ELSEVIER

Theoretical Computer Science 236 (2000) 133–178

Theoretical Computer Science

www.elsevier.com/locate/tcs

Termination of term rewriting using dependency pairs

Thomas Arts[a], Jürgen Giesl[b,*]

[a] *Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, Netherlands*
[b] *Department of Computer Science, Darmstadt University of Technology, Alexanderstraße 10, 64283 Darmstadt, Germany*

Abstract

We present techniques to prove termination and innermost termination of term rewriting systems automatically. In contrast to previous approaches, we do not compare left- and right-hand sides of rewrite rules, but introduce the notion of *dependency pairs* to compare left-hand sides with special subterms of the right-hand sides. This results in a technique which allows to apply existing methods for automated termination proofs to term rewriting systems where they failed up to now. In particular, there are numerous term rewriting systems where a *direct* termination proof with simplification orderings is not possible, but in combination with our technique, well-known simplification orderings (such as the recursive path ordering, polynomial orderings, or the Knuth–Bendix ordering) can now be used to prove termination automatically. Unlike previous methods, our technique for proving *innermost* termination automatically can also be applied to prove innermost termination of term rewriting systems that are not terminating. Moreover, as innermost termination implies termination for certain classes of term rewriting systems, this technique can also be used for termination proofs of such systems.

Keywords: Termination; Term rewriting; Dependency pairs; Verification; Automated theorem proving

1. Introduction

Termination is one of the most fundamental properties of a term rewriting system (TRS), cf. e.g. [18]. While in general this problem is undecidable [29], several methods for proving termination have been developed (e.g. path orderings [15, 17, 31, 42, 45], Knuth–Bendix orderings [19, 33], forward closures [17, 38], semantic interpretations [11, 12, 23, 37, 43, 47], transformation orderings [9, 10, 44], distribution elimination [47], dummy elimination [21], semantic labelling [48], etc. – for surveys see e.g. [16, 45]).

* Corresponding author.
E-mail addresses: thomas@cs.ruu.nl (T. Arts), giesl@informatik.tu-darmstadt.de (J. Giesl)

PII: S0304-3975(99)00207-8

We present a new approach for the *automation* of termination proofs. Most well-known techniques for proving termination automatically try to find a well-founded ordering such that for all rules of the TRS the left-hand sides are greater than the corresponding right-hand sides. In most practical applications the synthesized orderings are total on ground terms [20] and therefore virtually all orderings used are *simplification orderings* [15, 16, 41, 45]. However, numerous TRSs are not simply terminating, i.e. not compatible with a simplification ordering. Hence, standard techniques like the recursive path ordering, polynomial interpretations, and the Knuth–Bendix ordering fail in proving termination of these TRSs.

In Section 2 we introduce a new criterion for termination based on the notion of *dependency pairs*. The main advantage of our termination criterion is that it is especially well suited for automation. To check the criterion automatically, we have developed a procedure which generates a set of constraints for every TRS. If there exists a well-founded ordering satisfying these constraints, then the TRS is terminating. For the synthesis of suitable orderings existing techniques, such as the recursive path ordering or polynomial interpretations, may be used. It turns out that for many TRSs where a *direct* application of simplification orderings fails, the constraints generated by our technique are nevertheless satisfied by an automatically generated simplification ordering. Moreover, all TRSs that can be proved terminating directly by synthesizing a simplification ordering automatically, can automatically be proved terminating by this new technique, too.

Rewriting under strategies is often used for modelling certain programming paradigms. For example, *innermost* rewriting, i.e. rewriting where only innermost redexes are contracted, can be used to model call-by-value computation semantics. For that reason, there has been an increasing interest in research on properties of rewriting under strategies. In particular, the study of termination is important when regarding such restricted versions of rewriting [27, 28, 36]. To prove *innermost termination* (also called (strong) innermost normalization), one has to show that the length of every innermost reduction is finite. Techniques for proving innermost termination can for example be utilized for termination proofs of functional programs (modelled by TRSs) with eager reduction strategy or of logic programs. (When transforming well-moded logic programs into TRSs, innermost termination of the TRS is sufficient for left-termination of the logic program [8].) Up to now, the only way to prove innermost termination *automatically* was by showing termination of the TRS. Therefore, none of the existing techniques could prove innermost termination of non-terminating systems. However, in Section 3 we show that after some modification, the dependency pair technique can be used as the first *specific* method for *innermost* termination. In Section 4 we conclude and give some comments on related work.

