructors of $\pi(\mathcal{R})$, and let $\mathcal{D}_\pi$ be the set of defined symbols in $\pi(\mathcal{R})$.

LEMMA 4.11 (PROPERTIES OF $\mathcal{G}$-COMPATIBLE ARGUMENT FILTERINGS): *Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F} = \mathcal{C} \cup \mathcal{D}$ and let π be an argument filtering for $\mathcal{F}$ that is $\mathcal{G}$-compatible for $\mathcal{R}$. Then the following statements hold:*

(i) For every rule $l \to r \in \mathcal{R}$: if $\mathrm{root}(l) \in \mathcal{G}$, then $\mathrm{root}(\pi(l)) = \mathrm{root}(l)$.

(ii) For every rule $l \to r \in \mathcal{R}$: if $\mathrm{root}(\pi(l)) \in \mathcal{G}$, then $\mathrm{root}(\pi(l)) = \mathrm{root}(l)$.

(iii) $\mathcal{G} \cap \mathcal{D}_\pi \subseteq \mathcal{G} \cap \mathcal{D}$.

Proof:

(i) Immediate consequence of Definition 4.10 (a).

(ii) It follows from Definition 4.10 (b) that $\mathrm{root}(l) \in \mathcal{G}$. Hence (ii) is a consequence of (i).

[‖]If the argument filtering is non-collapsing (i.e., $\pi(f) \neq i$ for all defined symbols f), then this requirement is always fulfilled.

(iii) Follows directly from (ii).

□

The following lemma is crucial to our modularity result, because it states that if $\mathcal{R}_1$ and $\mathcal{R}_2$ are constructor-sharing, then applying an argument filtering π will also result in constructor-sharing TRSs $\pi(\mathcal{R}_1)$ and $\pi(\mathcal{R}_2)$ provided that π is compatible with the set of all shared symbols. In fact, this result even holds for *composable* TRSs instead of constructor-sharing ones.

Lemma 4.12 ($\mathcal{G}$-compat. Arg. Filterings Maintain Composability): *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be composable TRSs over the signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. If $\mathcal{F}_1 \cap \mathcal{F}_2 \subseteq \mathcal{G}$ and if π is an argument filtering for $\mathcal{F}_1 \cup \mathcal{F}_2$ that is $\mathcal{G}$-compatible for $\mathcal{R}_1$ and for $\mathcal{R}_2$, then $\pi(\mathcal{R}_1)$ and $\pi(\mathcal{R}_2)$ are also composable.*

Proof: We prove the following claims (where (B) and (C) imply that $\pi(\mathcal{R}_1)$ and $\pi(\mathcal{R}_2)$ are composable):

(A) $\{l \to r \in \pi(\mathcal{R}_1) \cup \pi(\mathcal{R}_2) \mid \text{root}(l) \in \mathcal{D}_1 \cap \mathcal{D}_2\} \subseteq \pi(\mathcal{R}_1) \cap \pi(\mathcal{R}_2)$

(B) $\{l \to r \in \pi(\mathcal{R}_1) \cup \pi(\mathcal{R}_2) \mid \text{root}(l) \in \mathcal{D}_{1\pi} \cap \mathcal{D}_{2\pi}\} \subseteq \pi(\mathcal{R}_1) \cap \pi(\mathcal{R}_2)$

(C) $\mathcal{C}_{1\pi} \cap \mathcal{D}_{2\pi} = \mathcal{D}_{1\pi} \cap \mathcal{C}_{2\pi} = \emptyset$

(A) If $l \to r \in \pi(\mathcal{R}_1) \cup \pi(\mathcal{R}_2)$, then we have $l = \pi(u)$ and $r = \pi(v)$ for some $u \to v \in \mathcal{R}_1 \cup \mathcal{R}_2$. Note that $\text{root}(\pi(u)) \in \mathcal{D}_1 \cap \mathcal{D}_2 \subseteq \mathcal{G}$ implies $\text{root}(\pi(u)) = \text{root}(u)$ by Lemma 4.11 (ii). As $\text{root}(u) \in \mathcal{D}_1 \cap \mathcal{D}_2$ and as $\mathcal{R}_1$ and $\mathcal{R}_2$ are composable, this implies $u \to v \in \mathcal{R}_1 \cap \mathcal{R}_2$. It follows that $\pi(u) \to \pi(v) \in \pi(\mathcal{R}_1) \cap \pi(\mathcal{R}_2)$ because $\pi(u) \to \pi(v) \in \pi(\mathcal{R}_1) \cup \pi(\mathcal{R}_2)$ implies $\pi(u) \neq \pi(v)$.

(B) If $f = \text{root}(l) \in \mathcal{D}_{1\pi} \cap \mathcal{D}_{2\pi}$, then a function symbol f (with possibly different arity) occurs in $\mathcal{F}_1 \cap \mathcal{F}_2 \subseteq \mathcal{G}$ by the definition of argument filterings. But then due to Definition 4.10 (a), $f \in \mathcal{F}_1 \cap \mathcal{F}_2 \subseteq \mathcal{G}$ has the same arity as $f \in \mathcal{D}_{1\pi} \cap \mathcal{D}_{2\pi}$. Hence, $f \in \mathcal{D}_1 \cap \mathcal{D}_2$ follows from Lemma 4.11 (iii). Now the claim is implied by (A).

(C) If there were an $f \in \mathcal{C}_{1\pi} \cap \mathcal{D}_{2\pi}$, then similar to the argumentation in (B), we would have $f \in \mathcal{F}_1 \cap \mathcal{F}_2 \subseteq \mathcal{G}$ by the definition of argument filterings and since π is $\mathcal{G}$-compatible. This implies $f \in \mathcal{F}_1 \cap \mathcal{D}_2$ according to Lemma 4.11 (iii). We know $\mathcal{C}_1 \cap \mathcal{D}_2 = \emptyset$ because $\mathcal{R}_1$ and $\mathcal{R}_2$ are composable. Thus, we have $f \in \mathcal{D}_1 \cap \mathcal{D}_2$. But since there is a rule $l \to r \in \pi(\mathcal{R}_2)$ with $\text{root}(l) = f$, (A) implies $l \to r \in \pi(\mathcal{R}_1)$ and thus, $\text{root}(l) = f \in \mathcal{D}_{1\pi}$, which is a contradiction to $f \in \mathcal{C}_{1\pi}$. The proof of $\mathcal{D}_{1\pi} \cap \mathcal{C}_{2\pi} = \emptyset$ is exactly the same. □

The restrictions needed for the desired modularity result are captured by the notion of $\mathcal{G}$*-restricted* DP simple termination.

Definition 4.13 ($\mathcal{G}$-restricted DP simple termination): *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is called $\mathcal{G}$-*restricted *DP simply terminating if and only* *if for each cycle $\mathcal{P}$ in the estimated dependency graph of $\mathcal{R}$ (including the* empty *one) there is an argument filtering $\pi_{\mathcal{P}}$ for $\mathcal{F}^\sharp$ that is $\mathcal{G}$-compatible for $\mathcal{R} \cup \mathcal{P}$ such that*

- *$\pi_{\mathcal{P}}(\mathcal{R} \cup \mathcal{P})$ is a simply terminating TRS and*
- *$\pi_{\mathcal{P}}(\mathcal{P}) \neq \emptyset$ whenever $\mathcal{P} \neq \emptyset$.*

So obviously, $\mathcal{G}$-restricted DP simple termination implies DP simple termination, cf. Corollary 4.3. The following theorem shows that under this $\mathcal{G}$-restriction, DP simple termination is modular for constructor-sharing and even for composable TRSs.

