Pushing the Frontiers of Combining Rewrite Systems Farther Outwards[*]

Jürgen Giesl[†] Enno Ohlebusch[‡]

Abstract

It is well known that simple termination is *modular* for certain kinds of combinations of term rewriting systems (TRSs). This result is of practical relevance because most techniques for (automated) termination proofs use *simplification orderings*, so they show in fact simple termination. On the other hand, in practice many systems are non-simply terminating. In order to cope with such systems, Arts and Giesl developed the *dependency pair* approach. By using (quasi-)simplification orderings in combination with dependency pairs, it is possible to prove termination of non-simply terminating systems *automatically.* It is natural to ask whether modularity of simple termination can be extended to the class of those systems which can be handled by this technique. In this paper we show that this is indeed the case. In this way, the class of TRSs for which termination can be proved in a modular way is extended significantly.

1 Introduction

Modularity is a well-known paradigm in computer science. Programs should be designed in a modular way, that is, as a combination of small programs. These so-called modules are implemented separately and are then integrated to form the whole program. Since TRSs have important applications in computer science, it is essential to know under which conditions a combined system inherits desirable properties from its constituent systems. For this reason modular aspects of term rewriting have been studied extensively. A property $\mathcal{P}$ of TRSs (like termination) is called *modular* if whenever $\mathcal{R}_1$ and $\mathcal{R}_2$ are TRSs both satisfying $\mathcal{P}$, then their combined system $\mathcal{R}_1 \cup \mathcal{R}_2$ also satisfies $\mathcal{P}$. The knowledge that (perhaps under certain conditions) a property $\mathcal{P}$ is modular facilitates software engineering because it allows an incremental development of programs. On the other hand, it provides

[*]To appear in the *Proceedings of FroCoS '98*, Amsterdam, The Netherlands, Logic and Computation Series, Research Studies Press, John Wiley & Sons, 2000.

[†]Department of Computer Science, Darmstadt University of Technology, Alexanderstr. 10, 64283 Darmstadt, Germany, Email: giesl@informatik.tu-darmstadt.de

[‡]Technische Fakultät, University of Bielefeld, P.O. Box 10 01 31, 33501 Bielefeld, Germany, Email: enno@TechFak.Uni-Bielefeld.DE

a divide and conquer approach to establishing properties of TRSs. If one wants to know whether a large TRS has a certain modular property $\mathcal{P}$, then this system can be decomposed into small subsystems and one merely has to check whether each of these subsystems has property $\mathcal{P}$.

As all interesting properties are in general not modular, the starting-point of research were disjoint unions, i.e. combinations of TRSs without common function symbols. Toyama [1987b] proved that confluence is modular for disjoint systems, but termination and completeness lack a modular behavior [Toyama, 1987a]. 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, 1996] for an overview. However, 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 methods for *automated* termination proofs work with so-called *simplification* orderings [Dershowitz, 1987; Steinbach, 1995; Middeldorp and Zantema, 1997], Kurihara and Ohuchi's [1992] result for constructor-sharing systems is thus of practical relevance. They showed that the combination of finite simply terminating TRSs (systems whose termination can be verified by a simplification ordering) 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].

However, there are numerous relevant TRSs where simplification orderings fail in proving termination. For that purpose, a new technique for automated termination proofs, viz. the so-called *dependency pair* approach, was developed by Arts and Giesl [1997a; 1997b; 1997c; 1998]. Given a TRS, this approach 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 techniques can often generate 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, 1997c].

Thus, the dependency pair approach pushed the frontier of those TRSs whose termination is provable automatically a lot further. Now the class of TRSs where automated termination proofs are (potentially) feasible are no longer just the simply terminating systems, but the *DP-(quasi) simply* *terminating* systems, i.e. those systems whose termination can be verified by using simplification orderings in combination with dependency pairs. Hence, a natural question is whether the current frontier of modularity can be pushed further as well by extending the modularity results from simple to DP-(quasi) simple termination. In this paper, we will show that this is indeed possible. Thus, the class of TRSs whose termination can be proved in a modular way is extended considerably.

The paper is organized as follows: First we briefly recall the basic notions of the combination of TRSs. Sect. 3 contains a short description of the dependency pair method. In Sect. 4 we introduce the concept of DP-(quasi) simple termination and show in Sect. 5 that DP-quasi simple termination is modular for disjoint unions. Sect. 6 contains similar results about constructor-sharing TRSs.

2 Basic Notions of the Union of TRSs

For an introduction to term rewriting see e.g. [Dershowitz and Jouannaud, 1990; Klop, 1992]. Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$. A function symbol $f \in \mathcal{F}$ is called a *defined symbol* if there is a rewrite rule $l \to r \in \mathcal{R}$ such that $f = root(l)$. Function symbols from $\mathcal{F}$ which are not defined symbols are called *constructors.* Thus, if a TRS consists of the following two rules

$$f(0,1,x) \to f(s(x),x,x) \tag{1}$$
$$f(x,y,s(z)) \to s(f(0,1,z)), \tag{2}$$

then f is the only defined symbol, whereas 0, 1, and s are constructors.

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be TRSs over the signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, resp. Their *combined system* is their 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: $\{l \to r \in \mathcal{R} \mid root(l) \in \mathcal{D}_1 \cap \mathcal{D}_2\} \subseteq \mathcal{R}_1 \cap \mathcal{R}_2$.

We next give a brief overview of the basic notions of disjoint unions. In the sequel let $t \in \mathcal{T}(\mathcal{F}_1 \cup \mathcal{F}_2, \mathcal{V})$. Let $\Box$ be a special constant $\notin \mathcal{F}_1 \cup \mathcal{F}_2$. 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 $root(t_1), \ldots root(t_n) \in \mathcal{F}_{3-i}$ for some $i \in \{1, 2\}$. In this case, the t_j are the *principal* subterms of t and C is the topmost $\mathcal{F}_i$-homogeneous part of t, denoted by $top_i(t)$ (whereas $top_{3-i}(t)$ is $\Box$). So for example, if $\mathcal{R}_1$ consists of the rules (1) and (2), and $\mathcal{R}_2$ contains the rules

$$g(x,y) \to x \tag{3}$$
$$g(x,y) \to y, \tag{4}$$

then $\mathcal{R}_1$ and $\mathcal{R}_2$ are disjoint and a term like $f(g(0,0),x,g(y,y))$ can be written as $C[\![g(0,0),g(y,y)]\!]$, where C is $f(\Box,x,\Box)$. Thus $top_1(f(g(0,0),x,g(y,y))) = f(\Box,x,\Box)$ and $top_2(f(g(0,0),x,g(y,y))) = \Box$.