2. Proving termination

In this section we present a new approach for automated termination proofs. In Section 2.1, we state our termination criterion and prove that it is a necessary and

sufficient criterion for termination. Section 2.2 shows how this criterion can be checked automatically by generating a set of constraints that are satisfied by a well-founded ordering if and only if the criterion is fulfilled. The generation of suitable well-founded orderings is described in Section 2.3. To increase the power of our method we introduce a refined approach for its automation in Section 2.4 and an additional refinement in Section 2.5. In this way we obtain a very powerful technique which performs automated termination proofs for many TRSs where termination could not be proved automatically before. An overview of this technique is given in Section 2.6.

2.1. Termination criterion

For *constructor systems* it is common to split the signature into two disjoint sets, the *defined symbols* and the *constructors*. The following definition extends these notions to arbitrary term rewriting systems $\mathcal{R}(\mathcal{F}, R)$ (with the rules R over a signature $\mathcal{F}$). For an introduction to term rewriting and its notations, we refer to Dershowitz and Jouannaud [18] and Klop [32], for example. Here, the *root* of a term $f(\ldots)$ is the leading function symbol f.

Definition 1 (*Defined symbols and constructors*). Let $\mathcal{R}(\mathcal{F}, R)$ be a TRS. The set $D_{\mathcal{R}}$ of *defined symbols* of $\mathcal{R}$ is defined as $\{root(l) \mid l \to r \in R\}$ and the set $C_{\mathcal{R}}$ of *constructors* of $\mathcal{R}$ is defined as $\mathcal{F} \backslash D_{\mathcal{R}}$.

To refer to the defined symbols and constructors explicitly, a rewrite system is written as $\mathcal{R}(D_{\mathcal{R}}, C_{\mathcal{R}}, R)$ and the subscripts are omitted if $\mathcal{R}$ is clear from the context.

Example 2. The following TRS has two defined symbols, viz. minus and quot, and two constructors, viz. 0 and s:

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

Most techniques for automated termination proofs are restricted to simplification orderings. However, the TRS above is not compatible with a simplification ordering, because the left-hand side of the last quot-rule is embedded in its right-hand side if y is instantiated with $\mathsf{s}(x)$. Therefore these techniques cannot prove termination of this TRS.

In contrast to previous methods which compare left- and right-hand sides of rules, the central idea of our approach is to compare left-hand sides of rules only with those *subterms* of the right-hand sides that may possibly start a new reduction.

The motivation for this approach is to regard TRSs as ‘programs’. Intuitively, such a program is terminating if the arguments are decreasing in each recursive call. For

example, to prove termination of quot, instead of comparing both sides of the *rules*, one only has to compare the input *arguments* $\mathsf{s}(x), \mathsf{s}(y)$ with the arguments $\mathsf{minus}(x, y), \mathsf{s}(y)$ of the corresponding recursive call. This way of looking at termination of TRSs motivates that only those subterms of the right-hand sides that have a defined root symbol are considered for the examination of the termination behaviour.

More precisely, if a term $f(s_1, \ldots, s_n)$ rewrites to a term $C[g(t_1, \ldots, t_m)]$ (where g is a defined symbol and C denotes some context), then for proving termination, the argument tuples $s_1, \ldots, s_n$ and $t_1, \ldots, t_m$ are compared. In order to avoid the handling of tuples, the signature $\mathscr{F}$ of the TRS is extended by a set of fresh symbols, i.e., disjoint from the symbols in the signature, such that there is a one-to-one mapping from the defined symbols to these fresh symbols. The fresh symbols are called *tuple symbols*, and to ease readability, in this paper we assume that the original signature $\mathscr{F}$ consists of lower case function symbols only, whereas the tuple symbols are denoted by the corresponding upper case symbols. Instead of comparing *tuples*, now the *terms* $F(s_1, \ldots, s_n)$ and $G(t_1, \ldots, t_m)$ are compared, where F and G are the tuple symbols for f and g, respectively. [1]