Theorem 4.14 (Modularity of $\mathcal{G}$-restricted DP simple terminat.): *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be composable TRSs over the signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. If $\mathcal{F}_1 \cap \mathcal{F}_2 \subseteq \mathcal{G}$, then their combined system $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ is $\mathcal{G}$-restricted DP simply terminating if and only if both $\mathcal{R}_1$ and $\mathcal{R}_2$ are $\mathcal{G}$-restricted DP simply terminating.*

Proof: The only-if direction is trivial. For the if direction, let $\mathcal{P}$ be a cycle in the estimated dependency graph of $\mathcal{R}$ (where $\mathcal{P}$ may also be empty). Then $\mathcal{P}$ is also a cycle in the estimated dependency graph of $\mathcal{R}_1$ or in the estimated dependency graph of $\mathcal{R}_2$ because $\mathcal{R}_1$ and $\mathcal{R}_2$ are composable. The reason is that dependency pairs of the form $f^\sharp(\ldots) \to g^\sharp(\ldots)$ where $g \in \mathcal{F}_1 \cap \mathcal{F}_2$ and $f \notin \mathcal{F}_1 \cap \mathcal{F}_2$ are not on cycles. Thus, the only dependency pairs $f^\sharp(\ldots) \to g^\sharp(\ldots)$ on cycles have $f, g \in \mathcal{F}_i \setminus \mathcal{F}_{3-i}$ or $f, g \in \mathcal{F}_1 \cap \mathcal{F}_2$. Without loss of generality let $\mathcal{P}$ be a cycle in the estimated dependency graph of $\mathcal{R}_1$. Let $\mathcal{S}_1 = \mathcal{R}_1 \cup \mathcal{P}$ and let $\mathcal{S}_2 = \mathcal{R}_2$. Note that $\mathcal{S}_1$ and $\mathcal{S}_2$ are composable, since the root symbols in the new rules $\mathcal{P}$ are tuple symbols which therefore do not occur in $\mathcal{R}_2$. We have to show that there is an argument filtering π for $\mathcal{F}_1^\sharp \cup \mathcal{F}_2$ that is $\mathcal{G}$-compatible for $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ such that $\pi(\mathcal{S})$ is simply terminating and such that $\pi(\mathcal{P}) \neq \emptyset$ if $\mathcal{P} \neq \emptyset$.

Since $\mathcal{R}_1$ and $\mathcal{R}_2$ are $\mathcal{G}$-restricted DP simply terminating, there are argument filterings π_1 and π_2 such that π_i is $\mathcal{G}$-compatible with $\mathcal{S}_i$ and such that $\pi_i(\mathcal{S}_i)$ is simply terminating (for both $i \in \{1, 2\}$). For $i = 2$ this is because we also regard the empty cycle in Definition 4.13. Moreover, $\pi_1(\mathcal{P}) \neq \emptyset$ if $\mathcal{P} \neq \emptyset$. Let π operate like π_1 on $\mathcal{F}_1$ and like π_2 on $\mathcal{F}_2$. (This is well defined, since π_1 and π_2 do not modify function symbols from $\mathcal{F}_1 \cap \mathcal{F}_2 \subseteq \mathcal{G}$.) Clearly, $\pi(\mathcal{P}) = \pi_1(\mathcal{P})$ and thus, $\pi(\mathcal{P}) \neq \emptyset$ if $\mathcal{P} \neq \emptyset$. Moreover, obviously π is $\mathcal{G}$-compatible for both $\mathcal{S}_1$ and $\mathcal{S}_2$ and hence, also for $\mathcal{S}$. Then by Lemma 4.12, $\pi(\mathcal{S}_1)$ and $\pi(\mathcal{S}_2)$ are composable, since $\mathcal{S}_1$ and $\mathcal{S}_2$ are composable as well. Thus, by [Ohlebusch, 1995, Theorem 5.16], the combined system $\pi(\mathcal{S}_1) \cup \pi(\mathcal{S}_2) = \pi(\mathcal{S}_1 \cup \mathcal{S}_2)$ is also simply terminating. This implies $\mathcal{G}$-restricted DP simple termination of $\mathcal{R}_1 \cup \mathcal{R}_2$. □

For example, let us extend both TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ from the end of Section 4.3 by the additional rules

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

and moreover, we also add the rule $(x - y) - z \to x - (y + z)$ to $\mathcal{R}_1$. Now the resulting TRSs are composable, since they both contain the same constructors $\mathsf{0}$ and s and they also share the defined symbol $+$, but both TRSs contain the same $+$-rules. As both TRSs are $\{\mathsf{0}, \mathsf{s}, +\}$-restricted DP simply terminating, Theorem 4.14 allows us to conclude $\{\mathsf{0}, \mathsf{s}, +\}$-restricted DP simple termination of the combined system.

There are even TRSs $\mathcal{R}_1 \cup \mathcal{R}_2$ where DP simple termination of both $\mathcal{R}_1$ and $\mathcal{R}_2$ can be proved with a standard technique like LPO, whereas such standard orderings fail if one wants to prove DP simple termination of their union directly. Hence, for such examples our result enables automatic termination proofs which were not possible before.

Example 4.15: *Let $\mathcal{R}_1$ be the TRS*

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

and let $\mathcal{R}_2$ consist of the rule $\mathsf{g}(\mathsf{c}(x, \mathsf{s}(y))) \to \mathsf{g}(\mathsf{c}(\mathsf{s}(x), y))$.

$\mathcal{R}_1$ is DP simply terminating (using the argument filtering $\pi(\mathsf{d}) = [\,]$ and LPO comparing subterms left-to-right), but it is not simply terminating. $\mathcal{R}_2$ is even simply terminating as can be shown with LPO comparing subterms right-to-left. Thus, DP simple termination of both systems can be verified by LPO.

By Theorem 4.14 their union is also DP simply terminating. However, the constraints for the cycle $\{\mathsf{G}(\mathsf{c}(x, \mathsf{s}(y))) \to \mathsf{G}(\mathsf{c}(\mathsf{s}(x), y))\}$ are not satisfied by LPO (nor by RPO nor by any polynomial ordering). Thus, there are indeed TRSs where termination of the subsystems can be shown with dependency pairs and LPO, but (without our modularity result) termination of their union cannot be proved with dependency pairs and LPO.

5. Modular Innermost Termination Proofs With Dependency Pairs

Arts and Giesl [2000] showed that the dependency pair approach can be modified in order to verify *innermost* termination. Unlike previous methods, this technique can even prove innermost termination of non-terminating systems automatically. Similar to the modular approach for termination in Section 3, this 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 Section 3, the notion of chains has to be restricted. A sequence of dependency pairs $s_1 \to t_1$, $s_2 \to t_2$, ... is an *innermost $\mathcal{R}$-chain* if there exists a substitution σ such that all $s_j\sigma$ are in normal form and $t_j\sigma \overset{\mathsf{i}}{\to}^*_{\mathcal{R}} s_{j+1}\sigma$ holds for every two consecutive pairs $s_j \to t_j$ and $s_{j+1} \to t_{j+1}$ in the sequence. Here, '$\overset{\mathsf{i}}{\to}$' denotes innermost reductions.

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

$$\begin{aligned} \mathsf{F}(x, \mathsf{c}(x), \mathsf{c}(y)) &\to \mathsf{F}(y, y, \mathsf{f}(y, x, y)) && (21)\\ \mathsf{F}(x, \mathsf{c}(x), \mathsf{c}(y)) &\to \mathsf{F}(y, x, y) && (22)\\ \mathsf{F}(\mathsf{s}(x), y, z) &\to \mathsf{F}(x, \mathsf{s}(\mathsf{c}(y)), \mathsf{c}(z)) && (23) \end{aligned}$$

The infinite sequence consisting of the dependency pair (21) 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. Arts and Giesl [2000] proved that the absence of infinite innermost chains is a sufficient and necessary criterion for innermost termination.