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

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

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

A rewrite step $s \to_{\mathcal{R}_1\cup\mathcal{R}_2} t$ is *destructive at level 1* if $root(s) \in \mathcal{F}_i$ and $root(t) \in \mathcal{F}_{3-i}$ 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 $f(g(0,0),x,g(y,y)) \to f(0,x,g(y,y))$ is destructive at level 2.

Finally, we recall that for every signature $\mathcal{F}$ the TRS $\mathcal{E}mb(\mathcal{F})$ (which is important in the context of simple termination) is defined by

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

3 Dependency Pairs

In the dependency pair approach of Arts and Giesl [1997a; 1997c; 1998] for showing termination, if $f(s_1,...,s_n)$ rewrites to $C[g(t_1,...,t_m)]$ (where g is a defined symbol), then one has to compare the argument tuples $s_1,...,s_n$ and $t_1,...,t_m$. To avoid the handling of *tuples*, a new *tuple symbol* $F \notin \mathcal{F}$ is introduced for every defined symbol f. Instead of comparing *tuples*, now the *terms* $F(s_1,...,s_n)$ and $G(t_1,...,t_m)$ are compared. Thus, to ease readability we assume that the signature $\mathcal{F}$ consists of lower case symbols only and that tuple symbols are denoted by the corresponding upper case symbols.

Definition 1 (Dependency Pair) *If $f(s_1,...,s_n) \to C[g(t_1,...,t_m)]$ is a rule of a TRS $\mathcal{R}$ and g is a defined symbol, then $\langle F(s_1,...,s_n), G(t_1,...t_m)\rangle$ is a* dependency pair *of $\mathcal{R}$.*

So for the TRS $\mathcal{R}_1 = \{(1),(2)\}$ we obtain the following dependency pairs

$$\langle F(0,1,x), F(s(x),x,x)\rangle \tag{5}$$
$$\langle F(x,y,s(z)), F(0,1,z)\rangle. \tag{6}$$

To trace those subterms which may start new reductions, we examine special sequences of dependency pairs, so-called *chains.* In the following, we consider substitutions whose domains may be infinite and assume that different (occurrences of) dependency pairs have disjoint sets of variables.

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

For instance, in our example we have the chain

$\langle F(0,1,x_1), F(s(x_1),x_1,x_1)\rangle \;\; \langle F(x_2,y_2,s(z_2)), F(0,1,z_2)\rangle \;\; \langle F(0,1,x_3), F(s(x_3),x_3,x_3)\rangle$

because with $\sigma = \{x_1 \mapsto s(x_3), x_2 \mapsto s(s(x_3)), y_2 \mapsto s(x_3), z_2 \mapsto x_3\}$ we have $F(s(x_1),x_1,x_1)\sigma \to_{\mathcal{R}_1}^* F(x_2,y_2,s(z_2))\sigma$ and $F(0,1,z_2)\sigma \to_{\mathcal{R}_1}^* F(0,1,x_3)\sigma$. In fact, every finite alternating sequence of (5) and (6) is a chain. Arts and Giesl [1997a; 1997c] proved that the absence of *infinite* chains is a sufficient and necessary criterion for termination.

Theorem 3 (Termination Criterion) *A TRS $\mathcal{R}$ is terminating if and only if there exists no infinite $\mathcal{R}$-chain.*

Note that the first dependency pair (5) can never follow itself in a chain, because $F(s(x_1),x_1,x_1)\sigma \to_{\mathcal{R}_1}^* F(0,1,x_2)\sigma$ does not hold for any substitution σ. To estimate which dependency pairs may occur consecutive in a chain, the *estimated dependency graph* has been introduced, cf. Arts and Giesl [1997a; 1997c; 1998]. We first recall the needed notions. CAP(t) results from replacing all subterms of t that have a defined root symbol by different fresh variables and REN(t) results from replacing all variables in t by different fresh variables. Then, in order to determine whether $\langle u, v\rangle$ can follow $\langle s, t\rangle$ in a chain, we check whether REN(CAP(t)) unifies with u. The function REN is needed to rename multiple occurrences of the same variable x in t because when instantiated with σ, two occurrences of $x\sigma$ could reduce to different terms. So in our example, the estimated dependency graph contains an arc from (5) to (6) and arcs from (6) to (5) and to itself.

Definition 4 (Estimated Dependency Graph) *The* estimated dependency graph *is the directed graph whose nodes are the dependency pairs and there is an arc from $\langle s, t\rangle$ to $\langle u, v\rangle$ if REN(CAP(t)) and u are unifiable.*

A set $\mathcal{P}$ of dependency pairs is called a *cycle* if for any two dependency pairs $\langle s, t\rangle, \langle u, v\rangle \in \mathcal{P}$ there is a path from $\langle s, t\rangle$ to $\langle u, v\rangle$ and from $\langle u, v\rangle$ to $\langle s, t\rangle$ in the estimated dependency graph which traverses dependency pairs from $\mathcal{P}$ only. (In particular, there must also be a path from $\langle s, t\rangle$ to itself.) Thus, the only non-empty cycles in our example are $\{(6)\}$ and $\{(5),(6)\}$. In the remainder of the paper, we always restrict ourselves to finite TRSs (and to finite signatures). Then any infinite chain corresponds to a cycle, i.e. it suffices to prove that there is no infinite chain of dependency pairs *from any cycle*, cf. [Arts and Giesl, 1998].

For an automation of this criterion, we generate a set of inequalities such that the existence of a well-founded *quasi-ordering* satisfying these inequalities is sufficient for the absence of infinite chains. As usual, a *quasi-ordering* $\succsim$ is a reflexive and transitive relation. The corresponding *strict* relation $\succ^s$ is defined as $t \succ^s u$ iff $t \succsim u$ and $u \not\succsim t$. Moreover, we also define a corresponding *stable-strict* relation $\succ^{ss}$ as $t \succ^{ss} u$ iff $t\sigma \succ^s u\sigma$ 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 $t \succsim u$ iff $|t\sigma| \geq_{\mathbb{N}} |u\sigma|$ holds for all ground substitutions σ. A natural way to define a corresponding irreflexive ordering $\succ$ is to let $t \succ u$ hold iff $|t\sigma| >_{\mathbb{N}} |u\sigma|$ for all ground substitutions σ. However, now $\succ$ is not the corresponding strict relation, but the corresponding stable-strict relation of $\succsim$. 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 under substitutions, then the corresponding stable-strict relation $\succ^{ss}$ is stable under substitutions too, whereas this is not necessarily true for the strict relation $\succ^s$.

For example, if $|a| = 0$, $|f_1(t)| = |t|$, and $|f_2(t)| = 2|t|$ for all ground terms t, then we have $f_2(x) \succsim f_1(x)$ and $f_1(x) \not\succsim f_2(x)$. Hence, this implies $f_2(x) \succ^s f_1(x)$. However, $\succ^s$ is not stable under substitutions because $f_2(a) \succ^s f_1(a)$ 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 $f_2(x) \not\succ^{ss} f_1(x)$.