Definition 3 (*Dependency pair*). Let $\mathscr{R}(D, C, R)$ be a TRS. If

$$f(s_1, \ldots, s_n) \to C[g(t_1, \ldots, t_m)]$$

is a rewrite rule of R with $g \in D$, then $\langle F(s_1, \ldots, s_n), G(t_1, \ldots, t_m)\rangle$ is called a *dependency pair* of $\mathscr{R}$.

The dependency pairs of a TRS are easily determined and if the TRS is finite, then only finitely many dependency pairs exist.

Example 4. The dependency pairs of the TRS in Example 2 are

$$\langle \mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{M}(x, y)\rangle \tag{1}$$

$$\langle \mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{M}(x, y)\rangle \tag{2}$$

$$\langle \mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{Q}(\mathsf{minus}(x, y), \mathsf{s}(y))\rangle \tag{3}$$

where M and Q denote the tuple symbols for minus and quot, respectively.

The notion of dependency pairs is the basis for our termination criterion. Since every left-hand side has a defined root symbol, no rule matches a term without defined symbols, hence such a term is a normal form. Thus, infinite reductions originate from the fact that defined symbols are introduced by the right-hand sides of rewrite rules. By tracing the introduction of these defined symbols, information is obtained about the

[1] Intuitively, tuple symbols are used to 'measure' the arguments of defined symbols during the termination proofs. Sometimes the arguments of two defined function symbols f and g must be measured in a different way. Thus, we introduce a different tuple symbol for each defined symbol to distinguish whether a tuple of terms serves as input for f or for g.

termination behaviour of the TRS. For that purpose we consider special sequences of dependency pairs, so-called *chains*, such that the right-hand side of every dependency pair in a chain corresponds to the newly introduced redex that should be traced.

Definition 5 (*Chain*). Let $\mathscr{R}(D,C,R)$ be a TRS. A sequence of dependency pairs $\langle s_1,t_1\rangle\langle s_2,t_2\rangle\ldots$ is an *$\mathscr{R}$-chain* if there exists a substitution σ such that $t_j\sigma \to^*_{\mathscr{R}} s_{j+1}\sigma$ holds for every two consecutive pairs $\langle s_j,t_j\rangle$ and $\langle s_{j+1},t_{j+1}\rangle$ in the sequence.

If $\mathscr{R}$ is clear from the context we often write 'chain' instead of '$\mathscr{R}$-chain'. We always assume that different (occurrences of) dependency pairs have disjoint sets of variables and we always regard substitutions whose domain may be infinite. Hence, in our example, we have the chain

$$\langle \mathsf{Q}(\mathsf{s}(x_1),\mathsf{s}(y_1)),\mathsf{Q}(\mathsf{minus}(x_1,y_1),\mathsf{s}(y_1))\rangle$$

$$\langle \mathsf{Q}(\mathsf{s}(x_2),\mathsf{s}(y_2)),\mathsf{Q}(\mathsf{minus}(x_2,y_2),\mathsf{s}(y_2))\rangle,$$

because $\mathsf{Q}(\mathsf{minus}(x_1,y_1),\mathsf{s}(y_1))\sigma \to^*_{\mathscr{R}} \mathsf{Q}(\mathsf{s}(x_2),\mathsf{s}(y_2))\sigma$ holds for the substitution σ that replaces x_1 by $\mathsf{s}(\mathsf{0})$, x_2 by $\mathsf{0}$, and both y_1 and y_2 by $\mathsf{0}$. In fact, any finite sequence of the dependency pair (3) in Example 4 is a chain. However, in the next section, we show that the above TRS has no *infinite* chain. The following theorem proves that the absence of infinite chains is a sufficient and necessary criterion for termination.