Theorem 5.1 (Innermost Termination Criterion): *A TRS $\mathcal{R}$ is innermost terminating if and only if there exists no infinite innermost $\mathcal{R}$-chain.*

Analogous to Section 3, the notion of a graph is defined for innermost chains.

Definition 5.2 (Innermost dependency graph): *The* innermost dependency graph *of a TRS $\mathcal{R}$ is the directed graph whose nodes are the dependency pairs and there is an arc from $s \to t$ to $v \to w$ iff $s \to t$, $v \to w$ is an innermost chain.*

For the purpose of automation we again need an estimation, since in general it is undecidable whether two dependency pairs form an innermost chain. To this end, we again replace subterms in t with defined root symbols by new variables and check whether this modification of t unifies with v, but in contrast to Section 3 we do not rename multiple occurrences of the same variable.

Definition 5.3 (Estimated Innermost Dependency Graph): *The* estimated innermost dependency graph *of a TRS $\mathcal{R}$ is the directed graph whose nodes are the dependency pairs and there is an arc from $s \to t$ to $v \to w$ iff* $\textsc{cap}(t)$ *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 (22) to each dependency pair, from (21) to (23), and from (23) to itself.

However, there is no arc from (21) to itself, because $\textsc{cap}(\mathsf{F}(y_1, y_1, \mathsf{f}(y_1, x_1, y_1))) = \mathsf{F}(y_1, y_1, z)$ does not unify with $\mathsf{F}(x_2, \mathsf{c}(x_2), \mathsf{c}(y_2))$. Hence, the only cycles are $\{(22)\}$ and $\{(23)\}$. In fact, in this example the estimated innermost dependency graph coincides with the (real) innermost dependency graph. Similar to Theorem 3.3 one can show that it suffices to prove the absence of infinite innermost chains separately for every cycle.

Theorem 5.4 (Modular Innermost Termination Criterion): *A TRS $\mathcal{R}$ is innermost terminating if and only if for each cycle $\mathcal{P}$ in the innermost dependency graph there is no infinite innermost $\mathcal{R}$-chain of dependency pairs from $\mathcal{P}$.*

Proof: The proof is absolutely analogous to the proof of Theorem 3.3: If $\mathcal{R}$ is not innermost terminating, then by Theorem 5.1 there exists an infinite innermost chain and its tail corresponds to a cycle in the innermost dependency graph. □

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

Definition 5.5 (Usable Rules): *Let $\mathcal{R}$ be a TRS. For any symbol f let $Rules_{\mathcal{R}}(f) = \{l \to r \in \mathcal{R} \mid \mathrm{root}(l) = f\}$. For any term we define the* usable rules*:*

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

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

So we have $\mathcal{U}_{\mathcal{R}}(\mathsf{F}(y, y, \mathsf{f}(y, x, y))) = \mathrm{Rules}_{\mathcal{R}}(\mathsf{f})$ and $\mathcal{U}_{\mathcal{R}}(\{(22)\}) = \mathcal{U}_{\mathcal{R}}(\{(23)\}) = \emptyset$, i.e., there are no usable rules for the cycles. Note that $\mathrm{Rules}_{\mathcal{R}}(f) = \emptyset$ for any constructor f. Now our theorem for automatic** modular verification of innermost termination can be proved analogously to Theorem 3.5.

Theorem 5.6 (Modular Innermost Termination Proofs): *A TRS $\mathcal{R}$ is innermost terminating if for each cycle $\mathcal{P}$ in the (estimated) innermost dependency graph there is a reduction pair $(\succsim_{\mathcal{P}}, \succ_{\mathcal{P}})$ such that*

(a) $\mathcal{U}_{\mathcal{R}}(\mathcal{P}) \cup \mathcal{P} \subseteq \succsim_{\mathcal{P}}$ and
(b) $\mathcal{P} \cap \succ_{\mathcal{P}} \neq \emptyset$.

**Detailed explanations and additional refinements for the automated checking of the innermost termination criterion can be found in [Arts and Giesl, 2000, Giesl and Arts, 2001].

Proof: An infinite innermost chain of dependency pairs from some cycle $\mathcal{P}$ gives rise to an infinite sequence of inequalities in contradiction to the well-foundedness of $\succ_{\mathcal{P}}$ (similar to the proof of Theorem 3.5). The only difference is that now σ is a substitution with normal forms and therefore the reductions $w_{i,j}\sigma \overset{\mathsf{i}}{\to}{}^*_{\mathcal{R}} v_{i,j+1}\sigma$, $w_{i,n_i}\sigma \overset{\mathsf{i}}{\to}{}^*_{\mathcal{R}} s\rho_i\sigma$, and $t\rho_i\sigma \overset{\mathsf{i}}{\to}{}^*_{\mathcal{R}} v_{i+1,1}\sigma$ only require usable rules from $\mathcal{U}_{\mathcal{R}}(\mathcal{P})$. □

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

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

For $\succ_1$ we may use LPO comparing subterms right-to-left and for $\succ_2$ we may use LPO comparing subterms left-to-right. Hence, innermost termination of this example can easily be proved automatically. Without our modularity result, the above innermost termination proof would not be possible, because there exists no simplification ordering satisfying *both* inequalities (not even after argument filtering).

Note that unlike Theorem 3.5, the reverse direction of Theorem 5.6 does not hold, i.e., this criterion is only sufficient, but not necessary for innermost termination. As an example regard the TRS $\mathcal{R}$ with the rules

$$\begin{aligned} \mathsf{f}(\mathsf{a}(x), y) &\to \mathsf{g}(x, y) && (24)\\ \mathsf{g}(x, y) &\to \mathsf{h}(x, y) && (25)\\ \mathsf{h}(\mathsf{0}, y) &\to \mathsf{f}(y, y) && (26)\\ \mathsf{a}(\mathsf{0}) &\to \mathsf{0}. && (27) \end{aligned}$$

The only cycle of its innermost dependency graph is $\{\mathsf{F}(\mathsf{a}(x), y) \to \mathsf{G}(x, y), \mathsf{G}(x, y) \to \mathsf{H}(x, y), \mathsf{H}(\mathsf{0}, y) \to \mathsf{F}(y, y)\}$. In fact, this TRS is innermost terminating. However, the constraints of Theorem 5.6 imply

$$\mathsf{F}(\mathsf{a}(\mathsf{0}), \mathsf{a}(\mathsf{0})) \succsim \mathsf{G}(\mathsf{0}, \mathsf{a}(\mathsf{0})) \succsim \mathsf{H}(\mathsf{0}, \mathsf{a}(\mathsf{0})) \succsim \mathsf{F}(\mathsf{a}(\mathsf{0}), \mathsf{a}(\mathsf{0})),$$

where one of these inequalities must also hold for the strict ordering $\succ$. Thus, they are not satisfied by any reduction pair.

Of course, Criteria 3.7 and 3.8 can also be modified into sufficient criteria for innermost termination proofs as follows.

Criterion 5.7 (Modular Automated Innermost Termination Crit.): *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is innermost terminating if for each cycle $\mathcal{P}$ in the (estimated) innermost dependency graph there is an argument filtering $\pi_{\mathcal{P}}$ for $\mathcal{F}^\sharp$ and a reduction pair $(\succsim_{\mathcal{P}}, \succ_{\mathcal{P}})$ such that*

(a) $\pi_{\mathcal{P}}(\mathcal{U}_{\mathcal{R}}(\mathcal{P}) \cup \mathcal{P}) \subseteq \succsim_{\mathcal{P}}$ and
(b) $\pi_{\mathcal{P}}(\mathcal{P}) \cap \succ_{\mathcal{P}} \neq \emptyset$.