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 $h(a) \to a$ and $\succsim$ is defined as $\to^*_{\mathcal{R}}$, then we have $h(x) \succ^{ss} x$, but $h(x) \not\succsim x$.

The following lemma states some straightforward properties of stable-strict relations, where in the following we always assume that our signature contains at least one constant (i.e. that there exist ground terms).

Lemma 5 (Properties of Stable-Strict Relations) *Let $\succsim$ be a quasi-ordering that is stable under substitutions. Then we have*

(i) $\succ^{ss}$ *is irreflexive*
(ii) $\succ^{ss}$ *is transitive*
(iii) $\succ^{ss}$ *is stable under substitutions*
(iv) *if* $\succ^s$ *is stable under substitutions, then* $\succ^s \subseteq \succ^{ss}$
(v) *if* $\succ^s$ *is well founded, then* $\succ^{ss}$ *is well founded, too*

(vi) $s \succsim t \succ^{ss} u$ *implies* $s \succ^{ss} u$
(vii) $s \succ^{ss} t \succsim u$ *implies* $s \succ^{ss} u$.

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

In the following, instead of the corresponding strict relations we always consider the corresponding stable-strict relations of quasi-orderings $\succsim$. For the sake of brevity, we write $\succ$ instead of $\succ^{ss}$, i.e. in this paper $\succ$ always denotes the stable-strict relation corresponding to $\succsim$. Analogously, we will call a quasi-ordering *well-founded* if the corresponding stable-strict relation is well founded.

The following theorem is from [Arts and Giesl, 1998], where instead of the strict relation corresponding to the quasi-ordering we now use the stable-strict relation. Note that the present formulation of Thm. 6 with stable-strict relations is more powerful than the formulation with strict relations. To use the strict relation $\succ^{s}$ of a quasi-ordering in Thm. 6, $\succ^{s}$ would have to be stable under substitutions; cf. [Arts and Giesl, 1998, Thm. 6]. But then by Lemma 5 (iv), $s \succ^{s} t$ always implies $s \succ^{ss} t$. Hence, all constraints satisfied by $\succ^{s}$ are satisfied by the corresponding stable-strict relation $\succ^{ss}$ as well. Using Lemma 5 (vii), the proof for the if-part of this slightly modified theorem is identical to the corresponding one in [Arts and Giesl, 1998]. The proof for the only-if-part can be found in [Arts and Giesl, 1997c].

Theorem 6 (Dependency Pair Approach) *A TRS $\mathcal{R}$ is terminating iff for each cycle $\mathcal{P}$ in the estimated dependency graph there is a well-founded weakly monotonic quasi-ordering $\succsim$ stable under substitutions such that*

- $l \succsim r$ *for all rules* $l \to r$ *in* $\mathcal{R}$,
- $s \succsim t$ *for all dependency pairs* $\langle s,t\rangle$ *from* $\mathcal{P}$,
- $s \succ t$ *for at least one dependency pair* $\langle s,t\rangle$ *from* $\mathcal{P}$.

Thus, to prove the absence of infinite chains from the cycle $\{(6)\}$ we have to find a quasi-ordering satisfying

$$F(x,y,s(z)) \succ F(0,1,z) \tag{7}$$
$$f(0,1,x) \succsim f(s(x),x,x) \tag{8}$$
$$f(x,y,s(z)) \succsim s(f(0,1,z)). \tag{9}$$

4 DP-(quasi) simple termination

As mentioned, our aim is to use standard techniques to generate a suitable quasi-ordering satisfying the constraints of Thm. 6. However, most existing methods generate orderings which are *strongly* monotonic, whereas for the dependency pair approach we only need a *weakly* monotonic ordering. For that reason, before synthesizing a suitable ordering, some of the arguments of the function symbols can be eliminated, cf. Arts and Giesl [1997a; 1997c]. For instance, one may eliminate the first two arguments of the function symbol f. Then every term $f(t_1, t_2, t_3)$ in the inequalities is replaced by $f'(t_3)$, where f' is a new function symbol. So instead of (8) and (9) we would obtain the inequalities $f'(x) \succsim f'(x)$ and $f'(s(z)) \succsim s(f'(z))$. Now the resulting constraints are satisfied by the recursive path ordering (rpo) with the precedence $f' \rhd s \rhd 0 \rhd 1$. Similarly, (by eliminating the first two arguments of F) one can also prove the absence of infinite chains from the cycle $\{(5), (6)\}$. Hence, termination of the TRS consisting of the rules (1) and (2) is proved. Note that this TRS is not simply terminating. So in the dependency pair approach, simplification orderings like the rpo can be used to prove termination of TRSs where 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 first two arguments of f, one could replace all terms $f(t_1, t_2, t_3)$ by f's third argument t_3. Then the resulting inequalities are again satisfied by the rpo. To perform this elimination of arguments resp. of function symbols the following concept was introduced in [Arts and Giesl, 1997c].

Definition 7 (AFS) *An* argument filtering system[1] *(AFS) over $\mathcal{F}$ is a TRS whose rewrite rules are of the form*

$$f(x_1, \ldots, x_n) \to r$$

with $f \in \mathcal{F}$ and there is at most one such rule for every $f \in \mathcal{F}$. Here $x_1, \ldots, x_n$ are pairwise distinct variables and r is either one of these variables or it is a term $f'(y_1, \ldots, y_m)$, where $f' \notin \mathcal{F}$ is a fresh function symbol and $y_1, \ldots, y_m$ are pairwise distinct variables out of $x_1, \ldots, x_n$.

As proved in [Arts and Giesl, 1997c], in order to find a quasi-ordering satisfying a particular set of inequalities, one may first normalize the terms in the inequalities with respect to an AFS (where the AFS may also contain rules for the tuple symbols). Subsequently, one only has to find a quasi-ordering that satisfies these modified inequalities. Hence, by combining the synthesis of a suitable AFS with well-known techniques for the generation of (*strongly* monotonic) simplification orderings, now the search for a *weakly* monotonic ordering satisfying the constraints can be automated.

In this paper, we impose a (minor) restriction[2] on the AFSs used, viz. we restrict ourselves to AFSs $\mathcal{A}$ such that

[1]AFSs are a special form of recursive program schemes [Courcelle, 1990; Klop, 1992].