Theorem 6 (Termination criterion). *A TRS $\mathscr{R}(D,C,R)$ is terminating if and only if no infinite $\mathscr{R}$-chain exists.*

Proof. We first prove that the above criterion is *sufficient* for termination, i.e., we show that for any infinite reduction we can construct an infinite $\mathscr{R}$-chain.

Let t be a term that starts an infinite reduction. By a minimality argument, the term t contains a subterm[2] $f_1(\vec{u}_1)$ that starts an infinite reduction, but none of the terms $\vec{u}_1$ starts an infinite reduction, i.e., the terms $\vec{u}_1$ are strongly normalizing.

Let us consider an infinite reduction starting with $f_1(\vec{u}_1)$. First, the arguments $\vec{u}_1$ are reduced in zero or more steps to arguments $\vec{v}_1$ and then a rewrite rule $f_1(\vec{w}_1)\to r_1$ is applied to $f_1(\vec{v}_1)$, i.e., a substitution σ_1 exists such that $f_1(\vec{v}_1)=f_1(\vec{w}_1)\sigma_1 \to_{\mathscr{R}} r_1\sigma_1$. Now the infinite reduction continues with $r_1\sigma_1$, i.e., the term $r_1\sigma_1$ starts an infinite reduction, too.

By assumption there exists no infinite reduction beginning with one of the terms $\vec{v}_1=\vec{w}_1\sigma_1$. Hence, for all variables x occurring in $f_1(\vec{w}_1)$ the terms $\sigma_1(x)$ are strongly normalizing. Thus, since $r_1\sigma_1$ starts an infinite reduction, there occurs a subterm $f_2(\vec{u}_2)$ in r_1, i.e. $r_1=C[f_2(\vec{u}_2)]$ for some context C, such that

- $f_2(\vec{u}_2)\sigma_1$ starts an infinite reduction and
- $\vec{u}_2\sigma_1$ are strongly normalizing terms.

[2] Tuples of terms $t_1,\ldots,t_n$ are denoted by $\vec{t}$.

The first dependency pair of the infinite $\mathscr{R}$-chain that we construct is $\langle F_1(\vec{w}_1), F_2(\vec{u}_2)\rangle$ corresponding to the rewrite rule $f_1(\vec{w}_1) \to C[f_2(\vec{u}_2)]$. The other dependency pairs of the infinite $\mathscr{R}$-chain are determined in the same way: Let $\langle F_{j-1}(\vec{w}_{j-1}), F_j(\vec{u}_j)\rangle$ be a dependency pair such that $f_j(\vec{u}_j)\sigma_{j-1}$ starts an infinite reduction and the terms $\vec{u}_j\sigma_{j-1}$ are strongly normalizing. Again, in zero or more steps $f_j(\vec{u}_j)\sigma_{j-1}$ reduces to $f_j(\vec{v}_j)$ to which a rewrite rule $f_j(\vec{w}_j) \to r_j$ can be applied such that $r_j\sigma_j$ starts an infinite reduction for some substitution σ_j with $\vec{v}_j = \vec{w}_j\sigma_j$.

Similar to the observations above, since $r_j\sigma_j$ starts an infinite reduction, there must be a subterm $f_{j+1}(\vec{u}_{j+1})$ in r_j such that

- $f_{j+1}(\vec{u}_{j+1})\sigma_j$ starts an infinite reduction and
- $\vec{u}_{j+1}\sigma_j$ are strongly normalizing terms.

This results in the jth dependency pair of the chain, viz. $\langle F_j(\vec{w}_j), F_{j+1}(\vec{u}_{j+1})\rangle$. In this way, one obtains the infinite sequence

$$\langle F_1(\vec{w}_1), F_2(\vec{u}_2)\rangle\ \langle F_2(\vec{w}_2), F_3(\vec{u}_3)\rangle\ \langle F_3(\vec{w}_3), F_4(\vec{u}_4)\rangle \dots$$

It remains to prove that this sequence is really an $\mathscr{R}$-chain.