Criterion 5.8 (Innermost Termination Crit. by Transformation): *A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is innermost terminating if for each cycle $\mathcal{P}$ in the (estimated) innermost dependency graph there is an argument filtering $\pi_{\mathcal{P}}$ for $\mathcal{F}^\sharp$ such that $\pi_{\mathcal{P}}(\mathcal{U}_{\mathcal{R}}(\mathcal{P}) \cup \mathcal{P})$ is a terminating TRS and such that $\pi_{\mathcal{P}}(\mathcal{P}) \neq \emptyset$.*

6. Modularity Results for Innermost Termination

In Section 6.1 we introduce modularity criteria which can be derived from the results of the previous section. Section 6.2 compares these criteria with related work.

6.1. Modularity Criteria

In this section we present two corollaries of our results from Section 5 which are particularly useful in practice.

6.1.1. Hierarchical Combinations

A straightforward corollary of Theorems 5.4 and 5.6 can be obtained for *hierarchical combinations*. As an example consider the following TRS. Here, '$n : m : x$' abbreviates '$n : (m : x)$'. The function $\mathsf{add}(x, y)$ adds all elements of the list x to the first element of the list y, i.e., $\mathsf{add}(n_0 : n_1 : \ldots : n_k : \mathsf{nil}, m : y) = (m + \sum_{i=0}^{k} n_i) : y$. The function weight computes the weighted sum, i.e., $\mathsf{weight}(n_0 : n_1 : \ldots : n_k : \mathsf{nil}) = n_0 + \sum_{i=1}^{k} i \cdot n_i$.

$$\begin{array}{rcl}
\mathsf{add}(\mathsf{s}(n) : x, m : y) & \to & \mathsf{add}(n : x, \mathsf{s}(m) : y) \\
\mathsf{add}(\mathsf{0} : x, y) & \to & \mathsf{add}(x, y) \\
\mathsf{add}(\mathsf{nil}, y) & \to & y \\
\mathsf{weight}(n : m : x) & \to & \mathsf{weight}(\mathsf{add}(n : m : x, \mathsf{0} : x)) \\
\mathsf{weight}(n : \mathsf{nil}) & \to & n
\end{array}$$

Let $\mathcal{R}_1$ consist of the three add-rules and let $\mathcal{R}_2$ be the system consisting of the two weight-rules. Then these two systems form a hierarchical combination, where add is a defined symbol of $\mathcal{R}_1$ and a constructor of $\mathcal{R}_2$.

Note that tuple symbols from dependency pairs of $\mathcal{R}_1$ do not occur in left-hand sides of $\mathcal{R}_2$-dependency pairs. Hence, a cycle in the innermost dependency graph either consists of $\mathcal{R}_1$-dependency pairs or of $\mathcal{R}_2$-dependency pairs only. So in our example, every cycle either contains just ADD- or just WEIGHT-dependency pairs. Thus, we obtain the following corollary.[††]

COROLLARY 6.1 (INNERMOST TERM. FOR HIERARCHICAL COMBINATIONS): *Let $\mathcal{R}$ be the hierarchical combination of $\mathcal{R}_1$ and $\mathcal{R}_2$.*

(a) $\mathcal{R}$ is innermost terminating if and only if $\mathcal{R}_1$ is innermost terminating and there exists no infinite innermost $\mathcal{R}$-chain of $\mathcal{R}_2$-dependency pairs.

(b) $\mathcal{R}$ is innermost terminating if $\mathcal{R}_1$ is innermost terminating and if there exists an argument filtering π and a reduction pair $(\succsim, \succ)$ such that for all dependency pairs $s \to t$ of $\mathcal{R}_2$

[††]Of course, an obvious refinement of Corollary 6.1 (b) is to regard the different cycles of $\mathcal{R}_2$-dependency pairs in $\mathcal{R}$'s (estimated) innermost dependency graph separately. Moreover, a variant of Corollary 6.1 also holds for $\mathcal{C}_{\mathcal{E}}$-termination instead of innermost termination [Urbain, 2001].

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

Proof: The corollary is a direct consequence of Theorems 5.4 and 5.6, since every cycle consists of $\mathcal{R}_1$- or of $\mathcal{R}_2$-dependency pairs only and since for any dependency pair $s \to t$ of $\mathcal{R}_1$ the only rules that can be used to reduce a normal instantiation of t are the rules from $\mathcal{R}_1$ (i.e., $\mathcal{U}_\mathcal{R}(t) \subseteq \mathcal{R}_1$). □

(Innermost) termination of the add-system ($\mathcal{R}_1$) is easily proved (e.g., by LPO with the precedence add $>$: and add $>$ s). For the weight-subsystem ($\mathcal{R}_2$) we obtain the following constraints. (Note that $\mathsf{WEIGHT}(\ldots) \to \mathsf{ADD}(\ldots)$ is no dependency pair of $\mathcal{R}_2$, since add $\notin \mathcal{D}_2$.)

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

By choosing the argument filtering $\pi(\mathsf{add}) = \pi(:) = [2]$, the inequalities are also satisfied by LPO, but now we have to use the precedence $: >$ add.

In this way, innermost termination of this non-simply terminating example can be proved automatically. Moreover, as the system is non-overlapping, this also proves its termination. A criterion like Corollary 6.1 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 $\mathcal{R}_1$ and $\mathcal{R}_2$ either. But in contrast to the innermost termination case, rules of $\mathcal{R}_2$ can be used to reduce instantiated right-hand sides of $\mathcal{R}_1$-dependency pairs (as we cannot restrict ourselves to normal substitutions then). Hence, to prove the absence of infinite $\mathcal{R}_1$-chains we have to use a quasi-ordering where the rules of $\mathcal{R}_2$ are also weakly decreasing. Therefore, the constraints for the termination proof of the add and weight-example (according to Section 3) are not satisfied by any reduction pair with a quasi-simplification ordering amenable to automation [Arts and Giesl, 2001], 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.

6.1.2. Splitting into Subsystems

The modularity results for innermost termination presented so far were all used in the context of dependency pairs. However as already mentioned, the classical approach to modularity is to split a TRS into subsystems and to prove their (innermost) termination separately. The following corollary of Theorem 5.4 shows that the consideration of cycles in the innermost dependency graph can also be used to decompose a TRS into modular subsystems. (Similarly, the cycles of the estimated innermost dependency graph may be used as well for this decomposition.)

In the following, let $\mathcal{O}(\mathcal{P})$ denote the *origin* of the dependency pairs in $\mathcal{P}$, i.e., $\mathcal{O}(\mathcal{P})$ is a set of those rules where the dependency pairs of $\mathcal{P}$ stem from. If a dependency pair of $\mathcal{P}$ may stem from *several* rules, then it is sufficient if $\mathcal{O}(\mathcal{P})$ just contains one of them. So for the example of Section 5 we have $\mathcal{O}(\{(22)\}) = \{\mathsf{f}(x,\mathsf{c}(x),\mathsf{c}(y)) \to \mathsf{f}(y,y,\mathsf{f}(y,x,y))\}$ and $\mathcal{O}(\{(23)\}) = \{\mathsf{f}(\mathsf{s}(x),y,z) \to \mathsf{f}(x,\mathsf{s}(\mathsf{c}(y)),\mathsf{c}(z))\}$.