[2]This restriction is not very severe. If there exists a quasi-simplification ordering satisfying the constraints in Thm. 6 and if these constraints include at least one strict inequality

- $\mathcal{V}ar(r\downarrow_{\mathcal{A}}) \subseteq \mathcal{V}ar(l\downarrow_{\mathcal{A}})$ and $l\downarrow_{\mathcal{A}} \notin \mathcal{V}$ or $l\downarrow_{\mathcal{A}} = r\downarrow_{\mathcal{A}}$ for all rules $l \to r$ in $\mathcal{R}$
- $\mathcal{V}ar(t\downarrow_{\mathcal{A}}) \subseteq \mathcal{V}ar(s\downarrow_{\mathcal{A}})$ and $s\downarrow_{\mathcal{A}} \notin \mathcal{V}$ or $s\downarrow_{\mathcal{A}} = t\downarrow_{\mathcal{A}}$ for all $\langle s, t\rangle$ in cycles

As already mentioned, most methods for the automated generation of well-founded orderings construct simplification orderings or quasi-simplification orderings [Dershowitz, 1987; Steinbach, 1995; Middeldorp and Zantema, 1997]. Here we use the following definition of [Middeldorp and Zantema, 1997]: A *simplification ordering* is an ordering (i.e. an irreflexive and transitive relation) that is monotonic (closed under contexts), closed under substitutions, and possesses the subterm property. It is a well-known consequence of Kruskal's theorem that every simplification ordering on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is well founded provided that $\mathcal{F}$ is finite.[3]

Analogously, a *quasi-simplification ordering* (qso) is a quasi-ordering which is (weakly) monotonic, closed under substitutions, and has the (weak) subterm property. Since we restrict ourselves to finite signatures, every quasi-simplification ordering (more precisely, the corresponding stable-strict relation) is well founded, too.

Examples of simplification orderings and qso's include path orderings like the rpo, the lexicographic path ordering (lpo), etc. [Dershowitz, 1987; Steinbach, 1995]. Polynomial orderings, however, are not qso's in general. For instance, if the constant 0 is associated with the number 0, $s(x)$ is associated with $x + 1$, and $f(x, y)$ is associated with the multiplication of x and y, then this polynomial ordering does not satisfy the subterm property (for example, $f(s(0), 0) \succsim s(0)$ does not hold). However, the following lemma shows that if the polynomial ordering respects some restrictions, then it is indeed a qso.

Lemma 8 (Polynomial Orderings as qso's) *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 containing all variables $x_1, \ldots, x_n$ and if every constant is associated with a number > 0, then $\succsim$ is a qso.*
- *If every function symbol $f(x_1, \ldots, x_n)$ is associated with a polynomial which contains a (non-mixed) monomial of the form $m\, x_i^k$ (with $m, k \geq 1$) for every $i = 1, \ldots, n$, then $\succsim$ is a qso.*

Proof. Straightforward. □

In fact, whenever polynomial orderings can be used in connection with the dependency pair approach, one can usually apply a polynomial ordering

with variables in its right-hand side, then $\mathcal{V}ar(r\downarrow_{\mathcal{A}}) \subseteq \mathcal{V}ar(l\downarrow_{\mathcal{A}})$ and $\mathcal{V}ar(t\downarrow_{\mathcal{A}}) \subseteq \mathcal{V}ar(s\downarrow_{\mathcal{A}})$ are always satisfied, because otherwise the constraints would imply $t \succ x$ for some term t with $x \notin \mathcal{V}ar(t)$.

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

which satisfies one of the above conditions. By restricting ourselves to qso's, we obtain the following restricted notion of termination. Again, $\succ$ denotes the stable-strict relation corresponding to $\succsim$.

Definition 9 (DP-quasi simple termination) *A TRS $\mathcal{R}$ is* DP-quasi simply terminating *iff for every non-empty cycle $\mathcal{P}$ in the estimated dependency graph there exists an AFS $\mathcal{A}$ and a qso $\succsim$ such that*

(a) $l\downarrow_{\mathcal{A}} \succsim r\downarrow_{\mathcal{A}}$ for all rules $l \to r$ in $\mathcal{R}$,
(b) $s\downarrow_{\mathcal{A}} \succsim t\downarrow_{\mathcal{A}}$ for all dependency pairs $\langle s,t\rangle$ from $\mathcal{P}$,
(c) $s\downarrow_{\mathcal{A}} \succ t\downarrow_{\mathcal{A}}$ for at least one dependency pair $\langle s,t\rangle$ from $\mathcal{P}$.

Definition 9 captures all TRSs where an automated termination proof using dependency pairs is potentially feasible. In fact, there are numerous DP-quasi simply terminating TRSs which are not simply terminating; cf. e.g. the collection in [Arts and Giesl, 1997c]. This observation motivated the development of the dependency pair approach and it also motivated the present work, as our aim is to extend well-known modularity results for simple termination to DP-quasi simple termination.

A straightforward way to generate a qso $\succeq$ from a simplification ordering $\succ$ is to define $t \succeq u$ iff $t \succ u$ 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 qso's, we obtain the notion of DP-simple termination.

Definition 10 (DP-simple termination) *A TRS $\mathcal{R}$ is* DP-simply terminating *iff for every non-empty cycle $\mathcal{P}$ in the estimated dependency graph there is an AFS $\mathcal{A}$ and a simplification ordering $\succ$ such that*

(a) $l\downarrow_{\mathcal{A}} \succeq r\downarrow_{\mathcal{A}}$ for all rules $l \to r$ in $\mathcal{R}$,
(b) $s\downarrow_{\mathcal{A}} \succeq t\downarrow_{\mathcal{A}}$ for all dependency pairs $\langle s,t\rangle$ from $\mathcal{P}$,
(c) $s\downarrow_{\mathcal{A}} \succ t\downarrow_{\mathcal{A}}$ for at least one dependency pair $\langle s,t\rangle$ from $\mathcal{P}$.

Note that (a) - (c) are equivalent to simple termination of the TRS

$$\begin{aligned}\mathcal{S}_{\mathcal{P}} \;=\; & \{l\downarrow_{\mathcal{A}} \to r\downarrow_{\mathcal{A}} \mid\; l \to r \in \mathcal{R} \text{ and } l\downarrow_{\mathcal{A}} \neq r\downarrow_{\mathcal{A}}\} \cup \\ & \{s\downarrow_{\mathcal{A}} \to t\downarrow_{\mathcal{A}} \mid\; \langle s,t\rangle \text{ is a dependency pair from } \mathcal{P} \text{ and } s\downarrow_{\mathcal{A}} \neq t\downarrow_{\mathcal{A}}\}\end{aligned}$$

provided that $s\downarrow_{\mathcal{A}} \neq t\downarrow_{\mathcal{A}}$ holds for at least one dependency pair $\langle s,t\rangle \in \mathcal{P}$.

It turns out that most of the examples in [Arts and Giesl, 1997c] are not only DP-quasi simply terminating but even DP-simply terminating. The following lemma illustrates the connections between the different notions.