Note that $F_j(\vec{u}_j\sigma_{j-1}) \to^*_{\mathscr{R}} F_j(\vec{v}_j)$ and $\vec{v}_j = \vec{w}_j\sigma_j$. Since we assume, without loss of generality, that the variables of different occurrences of dependency pairs are disjoint, we obtain one substitution $\sigma = \sigma_1 \circ \sigma_2 \circ \cdots$ (which is the disjoint union of $\sigma_1, \sigma_2, \dots$) such that $F_j(\vec{u}_j)\sigma \to^*_{\mathscr{R}} F_j(\vec{w}_j)\sigma$ for all j. Thus, we have in fact constructed an infinite $\mathscr{R}$-chain.

Now we show that our criterion is even *necessary* for termination, i.e., we prove that any infinite $\mathscr{R}$-chain corresponds to an infinite reduction. Assume there exists an infinite $\mathscr{R}$-chain.

$$\langle F_1(\vec{s}_1), F_2(\vec{t}_2)\rangle\ \langle F_2(\vec{s}_2), F_3(\vec{t}_3)\rangle\ \langle F_3(\vec{s}_3), F_4(\vec{t}_4)\rangle \dots$$

Hence, there is a substitution σ such that

$$F_2(\vec{t}_2)\sigma \to^*_{\mathscr{R}} F_2(\vec{s}_2)\sigma,\ F_3(\vec{t}_3)\sigma \to^*_{\mathscr{R}} F_3(\vec{s}_3)\sigma, \dots$$

thus also

$$f_2(\vec{t}_2)\sigma \to^*_{\mathscr{R}} f_2(\vec{s}_2)\sigma,\ f_3(\vec{t}_3)\sigma \to^*_{\mathscr{R}} f_3(\vec{s}_3)\sigma, \dots$$

as the tuple symbols $F_2, F_3, \dots$ are no defined symbols.

Note that every dependency pair $\langle F(\vec{s}), G(\vec{t})\rangle$ corresponds to a rewrite rule $f(\vec{s}) \to C[g(\vec{t})]$ for some context C. Therefore, this results in the reduction

$$\begin{array}{lcl} f_1(\vec{s}_1)\sigma & \to & C_1[f_2(\vec{t}_2)]\sigma \\ & & \downarrow_* \\ & & C_1[f_2(\vec{s}_2)]\sigma \ \to\ C_1[C_2[f_3(\vec{t}_3)]]\sigma \\ & & \qquad\qquad \downarrow_* \\ & & \qquad\qquad C_1[C_2[f_3(\vec{s}_3)]]\sigma \ \to \cdots \end{array}$$

which is infinite. □

2.2. Checking the termination criterion automatically

The advantage of our termination criterion is that it is particularly well suited for automation. In this section we present a method for proving the absence of infinite chains automatically. For that purpose, we introduce a procedure which, given a TRS, generates a set of inequalities such that the existence of a well-founded ordering satisfying these inequalities is sufficient for termination of the TRS. A well-founded ordering satisfying the generated inequalities can often be synthesized by standard techniques, even if a *direct* termination proof is not possible with these techniques (i.e. even if a well-founded ordering orienting the rules of the TRS cannot be synthesized). For the automation of our method we assume the TRSs to be finite, such that only finitely many dependency pairs have to be considered.

Note that if all chains correspond to a decreasing sequence w.r.t. some well-founded ordering, then all chains must be finite. Hence, to prove the absence of infinite chains, we try to synthesize a well-founded ordering $>$ such that all dependency pairs are decreasing w.r.t. this ordering. More precisely, if for any sequence of dependency pairs $\langle s_1,t_1\rangle\langle s_2,t_2\rangle\langle s_3,t_3\rangle\ldots$ and for any substitution σ with $t_j\sigma \to_{\mathcal{R}}^* s_{j+1}\sigma$ we have

$$s_1\sigma > t_1\sigma > s_2\sigma > t_2\sigma > s_3\sigma > t_3\sigma > \cdots,$$

then no infinite chain exists.