Corollary 6.2 (Modularity for Subsystems): *Let $\mathcal{R}$ be a TRS, let $\mathcal{P}_1, \ldots, \mathcal{P}_n$ be the cycles in its (estimated) innermost dependency graph, and let $\mathcal{R}_j$ be subsystems of $\mathcal{R}$ such that $\mathcal{U}_\mathcal{R}(\mathcal{P}_j) \cup \mathcal{O}(\mathcal{P}_j) \subseteq \mathcal{R}_j$ (for all $j \in \{1,\ldots,n\}$). If $\mathcal{R}_1, \ldots, \mathcal{R}_n$ are innermost terminating, then $\mathcal{R}$ is also innermost terminating.*

Proof: As $\mathcal{P}_j$ is a cycle, every dependency pair from $\mathcal{P}_j$ is an $\mathcal{R}_j$-dependency pair. (In order to see this, let $f^\sharp(\vec{s}) \to g^\sharp(\vec{t})$ be an $\mathcal{R}$-dependency pair in $\mathcal{P}_j$. Here, $\vec{s}$ and $\vec{t}$ denote *tuples* of terms $s_1, \ldots, s_n$ and $t_1, \ldots, t_m$, respectively. Clearly, g is a defined symbol of $\mathcal{R}_j$ because there is also a dependency pair $g^\sharp(\vec{v}) \to h^\sharp(\vec{w})$ in $\mathcal{P}_j$. Hence, since g is a defined symbol of $\mathcal{R}_j$, $f^\sharp(\vec{s}) \to g^\sharp(\vec{t})$ is also an $\mathcal{R}_j$-dependency pair.) 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 Theorem 5.4. □

For instance, in the example of Section 5 we only have two cycles, viz. $\{(22)\}$ and $\{(23)\}$. As these dependency pairs have no defined symbols in their right-hand sides, their sets of usable rules are empty. Hence, to prove innermost termination of the whole system, by Corollary 6.2 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 $5x+1$ [Lankford, 1979]. Methods for the automated generation of polynomial orderings have for instance been developed in [Steinbach, 1994, Giesl, 1995]. For the second system one can use LPO with the precedence $\mathsf{f} > \mathsf{s}$ and $\mathsf{f} > \mathsf{c}$.

Hence, the modularity criterion of Corollary 6.2 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 Section 3. Even better, if we modify the TRS into a non-overlapping one

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

then Corollary 6.2 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 LPO, but for the first rule one needs the precedence $\mathsf{c} > \mathsf{s}$ and $\mathsf{c} > \mathsf{f}$, whereas for the second rule the precedence $\mathsf{f} > \mathsf{s}$ and $\mathsf{f} > \mathsf{c}$ is required.

Note that the reverse direction of the corollary does not hold. Consider the TRS (24) - (27) from the end of Section 5 again. The only cycle of its innermost dependency graph is $\{\mathsf{F}(\mathsf{a}(x), y) \to \mathsf{G}(x, y), \mathsf{G}(x, y) \to \mathsf{H}(x, y), \mathsf{H}(\mathsf{0}, y) \to \mathsf{F}(y, y)\}$. Since this cycle does not have any usable rules, Corollary 6.2 states that innermost termination of the subsystem consisting of the first three rules is sufficient for innermost termination of the whole TRS. However, the converse does not hold, since the whole system is innermost terminating, whereas the subsystem consisting of the first three rules is not. (The term $\mathsf{f}(\mathsf{a}(\mathsf{0}), \mathsf{a}(\mathsf{0}))$ starts an infinite innermost reduction.)

6.2. 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. Section 6.2.1 focuses on composable TRSs and Section 6.2.2 gives a comparison with results on hierarchical combinations.

6.2.1. Shared Constructors and Composable Rewrite Systems

By the framework of the previous sections we can easily prove that innermost termination is modular for composable TRSs [Ohlebusch, 1995] and hence also for TRSs with disjoint sets of defined symbols and shared constructors [Gramlich, 1995]. In fact, Corollary 6.2 immediately implies[‡‡] the following result of Ohlebusch [1995].

Theorem 6.3 (Modularity for Composable TRSs): *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be composable TRSs. If $\mathcal{R}_1$ and $\mathcal{R}_2$ are innermost terminating, then $\mathcal{R}_1 \cup \mathcal{R}_2$ is also innermost terminating.*

Proof: Let $f^\sharp(\vec{s}) \to g^\sharp(\vec{t})$ be a dependency pair of $\mathcal{R}_1 \cup \mathcal{R}_2$. If $f \in \mathcal{D}_1$, then there exists a rule $f(\vec{s}) \to C[g(\vec{t})]$ in $\mathcal{R}_1$. (This rule cannot be from $\mathcal{R}_2 \setminus \mathcal{R}_1$, because $\mathcal{R}_1$ and $\mathcal{R}_2$ are composable.) Hence, $g \in \mathcal{D}_1$, because constructors of $\mathcal{R}_1$ are not defined symbols of $\mathcal{R}_2$. Similarly, $f \in \mathcal{D}_2$ implies $g \in \mathcal{D}_2$. So any dependency pair of $\mathcal{R}_1 \cup \mathcal{R}_2$ is an $\mathcal{R}_1$-dependency pair or an $\mathcal{R}_2$-dependency pair.

Moreover, there can only be an arc from $f^\sharp(\vec{s}) \to g^\sharp(\vec{t})$ to a dependency pair of the form $g^\sharp(\vec{v}) \to h^\sharp(\vec{w})$. Hence, if $f^\sharp(\vec{s}) \to g^\sharp(\vec{t})$ is an $\mathcal{R}_j$-dependency pair, then $g \in \mathcal{D}_j$ and therefore, $g^\sharp(\vec{v}) \to h^\sharp(\vec{w})$ is also an $\mathcal{R}_j$-dependency pair (for

[‡‡]A direct proof of Theorem 6.3 is not too difficult either, but our alternative proof serves to illustrate the connections between our criteria and existing modularity results.

$j \in \{1,2\}$). So every cycle $\mathcal{P}$ in the innermost dependency graph of $\mathcal{R}_1 \cup \mathcal{R}_2$ either consists of $\mathcal{R}_1$-dependency pairs or of $\mathcal{R}_2$-dependency pairs only.

If a cycle $\mathcal{P}$ only contains $\mathcal{R}_1$-dependency pairs, then $\mathcal{R}_1$ is a superset of $\mathcal{U}_{\mathcal{R}_1\cup\mathcal{R}_2}(\mathcal{P}) \cup \mathcal{O}(\mathcal{P})$, as the defined symbols of $\mathcal{R}_2 \setminus \mathcal{R}_1$ do not occur as constructors in $\mathcal{R}_1$. Similarly, for a cycle $\mathcal{P}$ of $\mathcal{R}_2$-dependency pairs, we have $\mathcal{U}_{\mathcal{R}_1\cup\mathcal{R}_2}(\mathcal{P}) \cup \mathcal{O}(\mathcal{P}) \subseteq \mathcal{R}_2$. Hence by Corollary 6.2, $\mathcal{R}_1 \cup \mathcal{R}_2$ is innermost terminating if $\mathcal{R}_1$ and $\mathcal{R}_2$ are innermost terminating. □

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 Section 6.1.2.

6.2.2. Proper Extensions

Krishna Rao [1995] proved that innermost termination is modular for (generalized) *proper* extensions which are a certain kind of hierarchical combinations. In this section we show that for finite TRSs this is also a direct consequence of our results.