Lemma 11 (Characterizing DP-(quasi) simple termination)
simple termination $\Rightarrow$ DP-simple termination $\Rightarrow$ DP-quasi simple termination $\Rightarrow$ termination

Proof. The second implication holds as $\succ$ is stable under substitutions and therefore $\succ$ is contained in the stable-strict relation of $\succeq$, cf. Lemma 5 (iv). The last implication follows from Thm. 6 by using the quasi-ordering $\succeq'$ where $u \succeq' v$ holds iff $u\downarrow_{\mathcal{A}} \succeq v\downarrow_{\mathcal{A}}$.

It remains to show the first implication. Let $\mathcal{R}$ be a simply terminating TRS over the signature $\mathcal{F} = \mathcal{C} \cup \mathcal{D}$ and let $\mathcal{T}up_{\mathcal{F}} = \{F \mid f \in \mathcal{D}\}$ be the set of tuple symbols. 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 in a term $s \in \mathcal{T}(\mathcal{F} \cup \mathcal{T}up_{\mathcal{F}}, \mathcal{V})$ replaces every tuple symbol F with its corresponding function symbol $f \in \mathcal{F}$. Then $\succ$ can be extended to a simplification ordering $\succ'$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{T}up_{\mathcal{F}}, \mathcal{V})$ by defining $t \succ' u$ iff $\Omega(t) \succ \Omega(u)$ holds. We claim that the simplification ordering $\succ'$ satisfies the constraints (a) - (c) of Def. 10 without applying an AFS.

Obviously, $l \succ' r$ holds for all rules $l \to r$ of $\mathcal{R}$. Thus $\succ'$ satisfies the constraint (a). Moreover, for every dependency pair $\langle s, t\rangle$ we have $s \succ' t$. The reason is that each dependency pair $\langle F(s_1, \ldots, s_n), G(t_1, \ldots, t_m)\rangle$ 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 $f(\ldots) \succ g(\ldots)$ which in turn implies $F(\ldots) \succ' G(\ldots)$. Hence, $\succ'$ also satisfies the constraints (b) and (c) of Def. 10. □

The following examples show that none of the converse implications of Lemma 11 holds.

Example 12 *The system $\{f(f(x)) \to f(c(f(x)))\}$ is DP-simply terminating as the only dependency pair on a cycle is $\langle F(f(x)), F(x)\rangle$. Hence, the resulting constraints are satisfied by the rpo if one uses the AFS $c(x) \to x$. However, this TRS is not simply terminating. The TRS*

$$\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl}
f(f(x)) & \to & f(c(f(x))) & g(c(x)) & \to & x & g(c(0)) & \to & g(d(1)) \\
f(f(x)) & \to & f(d(f(x))) & g(d(x)) & \to & x & g(c(1)) & \to & g(d(0))
\end{array}$$

is DP-quasi simply terminating as can be proved in a similar way using the AFS with the rules $c(x) \to x$ and $d(x) \to x$ and the 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 AFS must reduce $c(x)$ and $d(x)$ to their arguments. But then $g(0) \geq g(1)$ and $g(1) \geq g(0)$ lead to a contradiction.*

Finally, the system $\{f(0, 1, x) \to f(x, x, x)\}$ is terminating but not DP-quasi simply terminating. □

5 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 13 (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. In this case, 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

$$top_1(u\sigma) \to_{\mathcal{R}_1} top_1(v\sigma) \to^*_{\mathcal{R}_1} top_1(u\sigma)$$

is an $\mathcal{R}_1$-reduction sequence of ground terms over $\mathcal{F}_1$. 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 $\tau = \{x_i \mapsto top_1(x_i\sigma) \mid 1 \leq i \leq n\}$ and indeed

$$u\tau = top_1(u\sigma) \to_{\mathcal{R}_1} top_1(v\sigma) = v\tau \to^*_{\mathcal{R}_1} top_1(u\sigma) = u\tau$$

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, Thm. 3.13], the reduction sequence D can be transformed by a transformation function[4] 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 $root(t) \in \mathcal{F}_1$ 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 constant, and denote the

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

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 $\mathrm{Cons}(x_1, x_2) \to x_j$. □

Now we are in a position to prove our first modularity theorem.

Theorem 14 (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 iff 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 AFS $\mathcal{A}$ such that inequalities (a) - (c) of Def. 9 are satisfied for $\mathcal{R}_1$, $\mathcal{P}$, and some qso $\succsim$. Let $\mathcal{F}_1'$ be the set of all function symbols occurring in the inequalities (a) - (c). Without loss of generality we may assume that $\mathcal{A}$ contains no rules with root symbols from $\mathcal{F}_2$. Now let[5]

$$\begin{aligned}
\mathcal{S}_1 &= \{l\downarrow_{\mathcal{A}} \to r\downarrow_{\mathcal{A}} \mid l \to r \in \mathcal{R}_1 \text{ and } l\downarrow_{\mathcal{A}} \notin \mathcal{V}\} \cup \mathcal{E}mb(\mathcal{F}_1') \cup \\
&\quad\ \{s\downarrow_{\mathcal{A}} \to t\downarrow_{\mathcal{A}} \mid \langle s, t\rangle \in \mathcal{P} \text{ and } s\downarrow_{\mathcal{A}} \notin \mathcal{V}\} \\
\mathcal{S}_2 &= \mathcal{R}_2 \cup \mathcal{E}mb(\mathcal{F}_2).
\end{aligned}$$

$\mathcal{S}_1$ is a TRS over the signature $\mathcal{F}_1'$. Hence $\mathcal{R}' = \mathcal{S}_1 \cup \mathcal{S}_2$ is a TRS over $\mathcal{F}_1' \cup \mathcal{F}_2$. Note that $\to^*_{\mathcal{R}'}$ is a qso.[6] Thus, (as the cycle $\mathcal{P}$ was chosen arbitrarily) to prove DP-quasi simple termination of $\mathcal{R}$, we only have to show

(a) $l\downarrow_{\mathcal{A}} \to^*_{\mathcal{R}'} r\downarrow_{\mathcal{A}}$ for all rules $l \to r$ in $\mathcal{R}$
(b) $s\downarrow_{\mathcal{A}} \to^*_{\mathcal{R}'} t\downarrow_{\mathcal{A}}$ for all dependency pairs $\langle s, t\rangle$ from $\mathcal{P}$
(c) there exists a dependency pair $\langle s, t\rangle$ from $\mathcal{P}$ such that
  $t\downarrow_{\mathcal{A}} \sigma \not\to^*_{\mathcal{R}'} s\downarrow_{\mathcal{A}} \sigma$ holds for all ground substitutions σ.

[5] $l\downarrow_{\mathcal{A}} = x \in \mathcal{V}$ implies $l\downarrow_{\mathcal{A}} = x = r\downarrow_{\mathcal{A}}$ and $x \succsim x$ is satisfied by every qso $\succsim$.