However, for most TRSs, the above inequalities are not satisfied by any well-founded ordering $>$, because the terms $t_j\sigma$ and $s_{j+1}\sigma$ of consecutive dependency pairs in chains are often identical and therefore $t_j\sigma > s_{j+1}\sigma$ does not hold.

But obviously not *all* of the inequalities $s_j\sigma > t_j\sigma$ and $t_j\sigma > s_{j+1}\sigma$ have to be *strict*. For instance, to guarantee the absence of infinite chains it is sufficient if there exists a well-founded *quasi*-ordering[3] $\geqslant$ such that terms *in* dependency pairs are strictly decreasing (i.e. $s_j\sigma > t_j\sigma$) and terms *in between* dependency pairs are only weakly decreasing (i.e. $t_j\sigma \geqslant s_{j+1}\sigma$).

So for each sequence of dependency pairs as above we only demand

$$s_1\sigma > t_1\sigma \geqslant s_2\sigma > t_2\sigma \geqslant s_3\sigma > t_3\sigma \geqslant \cdots. \tag{4}$$

Note that we cannot determine automatically for which substitutions σ we have $t_j\sigma \to_{\mathcal{R}}^* s_{j+1}\sigma$ and moreover, it is practically impossible to examine infinite sequences of dependency pairs. Therefore, in the following, we restrict ourselves to *weakly monotonic* quasi-orderings $\geqslant$ where both $\geqslant$ and its strict part $>$ are *closed under substitution*. (A quasi-ordering $\geqslant$ is *weakly monotonic* if $s \geqslant t$ implies $f(\ldots s\ldots) \geqslant f(\ldots t\ldots)$.) Then, to guarantee $t_j\sigma \geqslant s_{j+1}\sigma$ whenever $t_j\sigma \to_{\mathcal{R}}^* s_{j+1}\sigma$ holds, it is sufficient to demand $l \geqslant r$ for all rewrite rules $l \to r$ of the TRS. To ensure $s_j\sigma > t_j\sigma$ for those dependency pairs occurring in possibly infinite chains, we demand $s > t$ for *all* dependency pairs

[3] A quasi-ordering $\geqslant$ is a reflexive and transitive relation and $\geqslant$ is called *well-founded* if its strict part $>$ is well founded.

$\langle s,t\rangle$. In fact, the existence of such a well-founded ordering is not only sufficient, but even necessary to ensure the absence of infinite chains.

Theorem 7 (Proving termination). *A TRS $\mathscr{R}(D,C,R)$ is terminating iff there exists a well-founded weakly monotonic quasi-ordering $\geqslant$, where both $\geqslant$ and $>$ are closed under substitution, such that*

- $l \geqslant r$ *for all rules* $l \to r$ *in* R *and*
- $s > t$ *for all dependency pairs* $\langle s,t\rangle$.

Proof. We first prove that the above conditions are sufficient, i.e. that the existence of such a quasi-ordering implies termination of $\mathscr{R}$. Note that as $l \geqslant r$ holds for all rules $l \to r$ in R and as $\geqslant$ is weakly monotonic and closed under substitution, we have $\to_{\mathscr{R}}^* \subseteq \geqslant$, i.e. if $t \to_{\mathscr{R}}^* s$ then $t \geqslant s$.

Suppose there is an infinite $\mathscr{R}$-chain $\langle s_1,t_1\rangle\langle s_2,t_2\rangle\ldots$ Then there exists a substitution σ such that $t_j\sigma \to_{\mathscr{R}}^* s_{j+1}\sigma$ holds for all j. As $\to_{\mathscr{R}}^* \subseteq \geqslant$, this implies $t_j\sigma \geqslant s_{j+1}\sigma$.

Since we have $s_j > t_j$ for all dependency pairs, we obtain the infinite descending sequence

$$s_1\sigma > t_1\sigma \geqslant s_2\sigma > t_2\sigma \geqslant s_3\sigma > \cdots,$$

which is a contradiction to the well-foundedness of $>$. Therefore, no infinite $\mathscr{R}$-chain exists and hence by Theorem 6, $\mathscr{R}$ is terminating.