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

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ form a hierarchical combination. Now the defined symbols $\mathcal{D}_2$ of $\mathcal{R}_2$ are split in two sets $\mathcal{D}_2^1$ and $\mathcal{D}_2^2$, where $\mathcal{D}_2^1$ contains all defined symbols which depend on a defined symbol of $\mathcal{R}_1$, i.e., $\mathcal{D}_2^1 = \{f | f \in \mathcal{D}_2, f \trianglerighteq_d g \text{ for some } g \in \mathcal{D}_1\}$ and $\mathcal{D}_2^2 = \mathcal{D}_2 \setminus \mathcal{D}_2^1$. $\mathcal{R}_2$ is a *proper extension* of $\mathcal{R}_1$ if every rule $l \to r \in \mathcal{R}_2$ satisfies the following condition: Whenever t is a subterm of r such that $\text{root}(t) \in \mathcal{D}_2^1$ and $\text{root}(t) \trianglerighteq_d \text{root}(l)$, then t contains no function symbol depending on $\mathcal{D}_1$ (i.e., from $\mathcal{D}_1 \cup \mathcal{D}_2^1$) except at its root.

For instance, in the add and weight-example from Section 6.1.1 we have $\mathcal{D}_1 = \{\mathsf{add}\}$, $\mathcal{D}_2^1 = \{\mathsf{weight}\}$ (because weight depends on the definition of add), and $\mathcal{D}_2^2 = \emptyset$. This example is not a proper extension, because there is a weight-rule in which the $\mathcal{D}_1$-symbol add occurs below the $\mathcal{D}_2^1$-symbol weight. Thus, in a proper extension functions depending on $\mathcal{R}_1$ are never called within a recursive call of $\mathcal{R}_2$-functions. As an example for a proper extension consider the TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ from the end of Section 4.3 again, where $\mathcal{R}_2$ is extended by the rule

$$\mathsf{avg}(l) \;\to\; \mathsf{quot}(\mathsf{hd}(\mathsf{sum}(l)), \mathsf{length}(l)).$$

Here, $\mathsf{avg}(l)$ computes the average of all elements in the list l. We have $\mathcal{D}_2^1 = \{\mathsf{avg}\}$, whereas all other symbols of $\mathcal{D}_2$ belong to $\mathcal{D}_2^2$. Since avg does not occur in a right-hand side, this modified TRS $\mathcal{R}_2$ is a proper extension of $\mathcal{R}_1$. The modified TRS $\mathcal{R}_2$ is still DP simply terminating (since the avg-rule does not give rise to additional dependency pairs). In fact, its innermost termination also follows directly from Corollary 6.1 (b), since the original TRS $\mathcal{R}_2$ and the avg-rule form a hierarchical combination. Corollaries 6.1 and 6.2 imply the following result of [Krishna Rao, 1995] which in turn ensures that the union of $\mathcal{R}_1$ and the extended system $\mathcal{R}_2$ in our example is innermost terminating.

Theorem 6.4 (Modularity for Proper Extensions): *Let $\mathcal{R}_2$ be a proper extension of $\mathcal{R}_1$. The TRS $\mathcal{R}_1 \cup \mathcal{R}_2$ is innermost terminating if $\mathcal{R}_1$ and $\mathcal{R}_2$ are innermost terminating.*

Proof: As in the proof of Corollary 6.1, since $\mathcal{R}_1$ and $\mathcal{R}_2$ form a hierarchical combination, every cycle in the innermost dependency graph of $\mathcal{R}_1 \cup \mathcal{R}_2$ consists solely of $\mathcal{R}_1$-dependency pairs or of $\mathcal{R}_2$-dependency pairs. If a cycle $\mathcal{P}$ consists of dependency pairs of $\mathcal{R}_1$, we have $\mathcal{U}_{\mathcal{R}_1 \cup \mathcal{R}_2}(\mathcal{P}) \cup \mathcal{O}(\mathcal{P}) \subseteq \mathcal{R}_1$, because dependency pairs of $\mathcal{R}_1$ do not contain any defined symbols of $\mathcal{R}_2$.

Otherwise, the cycle $\mathcal{P}$ consists of $\mathcal{R}_2$-dependency pairs. If $f^\sharp(\vec{s}) \to g^\sharp(\vec{t})$ is an $\mathcal{R}_2$-dependency pair in $\mathcal{P}$, then there exists a rule $f(\vec{s}) \to C[g(\vec{t})]$ in $\mathcal{R}_2$ and $f, g \in \mathcal{D}_2$. In addition, we have $f \trianglerighteq_d g$ and $g \trianglerighteq_d f$ (as $\mathcal{P}$ is a cycle).

If $g \in \mathcal{D}_2^2$, then f also belongs to $\mathcal{D}_2^2$, hence no defined symbol of $\mathcal{D}_1 \cup \mathcal{D}_2^1$ occurs in $\vec{t}$. Otherwise, if $g \in \mathcal{D}_2^1$, then by definition of a proper extension again all defined symbols in $\vec{t}$ are from $\mathcal{D}_2^2$. Thus, in both cases, all defined symbols of $\mathcal{U}_{\mathcal{R}_1 \cup \mathcal{R}_2}(g^\sharp(\vec{t}))$ belong to $\mathcal{D}_2^2$. Hence, $\mathcal{U}_{\mathcal{R}_1 \cup \mathcal{R}_2}(g^\sharp(\vec{t}))$ is a subsystem of $\mathcal{R}_2$.

So for any cycle $\mathcal{P}$ of $\mathcal{R}_2$-dependency pairs, we have $\mathcal{U}_{\mathcal{R}_1 \cup \mathcal{R}_2}(\mathcal{P}) \cup \mathcal{O}(\mathcal{P}) \subseteq \mathcal{R}_2$. Hence, by Corollary 6.2 innermost termination of $\mathcal{R}_1$ and $\mathcal{R}_2$ implies innermost termination of $\mathcal{R}_1 \cup \mathcal{R}_2$. $\square$

As another example regard the system $\mathcal{R}_0$ consisting of the following three rules.

$$\begin{aligned} \mathsf{hd}(x : l) &\to x \\ \mathsf{length}(\mathsf{nil}) &\to \mathsf{0} \\ \mathsf{length}(x : l) &\to \mathsf{s}(\mathsf{length}(l)) \end{aligned}$$

The TRS $\mathcal{R}_1 \cup \mathcal{R}_2$ (including the avg-rule) is a proper extension of $\mathcal{R}_0$ and therefore, Theorem 6.4 also implies innermost termination of $\mathcal{R}_0 \cup \mathcal{R}_1 \cup \mathcal{R}_2$.

The notions of "composability" and "proper extension" can be combined as follows. Suppose we are given two TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $\mathcal{D}_1 = \mathcal{D}'_1 \uplus \mathcal{D}'$, $\mathcal{D}_2 = \mathcal{D}'_2 \uplus \mathcal{D}'$, $\mathcal{R}_1 \cap \mathcal{R}_2 = \{l \to r \in \mathcal{R} \mid \mathrm{root}(l) \in \mathcal{D}'\}$, and $\mathcal{D}'_1 \cap \mathcal{D}'_2 = \mathcal{C}_1 \cap \mathcal{D}'_2 = \emptyset$.

Now $\mathcal{D}_2$ is split in two sets $\mathcal{D}_2^1$ and $\mathcal{D}_2^2$, where $\mathcal{D}_2^1 = \{f | f \in \mathcal{D}_2, f \trianglerighteq_d g \text{ for some } g \in \mathcal{D}'_1\}$ and $\mathcal{D}_2^2 = \mathcal{D}_2 \setminus \mathcal{D}_2^1$. $\mathcal{R}_2$ is a *generalized proper extension* [Krishna Rao, 1995] of $\mathcal{R}_1$ if every rewrite rule $l \to r \in \mathcal{R}_2$ satisfies the following condition: Whenever t is a subterm of r such that $\mathrm{root}(t) \in \mathcal{D}_2^1 \setminus \mathcal{D}'$ and $\mathrm{root}(t) \trianglerighteq_d \mathrm{root}(l)$, then t contains no function symbol depending on $\mathcal{D}'_1$ (i.e., from $\mathcal{D}'_1 \cup \mathcal{D}_2^1$) except at its root.