[6] 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 $\to^*_{\mathcal{R} \cup \mathcal{E}mb(\mathcal{F})} \subseteq \succsim$). Note however, that the *strict* part of a qso $\to^*_{\mathcal{R} \cup \mathcal{E}mb(\mathcal{F})}$ is not necessarily closed under substitutions. Hence, without the extension of the dependency pair approach in Thm. 6 and Def. 9 to *stable-strict* relations, such a qso cannot be used for termination proofs with dependency pairs. As this extension leads to a more powerful criterion, we did not investigate whether Thm. 14 would also hold for a formulation of Def. 9 with strict instead of stable-strict relations.

Conditions (a) and (b) are obviously satisfied as for all $l \to r \in \mathcal{R}_2$ we have $l \downarrow_{\mathcal{A}} = l$ and $r \downarrow_{\mathcal{A}} = r$. Hence, we only have to show conjecture (c). 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 $\langle s, t \rangle$ be a dependency pair from $\mathcal{P}$ such that $s \downarrow_{\mathcal{A}} \succ t \downarrow_{\mathcal{A}}$. Suppose that there exists a ground substitution $\sigma : \mathcal{V}ar(s \downarrow_{\mathcal{A}}) \to \mathcal{T}(\mathcal{F}'_1 \cup \mathcal{F}_2)$ such that $t \downarrow_{\mathcal{A}} \sigma \to^*_{\mathcal{R}'} s \downarrow_{\mathcal{A}} \sigma$. By Lemma 13, this implies the existence of a ground substitution $\tau : \mathcal{V}ar(s \downarrow_{\mathcal{A}}) \to \mathcal{T}(\mathcal{F}'_1)$ such that $t \downarrow_{\mathcal{A}} \tau \to^*_{\mathcal{S}_1} s \downarrow_{\mathcal{A}} \tau$. This, however, would imply $t \downarrow_{\mathcal{A}} \tau \succsim s \downarrow_{\mathcal{A}} \tau$ and contradicts the fact that $s \downarrow_{\mathcal{A}} \succ t \downarrow_{\mathcal{A}}$. Thus, $t \downarrow_{\mathcal{A}} \sigma \not\to^*_{\mathcal{R}'} s \downarrow_{\mathcal{A}} \sigma$ holds for all ground substitutions σ. This proves conjecture (c). □

Thus, if $\mathcal{R}_1$ is the TRS consisting of the rules (1) and (2) and $\mathcal{R}_2$ contains the rules (3) and (4), 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 $f(0,1,x) \to f(s(x),x,x)$ would be just slightly changed to $f(0,1,x) \to f(x,x,x)$, then $\mathcal{R}_1$ would still be terminating, but the union with $\mathcal{R}_2$ would not terminate any more, cf. [Toyama, 1987a].

DP-quasi simply terminating systems occur frequently in practice. For example, the two TRSs

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

are not simply terminating, but they are both DP-quasi simply terminating, cf. Arts and Giesl [1997c; 1997a]. Hence, Thm. 14 now also allows to conclude DP-quasi simple termination of their union.

6 Combining Constructor-Sharing Systems

It may be a bit surprising that Thm. 14 cannot be directly extended to constructor-sharing TRSs (even if we disallow the use of AFSs). 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 15 *Consider the following TRSs:*

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

The TRS $\mathcal{R}_1$ is simply terminating and $\mathcal{R}_2$ is DP-quasi simply terminating. The latter can be shown without any AFS by using a polynomial ordering which maps c, b, g, and G to the identity and which maps $d(x)$ to $x+1$. However, their union is not DP-quasi simply terminating. As $\langle F(c(x)), F(x)\rangle$ represents a cycle in the estimated dependency graph one would have to find a qso satisfying

$$\begin{array}{rcll} F(c(x)) & \succ & F(x) & (10) \\ f(c(x)) & \succsim & f(x) & (11) \\ f(b(x)) & \succsim & x & (12) \\ g(d(x)) & \succsim & g(x) & (13) \\ g(c(x)) & \succsim & c(g(b(c(x)))). & (14) \end{array}$$

With empty AFS, no qso satisfies (10) - (14), since otherwise we would have

$$\begin{array}{rcll} F(c(g(c(x)))) & \succ & F(g(c(x))) & \textit{due to (10)} \\ & \succsim & F(c(g(b(c(x))))) & \textit{due to (14)} \\ & \succsim & F(c(g(c(x)))) & \textit{due to the subterm property.} \end{array}$$

By (10), the AFS cannot contain any c-rules. If the argument of b would be eliminated, then (12) would be transformed into $f(b') \succsim x$. But as there exists the strict inequality (10) with a variable in its right-hand side, this results in the contradiction $F(c(f(b'))) \succ x$. Similarly, the argument of g cannot be eliminated either, since $g' \succsim c(g')$ would be a contradiction to (10).

Thus, the only possible rules in the AFS are f- and d-rules and rules that map b or g to its argument. But then we would again obtain $F(c(g(c(x)))) \succ F(c(g(c(x))))$ or $F(c(c(x))) \succ F(c(c(x)))$ as above. Hence, the TRS indeed is not DP-quasi simply terminating. □

Thus, in the remainder of the section we will restrict ourselves to DP-simple termination instead of DP-quasi simple termination. Without applying AFSs, DP-simple termination of the TRS $\mathcal{R}_2$ cannot be proved. However, if one uses for example the AFS $b(x) \to b'$, then its resulting constraints are again satisfied by a simplification ordering. Thus, to achieve modularity results for constructor-sharing TRSs, we have to restrict ourselves to systems where the AFSs contain no rules for shared symbols like b.

But we need another requirement to ensure modularity. For example, let us eliminate the first rule $g(d(x)) \to g(x)$ from $\mathcal{R}_2$. Now there is no non-empty 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 AFSs, 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 non-empty 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 Def. 10 should also be satisfied for the *empty* cycle $\mathcal{P}$.[7]

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.

Example 16 *Let $\mathcal{R}_1$ consist of the rules*

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

and let $\mathcal{R}_2$ consist of the rule $h(1) \to h(0)$. To prove DP-simple termination of $\mathcal{R}_1$ we have to use an AFS containing the rule $g(x) \to x$. This, however, would imply $0 \succ 1$ which is a contradiction to $h(1) \succeq h(0)$. Thus, the combination of both systems is not DP-simply terminating. □

So we also have to ensure that an application of the AFS 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).[8] The restrictions needed are captured by the notion of *C-restricted* DP-simple termination.