Now we prove that the above conditions are even necessary for termination. In fact, we prove a stronger result, i.e. that termination of $\mathscr{R}$ implies termination of the system $\mathscr{R}'$ with the rules

$$R' = R \cup \{s \to t \mid \langle s,t\rangle \text{ is a dependency pair of } \mathscr{R}\}.$$

Hence, the rewrite ordering of $\mathscr{R}'$ is (even) a well-founded *strongly* monotonic ordering $>$ (closed under substitution) satisfying $s > t$ and the *strict* inequalities $l > r$ for all rules of R.[4]

Assume that $\mathscr{R}'$ is not terminating. Hence, there exists a term q_1 starting an infinite $\mathscr{R}'$-reduction.

$$q_1 \to_{\mathscr{R}'} q_2 \to_{\mathscr{R}'} \cdots \to_{\mathscr{R}'} q_k \to_{\mathscr{R}'} \cdots$$

Clearly, q_1 must contain tuple symbols, because $\mathscr{R}$ is terminating. Without loss of generality we may assume that q_1 is 'minimal', i.e. that none of the proper subterms of q_1 starts an infinite reduction. We show that this implies that the *root* of q_1 is a tuple symbol.

[4] The first intuition to prove this might be to imitate $\mathscr{R}'$-reductions by the union of $\to_{\mathscr{R}}$ and the subterm relation $>_{\text{sub}}$. However, this approach fails because $\mathscr{R}'$-reductions may also take place within contexts, whereas $\to_{\mathscr{R}} \cup >_{\text{sub}}$ is not monotonic if $\mathscr{R}$ is not simply terminating.

For any term q, let $[\![q]\!]$ denote the result of replacing all subterms with a root tuple symbol by one and the same new variable y. Note that tuple symbols do not occur in rewrite rules of $\mathcal{R}$. Therefore, $q_j \to_{\mathcal{R}'} q_{j+1}$ implies $[\![q_j]\!] \to_{\mathcal{R}} [\![q_{j+1}]\!]$, if the contracted redex in q_j is on a position *above* all tuple symbols. If the contracted redex is below a tuple symbol, then $q_j \to_{\mathcal{R}'} q_{j+1}$ implies $[\![q_j]\!] = [\![q_{j+1}]\!]$. If the contracted redex has a tuple root symbol, then $q_j \to_{\mathcal{R}'} q_{j+1}$ also implies $[\![q_j]\!] = [\![q_{j+1}]\!]$. The reason is that in this case the reduct also has a tuple root symbol, since all rewrite rules of $\mathcal{R}'$ that have a tuple symbol as root of the left-hand side also have a tuple symbol as root of the right-hand side. Hence, as $\mathcal{R}$ is terminating, after a finite number of steps (say k) all contracted redexes in the infinite reduction are below a tuple symbol or have a tuple root symbol (otherwise $[\![q_1]\!]$ would start an infinite $\mathcal{R}$-reduction).

Let q_k have the form $C_k[t_{k,1}, \ldots, t_{k,n_k}]$, where C_k is a context without tuple symbols and $t_{k,j}$ are terms with tuple root symbols. Then one of the $t_{k,j}$ starts an infinite reduction. Now assume that the root symbol of q_1 is not a tuple symbol, i.e., q_1 has the form $C_1[t_{1,1}, \ldots, t_{1,n_1}]$, where C_1 is a (non-empty) context without tuple symbols and $t_{1,1}, \ldots, t_{1,n_1}$ have tuple symbols on their root positions. By induction on the length k of the reduction, one shows that for each $t_{k,j}$ there exists a $t_{1,i}$ such that $t_{1,i} \to^*_{\mathcal{R}'} t_{k,j}$. Thus, q_1 has a proper subterm $t_{1,i}$ which starts an infinite reduction. This is a contradiction to the minimality of q_1.