As an example, we again regard the TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ from the end of Section 4.3, where $\mathcal{R}_2$ also contains the rule for avg, $\mathcal{R}_1$ also contains the rule $(x - y) - z \to$

$x-(y+z)$, and both $\mathcal{R}_1$ and $\mathcal{R}_2$ are augmented by the additional rules $0+y \to y$ and $\mathsf{s}(x)+y \to \mathsf{s}(x+y)$, cf. Section 4.4.

$$
\begin{array}{lrcl@{\qquad}rcl}
\mathcal{R}_1: & x - \mathsf{0} & \to & x & \mathsf{0}+y & \to & y \\
& \mathsf{s}(x) - \mathsf{s}(y) & \to & x-y & \mathsf{s}(x)+y & \to & \mathsf{s}(x+y) \\
& \mathsf{quot}(\mathsf{0},\mathsf{s}(y)) & \to & \mathsf{0} & (x-y)-z & \to & x-(y+z) \\
& \mathsf{quot}(\mathsf{s}(x),\mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{quot}(x-y,\mathsf{s}(y))) & & &
\end{array}
$$

$$
\begin{array}{lrcl@{\qquad}rcl}
\mathcal{R}_2: & \mathsf{app}(\mathsf{nil},k) & \to & k & \mathsf{avg}(l) & \to & \mathsf{quot}(\mathsf{hd}(\mathsf{sum}(l)),\mathsf{length}(l)) \\
& \mathsf{app}(l,\mathsf{nil}) & \to & l & \mathsf{0}+y & \to & y \\
& \mathsf{app}(x:l,k) & \to & x:\mathsf{app}(l,k) & \mathsf{s}(x)+y & \to & \mathsf{s}(x+y) \\
& \mathsf{sum}(x:\mathsf{nil}) & \to & x:\mathsf{nil} & & & \\
& \mathsf{sum}(x:(y:l)) & \to & \mathsf{sum}((x+y):l) & & & \\
& \mathsf{sum}(\mathsf{app}(l,x:(y:k))) & \to & \mathsf{sum}(\mathsf{app}(l,\mathsf{sum}(x:(y:k)))) & & &
\end{array}
$$

Now we have $\mathcal{D}' = \{+\}$, $\mathcal{D}'_1 = \{-,\mathsf{quot}\}$, $\mathcal{D}'_2 = \{\mathsf{app},\mathsf{sum},\mathsf{avg}\}$, where $\mathcal{D}_2^1 = \{\mathsf{avg}\}$ and $\mathcal{D}_2^2 = \{+,\mathsf{app},\mathsf{sum}\}$. Thus, $\mathcal{R}_2$ is indeed a generalized proper extension of $\mathcal{R}_1$ and as both systems are innermost terminating (and even DP simply terminating), the following theorem allows us to conclude innermost termination of their union. Moreover, the union of this system with $\mathcal{R}_0$ is again innermost terminating by Theorem 6.4.

Theorem 6.5 (Modularity for Generalized Proper Extensions): *Let $\mathcal{R}_2$ be a generalized proper extension of $\mathcal{R}_1$. The TRS $\mathcal{R}_1 \cup \mathcal{R}_2$ is innermost terminating if $\mathcal{R}_1$ and $\mathcal{R}_2$ are innermost terminating.*

Proof: At first, we observe the following fact: If $f^\sharp(\vec{s}) \to g^\sharp(\vec{t})$ is a dependency pair with $f \in \mathcal{D}_1$, then $g \in \mathcal{D}_1$ because the rewrite rule $f(\vec{s}) \to C[g(\vec{t})]$ occurs in $\mathcal{R}_1$ and $\mathcal{D}'_2$-symbols are not allowed in $\mathcal{R}_1$. Moreover, $\mathcal{U}_{\mathcal{R}_1\cup\mathcal{R}_2}(g^\sharp(\vec{t})) \subseteq \mathcal{R}_1$, since all rules for the defined symbols in $\vec{t}$ are (also) contained in $\mathcal{R}_1$. So for any cycle $\mathcal{P}$ of $\mathcal{R}_1 \cup \mathcal{R}_2$ containing a dependency pair $f^\sharp(\ldots) \to g^\sharp(\ldots)$ with $f \in \mathcal{D}_1$, we obtain $\mathcal{U}_{\mathcal{R}_1\cup\mathcal{R}_2}(\mathcal{P}) \cup \mathcal{O}(\mathcal{P}) \subseteq \mathcal{R}_1$.

For all other dependency pairs $f^\sharp(\vec{s}) \to g^\sharp(\vec{t})$ on some cycle $\mathcal{P}$ we have $f \in \mathcal{D}'_2$. Hence, there is a rule $f(\vec{s}) \to C[g(\vec{t})]$ in $\mathcal{R}_2$. Note that $g \in \mathcal{D}'_2$ as well, otherwise the dependency pair $f^\sharp(\ldots) \to g^\sharp(\ldots)$ would not be on a cycle. As in the proof of Theorem 6.4 we have $f \trianglerighteq_d g \trianglerighteq_d f$.

If $g \in \mathcal{D}_2^2$, then we also have $f \in \mathcal{D}_2^2$ and thus, no symbol of $\mathcal{D}'_1 \cup \mathcal{D}_2^1$ occurs in $\vec{t}$. Similarly, if $g \in \mathcal{D}_2^1$ then this implies $g \in \mathcal{D}_2^1 \setminus \mathcal{D}'$ (since $g \in \mathcal{D}'_2$). By the definition of generalized proper extensions, $\vec{t}$ again contains no symbols of $\mathcal{D}'_1 \cup \mathcal{D}_2^1$, i.e., all defined symbols in $\vec{t}$ are from $\mathcal{D}_2^2$. Hence, we obtain $\mathcal{U}_{\mathcal{R}_1\cup\mathcal{R}_2}(\mathcal{P}) \cup \mathcal{O}(\mathcal{P}) \subseteq \mathcal{R}_2$. Therefore, innermost termination of $\mathcal{R}_1$ and $\mathcal{R}_2$ implies innermost termination of $\mathcal{R}_1 \cup \mathcal{R}_2$ by Corollary 6.2. $\square$

To summarize, we have shown that our results (in particular, Corollary 6.2) directly imply several modularity results for innermost termination from the literature. On the other hand, our modularity results significantly extend the class of TRSs where innermost termination can be proved in a modular way. In other words, they can handle many systems where all previously known criteria for modularity of innermost termination fail.

For example, we can deal with combinations which are neither composable nor hierarchical combinations (nor generalized proper extensions) as shown in Section 6.1.2. This is not possible with any of the previous modularity results. Moreover, in contrast to [Krishna Rao, 1995], our results are also applicable for hierarchical combinations in which $\mathcal{R}_2$ contains defined symbols of $\mathcal{R}_1$ in the arguments of its recursive calls, cf. the add and weight-example. Such systems occur frequently in practice.

Another modularity criterion for hierarchical combinations is due to Dershowitz [1994]. There, occurrences of $\mathcal{D}_1$-symbols in recursive calls of $\mathcal{D}_2$-symbols are allowed, but only if $\mathcal{R}_2$ is *oblivious* of the $\mathcal{R}_1$-rules, i.e., termination of $\mathcal{R}_2$ must not depend on the $\mathcal{R}_1$-rules. However, this criterion is not applicable to systems like the add and weight-example, because termination of the weight-rules of course depends on the result of $\mathsf{add}(n : m : x, \mathsf{0} : x)$.