Definition 17 (C-restricted DP-simple termination) *Let $\mathcal{R}$ be a TRS over $\mathcal{F}$ and let $C \subseteq \mathcal{F}$. $\mathcal{R}$ is C-*restricted *DP-simply terminating iff*

(1) For every cycle $\mathcal{P}$ (including $\mathcal{P} = \emptyset$), there is an AFS $\mathcal{A}$ such that

$$\begin{array}{rcl} \mathcal{S} & = & \{l \downarrow_{\mathcal{A}} \to r \downarrow_{\mathcal{A}} \mid l \to r \in \mathcal{R} \text{ and } l \downarrow_{\mathcal{A}} \neq r \downarrow_{\mathcal{A}}\} \cup \\ & & \{s \downarrow_{\mathcal{A}} \to t \downarrow_{\mathcal{A}} \mid \langle s,t \rangle \text{ is a dependency pair in } \mathcal{P} \text{ and } s \downarrow_{\mathcal{A}} \neq t \downarrow_{\mathcal{A}}\} \end{array}$$

is simply terminating and such that $s \downarrow_{\mathcal{A}} \neq t \downarrow_{\mathcal{A}}$ for some $\langle s,t \rangle \in \mathcal{P}$ if $\mathcal{P} \neq \emptyset$. (Equivalently, some simplification ordering $\succ$ satisfies the constraints (a), (b), and (c) of Def. 10 for every cycle, where (a) is also required for $\mathcal{P} = \emptyset$).

(2) For every rule $u \downarrow_{\mathcal{A}} \to v \downarrow_{\mathcal{A}}$ in $\mathcal{S}$: if $root(u) \notin C$, then $root(u \downarrow_{\mathcal{A}}) \notin C$.

(3) None of the AFSs contains a rule for a function symbol $f \in C$.

The following theorem shows that under this C-restriction, DP-simple termination is modular for constructor-sharing TRSs.

[7]In fact, without this additional requirement DP-simple termination is not even modular for *disjoint* combinations. For reasons of space, a counterexample is omitted.

[8]If all AFSs used are non-collapsing, then this requirement is always fulfilled.

Theorem 18 (Modularity of C-restricted DP-simple termination) *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be constructor-sharing TRSs over the signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. If $\mathcal{F}_1 \cap \mathcal{F}_2 \subseteq C$, then their combined system $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ over the signature $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ is C-restricted DP-simply terminating if and only if both $\mathcal{R}_1$ and $\mathcal{R}_2$ are C-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). It is not difficult to prove that $\mathcal{P}$ 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 constructor-sharing. Without loss of generality let $\mathcal{P}$ be a cycle in the estimated dependency graph of $\mathcal{R}_1$. We have to show that there is an AFS $\mathcal{A}$ such that the corresponding TRS $\mathcal{S}$ is simply terminating and moreover conditions (2) and (3) of Def. 17 are satisfied.

Since $\mathcal{R}_1$ is C-restricted DP-simply terminating and $\mathcal{P}$ is a cycle in the estimated dependency graph of $\mathcal{R}_1$, there is an AFS $\mathcal{A}_1$ such that the TRS

$$\begin{aligned}\mathcal{S}_1 \;=\; & \{l\downarrow_{\mathcal{A}_1} \to r\downarrow_{\mathcal{A}_1} \mid\ l \to r \in \mathcal{R}_1 \text{ and } l\downarrow_{\mathcal{A}_1} \neq r\downarrow_{\mathcal{A}_1}\} \cup \\ & \{s\downarrow_{\mathcal{A}_1} \to t\downarrow_{\mathcal{A}_1} \mid\ \langle s,t\rangle \text{ is a dependency pair in } \mathcal{P} \text{ and } s\downarrow_{\mathcal{A}_1} \neq t\downarrow_{\mathcal{A}_1}\}\end{aligned}$$

is simply terminating and such that $s\downarrow_{\mathcal{A}_1} \neq t\downarrow_{\mathcal{A}_1}$ holds for at least one dependency pair $\langle s,t\rangle$ from $\mathcal{P}$ if $\mathcal{P} \neq \emptyset$. On the other hand, there is an AFS $\mathcal{A}_2$ such that the TRS

$$\mathcal{S}_2 = \{l\downarrow_{\mathcal{A}_2} \to r\downarrow_{\mathcal{A}_2} \mid\ l \to r \in \mathcal{R}_2 \text{ and } l\downarrow_{\mathcal{A}_2} \neq r\downarrow_{\mathcal{A}_2}\}$$

is simply terminating because $\mathcal{R}_2$ is C-restricted DP-simply terminating. (Here, $\mathcal{A}_2$ is the AFS corresponding to the empty cycle.)

Let $\mathcal{F}_i'$ be the set of all function symbols in $\mathcal{S}_i$ for $i \in \{1,2\}$ and let $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$. Since $\mathcal{R}_1$ and $\mathcal{R}_2$ are C-restricted DP-simply terminating, $\mathcal{A}$ does not contain rules for elements of C. Consequently, it follows for every term $u \in \mathcal{T}(\mathcal{F}_i, \mathcal{V})$ that $u\downarrow_{\mathcal{A}} = u\downarrow_{\mathcal{A}_i}$. So in particular, $s\downarrow_{\mathcal{A}_1} \neq t\downarrow_{\mathcal{A}_1}$ implies $s\downarrow_{\mathcal{A}} \neq t\downarrow_{\mathcal{A}}$. Moreover, by requirements (2) and (3), the TRSs

$$\begin{aligned}\mathcal{S}_1 \;=\; & \{l\downarrow_{\mathcal{A}} \to r\downarrow_{\mathcal{A}} \mid\ l \to r \in \mathcal{R}_1 \text{ and } l\downarrow_{\mathcal{A}} \neq r\downarrow_{\mathcal{A}}\} \cup \\ & \{s\downarrow_{\mathcal{A}} \to t\downarrow_{\mathcal{A}} \mid\ \langle s,t\rangle \text{ is a dependency pair in } \mathcal{P} \text{ and } s\downarrow_{\mathcal{A}} \neq t\downarrow_{\mathcal{A}}\}, \\ \mathcal{S}_2 \;=\; & \{l\downarrow_{\mathcal{A}} \to r\downarrow_{\mathcal{A}} \mid\ l \to r \in \mathcal{R}_2 \text{ and } l\downarrow_{\mathcal{A}} \neq r\downarrow_{\mathcal{A}}\}\end{aligned}$$

are also constructor-sharing. This is because a former constructor can only be a defined symbol in $\mathcal{S}_1$ or $\mathcal{S}_2$ if it is not in C, i.e. if it is not shared. Thus by [Kurihara and Ohuchi, 1992, Thm. 3.10], their combined system $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ is also simply terminating.

Since the AFS $\mathcal{A}$ and the TRS $\mathcal{S}$ obviously satisfy the conditions (2) and (3) of Def. 17, $\mathcal{R}$ is C-restricted DP-simply terminating. □

For example, let $\mathcal{R}_2$ be the (simply terminating) TRS

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

As the TRS $\mathcal{R}_1$ for subtraction and division from the end of Sect. 5 is DP-simply terminating, now Thm. 18 allows us to conclude DP-simple termination of the combined system $\mathcal{R}_1 \cup \mathcal{R}_2$, since these two TRSs are constructor-sharing (they both contain the same constructors 0 and s).

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 the 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 19 *Let $\mathcal{R}_1$ be the TRS*

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

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

$\mathcal{R}_1$ is DP-simply terminating (using the AFS $d(x) \to x$ and the lpo comparing left-to-right), but it is not simply terminating. $\mathcal{R}_2$ is even simply terminating as can be shown with the lpo comparing right-to-left. Thus, DP-simple termination of both systems can be verified by the lpo.

By Thm. 18 their union is also DP-simply terminating. However, the constraints for the cycle $\{\langle G(c(x,s(y))), G(c(s(x),y))\rangle\}$ are not satisfied by any lpo (nor by any rpo nor by any polynomial ordering). Thus, there are indeed TRSs where termination of the subsystems can be shown with dependency pairs and the lpo, but (without our modularity result) termination of their union cannot be proved with dependency pairs and the lpo. $\square$

We stress that Thm. 18 can be extended to composable systems. The proof is almost identical because simple termination is also modular for composable systems [Ohlebusch, 1995, Thm. 5.16]. Only the proof that $\mathcal{S}_1$ and $\mathcal{S}_2$ are indeed composable needs an extra effort. Therefore, C-restricted DP-simple termination is even modular for combinations of TRSs which have common defined symbols provided that both systems contain the same rules for shared defined symbols.

7 Conclusion

We have shown that the existing modularity results for simple termination of disjoint unions can be extended to DP-quasi simple termination. Under certain restrictions a similar modularity result also holds for constructor-sharing (and even composable) combinations. Future work will include an examination of whether such an extension is also possible for hierarchical combinations. In summary, 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. Dependency pairs enable automated termination proofs of non-simply terminating TRSs and now our results allow to perform them in a modular way.

The present work completes the results in [Arts and Giesl, 1998], where the theory of dependency pairs was refined in a modular way (cf. Thm. 6) and where several new modularity criteria for *innermost* termination were presented. However, these criteria are only applicable for termination proofs of 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. Up to now, due to missing modularity results, the advantages of dependency pairs could not be fully exploited for these systems. So compared to previous work on modularity, the modularity criteria developed in the present paper represent a significant extension.

Acknowledgements. We thank Aart Middeldorp for many helpful remarks and hints. This work was partially supported by the DFG under grant Wa 652/7-2 as part of the focus program "Deduktion".

References

[Arts and Giesl, 1997a] T. Arts and J. Giesl. Automatically Proving Termination Where Simplification Orderings Fail. In *Proc. TAPSOFT '97*, pages 261–272. LNCS 1214, 1997.

[Arts and Giesl, 1997b] T. Arts and J. Giesl. Proving Innermost Normalisation Automatically. In *Proc. RTA '97*, pages 157–171. LNCS 1232, 1997.

[Arts and Giesl, 1997c] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. Technical Report IBN 97/46, TU Darmstadt, 1997. http://www.inferenzsysteme.informatik.tu-darmstadt.de/~reports/notes/ibn-97-46.ps. Revised version to appear in *Theoretical Computer Science.*

[Arts and Giesl, 1998] T. Arts and J. Giesl. Modularity of Termination Using Dependency Pairs. In *Proc. RTA '98*, pages 226–240. LNCS 1379, 1998.

[Courcelle, 1990] B. Courcelle. Recursive Applicative Program Schemes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 459–492. North-Holland, 1990.

[Dershowitz and Jouannaud, 1990] N. Dershowitz and J.-P. Jouannaud. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–320. North-Holland, 1990.

[Dershowitz, 1987] N. Dershowitz. Termination of Rewriting. *Journal of Symbolic Computation*, 3:69–115, 1987.

[Gramlich, 1994] B. Gramlich. Generalized Sufficient Conditions for Modular Termination of Rewriting. *Applicable Algebra in Engineering, Commutation and Computing*, 5:131–158, 1994.

[Gramlich, 1995] B. Gramlich. Abstract Relations Between Restricted Termination and Confluence Properties of Rewrite System. *Fund. Informaticae*, 24:3–23, 1995.

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

[Klop, 1992] J. W. Klop. Term Rewriting Systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–116. Oxford University Press, New York, 1992.

[Krishna Rao, 1994] M. R. K. Krishna Rao. Simple Termination of Hierarchical Combinations of Term Rewriting Systems. In *Proc. STACS '94*, pages 203–223. LNCS 789, 1994.

[Kurihara and Ohuchi, 1992] M. Kurihara and A. Ohuchi. Modularity of Simple Termination of Term Rewriting Systems with Shared Constructors. *Theoretical Computer Science*, 103:273–282, 1992.

[Lankford, 1979] D. S. Lankford. On Proving Term Rewriting Systems are Noetherian. Technical Report MTP-3, Louisiana Tech. Univ., Ruston, LA, 1979.

[Middeldorp and Zantema, 1997] A. Middeldorp and H. Zantema. Simple Termination of Rewrite Systems. *Theoretical Computer Science*, 175:127–158, 1997.

[Middeldorp, 1989] A. Middeldorp. A Sufficient Condition for the Termination of the Direct Sum of Term Rewriting Systems. In *Proc. LICS '89,* p. 396-401, 1989.

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

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

[Ohlebusch, 1994b] E. Ohlebusch. On the Modularity of Termination of Term Rewriting Systems. *Theoretical Computer Science*, 136:333–360, 1994.

[Ohlebusch, 1995] E. Ohlebusch. Modular Properties of Composable Term Rewriting Systems. *Journal of Symbolic Computation*, 20:1–41, 1995.

[Rusinowitch, 1987] M. Rusinowitch. On Termination of the Direct Sum of Term Rewriting Systems. *Information Processing Letters*, 26:65–70, 1987.

[Steinbach, 1995] J. Steinbach. Simplification Orderings: History of Results. *Fundamenta Informaticae*, 24:47–87, 1995.

[Toyama, 1987a] Y. Toyama. Counterexamples to Termination for the Direct Sum of Term Rewriting Systems. *Information Processing Letters*, 25:141–143, 1987.

[Toyama, 1987b] Y. Toyama. On the Church-Rosser Property for the Direct Sum of Term Rewriting Systems. *Journal of the ACM*, 34:128–143, 1987.

[Toyama *et al.*, 1995] Y. Toyama, J. W. Klop, and H. P. Barendregt. Termination for the Direct Sum of Left-Linear Term Rewriting Systems. *Journal of the ACM*, 42:1275–1304, 1995.