An alternative modularity result for hierarchical combinations was presented by Fernández and Jouannaud [1995]. However, their result is restricted to systems where the arguments of recursive calls in $\mathcal{R}_2$ decrease w.r.t. the subterm relation (compared as multisets or lexicographically). Hence, their result is not applicable to the add and weight-example either (and also not to most other systems where $\mathcal{R}_2$ is not simply terminating), whereas our modularity results are often successful in these examples.

7. Conclusion

In this article 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 for which such proofs were not possible before. For a collection of such examples see [Arts and Giesl, 2001].

Using our modular refinement of the dependency pair framework, we developed several new modularity criteria which extend previous results for modularity of innermost termination. Within this framework, we also obtain easy proofs for existing modularity theorems.

However, criteria for *innermost* termination are only applicable for termination proofs of certain restricted TRSs (e.g., locally confluent overlay systems and in particular, non-overlapping systems [Gramlich, 1995]). But in practice there are many cases in which innermost termination is not sufficient for termination.

Thus, to fully exploit the advantages of dependency pairs for these systems as well, we showed that the well-known modularity result for simple termination of disjoint unions can be extended to DP quasi-simple termination. Furthermore,

$\mathcal{G}$-restricted DP simple termination is even modular for constructor-sharing and composable systems.

To conclude, [Arts and Giesl, 2000] presented the dependency pair technique to perform automated termination and innermost termination proofs. However, in that article dependency pairs were not used in a modular way and thus one had to prove termination of a TRS at once (i.e., without being able to decompose it into subsystems and to use several different orderings for its termination proof). In particular, whenever a TRS was constructed by combining several systems whose termination had been proved before, then the whole termination proof had to be re-done.

Therefore, the present article develops the ideas of [Arts and Giesl, 2000] further in a significant way. The progress in automated termination proving which was made possible by the development of dependency pairs now also has a counterpart in the area of modularity. With dependency pairs one can obtain automated termination proofs of non-simply terminating TRSs and with the results of the present article one can perform them in a modular way. In fact, it is this modularity which makes an application of dependency pairs to large and realistic systems possible; see [Giesl and Arts, 2001] for an industrial case study. Compared to previous work on modularity, the modularity criteria developed in this article represent a substantial extension.

Acknowledgements. We thank Aart Middeldorp for many helpful remarks and hints. Jürgen Giesl was supported by the DFG under grant GI 274/4-1.

References

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

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

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

T. Arts and J. Giesl. A collection of examples for termination of term rewriting using dependency pairs. Technical Report AIB 2001-09, RWTH Aachen, Germany, 2001. http://aib.informatik.rwth-aachen.de.

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

CiME 2, 1999. Available at http://cime.lri.fr.

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

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

N. Dershowitz. Hierarchical termination. In *Proceedings of the 4th International Workshop on Conditional and Typed Rewriting Systems, CTRS-94*, volume 968 of *Lecture Notes in Computer Science*, pages 89–105, Jerusalem, Israel, 1994. Springer Verlag, Berlin.

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

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

J. Dick, J. Kalmus, and U. Martin. Automating the Knuth-Bendix ordering. *Acta Informatica*, 28:95–119, 1990.

M. Fernández and J.-P. Jouannaud. Modular termination of term rewriting systems revisited. In *Proceedings of the 10th Workshop on Specification of Abstract Data Types*, volume 906 of *Lecture Notes in Computer Science*, pages 255–273, S. Margherita, Italy, 1995. Springer Verlag, Berlin.

M. Ferreira. *Termination of Term Rewriting – Well-Foundedness, Totality, and Transformations.* PhD thesis, University of Utrecht, The Netherlands, 1995.

M. Ferreira and H. Zantema. Syntactical analysis of total termination. In *Proceedings of the 4th International Conference on Algebraic and Logic Programming, ALP-94*, volume 850 of *Lecture Notes in Computer Science*, pages 204–222, Madrid, Spain, 1994. Springer Verlag, Berlin.

J. Giesl. Generating polynomial orderings for termination proofs. In *Proceedings of the 6th International Conference on Rewriting Techniques and Applications, RTA-95*, volume 914 of *Lecture Notes in Computer Science*, pages 426–431, Kaiserslautern, Germany, 1995. Springer Verlag, Berlin.

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

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

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

B. Gramlich. Generalized sufficient conditions for modular termination of rewriting. *Applicable Algebra in Engineering, Communication, and Computing*, 5: 131–158, 1994.

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

B. Gramlich. On proving termination by innermost termination. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications, RTA-96*, volume 1103 of *Lecture Notes in Computer Science*, pages 93–107, New Brunswick, NJ, USA, 1996a. Springer Verlag, Berlin.

B. Gramlich. *Termination and Confluence Properties of Structured Rewrite Systems.* PhD thesis, Universität Kaiserslautern, Germany, 1996b.

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

S. Kamin and J.-J. Lévy. Two generalizations of the recursive path ordering. Department of Computer Science, University of Illinois, IL, USA, 1980.

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

D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

K. Korovin and A. Voronkov. Verifying orientability of rewrite rules using the Knuth-Bendix order. In *Proceedings of the 5th International Conference on Rewriting Techniques and Applications, RTA-01*, volume 2051 of *Lecture Notes in Computer Science*, pages 137–153, Utrecht, The Netherlands, 2001. Springer Verlag, Berlin.

M. R. K. Krishna Rao. Simple termination of hierarchical combinations of term rewriting systems. In *Proceedings of the Symposium on Theoretical Aspects of Computer Software, TACS-94*, volume 789 of *Lecture Notes in Computer Science*, pages 203–223, Sendai, Japan, 1994. Springer Verlag, Berlin.

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

M. Kurihara and A. Ohuchi. Modularity of simple termination of term rewriting systems with shared constructors. *Theoretical Computer Science*, 103:273–282, 1992.

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

D. S. Lankford. On proving term rewriting systems are Noetherian. Technical Report Memo MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.

A. Middeldorp. A sufficient condition for the termination of the direct sum of term rewriting systems. In *Proceedings of the 4th Annual Symposium on Logic in Computer Science, LICS-89*, pages 396–401, Pacific Grove, CA, USA, 1989.

A. Middeldorp. *Modular Properties of Term Rewriting Systems.* PhD thesis, Vrije Universiteit te Amsterdam, The Netherlands, 1990.

A. Middeldorp. Approximating dependency graphs using tree automata techniques. In *Proceedings of the First International Joint Conference on Automated Reasoning, IJCAR 2001*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 593–610, Siena, Italy, 2001. Springer Verlag, Berlin.

A. Middeldorp and H. Ohsaki. Type introduction for equational rewriting. *Acta Informatica*, 36(12):1007–1029, 2000.

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

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

E. Ohlebusch. *Modular Properties of Composable Term Rewriting Systems.* PhD thesis, Universität Bielefeld, Germany, 1994a.

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

E. Ohlebusch. Modular properties of composable term rewriting systems. *Journal of Symbolic Computation*, 20:1–41, 1995.

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

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

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

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

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

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

Y. Toyama, J. W. Klop, and H. Barendregt. Termination for the direct sum of left-linear term rewriting systems. *Journal of the ACM*, 42:1275–1304, 1995.

X. Urbain. Automated incremental termination proofs for hierarchically defined term rewriting systems. In *Proceedings of the First International Joint Conference on Automated Reasoning, IJCAR 2001*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 485–498, Siena, Italy, 2001. Springer Verlag, Berlin.

H. Zantema. Termination of term rewriting: Interpretation and type elimination. *Journal of Symbolic Computation*, 17:23–50, 1994.