Hence, q_1 has the form $F_1(\vec{u}_1)$ where $\vec{u}_1$ are strongly normalizing terms. So in the infinite reduction, first the arguments $\vec{u}_1$ are reduced in zero or more steps to $\vec{v}_1$, and then $F_1(\vec{v}_1)$ is reduced to $F_2(\vec{u}_2)$, i.e., $\langle F_1(\vec{v}_1), F_2(\vec{u}_2)\rangle$ is an instantiation of a dependency pair. Note that $\vec{u}_2$ are again strongly normalizing terms (this is due to the above observations, because all subterms of $\vec{u}_2$ with tuple root symbols already occur in $\vec{v}_1$). So the infinite reduction has the form

$$F_1(\vec{u}_1) \to^*_{\mathcal{R}'} F_1(\vec{v}_1) \to_{\mathcal{R}'} F_2(\vec{u}_2) \to^*_{\mathcal{R}'} F_2(\vec{v}_2) \to_{\mathcal{R}'} F_3(\vec{u}_3) \to^*_{\mathcal{R}'} \cdots,$$

where $\vec{u}_j \to^*_{\mathcal{R}'} \vec{v}_j$ holds for all j and $\langle F_j(\vec{v}_j), F_{j+1}(\vec{u}_{j+1})\rangle$ is an instantiation of a dependency pair of $\mathcal{R}$. Let

$$\langle F_1(\vec{s}_1), F_2(\vec{t}_2)\rangle \; \langle F_2(\vec{s}_2), F_3(\vec{t}_3)\rangle \ldots$$

be the sequence of these dependency pairs and let $\vec{s}_j\sigma = \vec{v}_j$ and $\vec{t}_j\sigma = \vec{u}_j$. If $\sigma'(x)$ is defined to be $[\![\sigma(x)]\!]$, then $F_j(\vec{t}_j)\sigma' \to^*_{\mathcal{R}} F_j(\vec{s}_j)\sigma'$ holds for all j. The reason is that in dependency pairs, tuple symbols occur on root positions only (i.e., $\vec{s}_j$ and $\vec{t}_j$ do not contain tuple symbols). Therefore, $\vec{s}_j\sigma' = [\![\vec{v}_j]\!]$, $\vec{t}_j\sigma' = [\![\vec{u}_j]\!]$ and again $\vec{u}_j \to^*_{\mathcal{R}'} \vec{v}_j$ implies $[\![\vec{u}_j]\!] \to^*_{\mathcal{R}} [\![\vec{v}_j]\!]$. So the above sequence of dependency pairs is an infinite $\mathcal{R}$-chain. By Theorem 6, this is a contradiction to the termination of $\mathcal{R}$. Hence, $\mathcal{R}'$ must also be terminating. □

By the above theorem, termination proofs are now reduced to the search for quasi-orderings satisfying certain constraints. Therefore, the technique of Theorem 7 is very

useful to apply standard methods like the recursive path ordering or polynomial interpretations to TRSs where they are not directly applicable.[5]

Example 8. For instance, in our example, we have to find a quasi-ordering satisfying the following inequalities.

$$\begin{aligned}
\mathsf{minus}(x,\mathsf{0}) &\geqslant x\\
\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) &\geqslant \mathsf{minus}(x,y)\\
\mathsf{quot}(\mathsf{0},\mathsf{s}(y)) &\geqslant \mathsf{0}\\
\mathsf{quot}(\mathsf{s}(x),\mathsf{s}(y)) &\geqslant \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x,y),\mathsf{s}(y)))\\
\mathsf{M}(\mathsf{s}(x),\mathsf{s}(y)) &> \mathsf{M}(x,y)\\
\mathsf{Q}(\mathsf{s}(x),\mathsf{s}(y)) &> \mathsf{M}(x,y)\\
\mathsf{Q}(\mathsf{s}(x),\mathsf{s}(y)) &> \mathsf{Q}(\mathsf{minus}(x,y),\mathsf{s}(y))
\end{aligned}$$

In the next section we show how quasi-orderings satisfying such sets of inequalities can be synthesized automatically using standard techniques.

2.3. Generating suitable quasi-orderings

A well-founded ordering satisfying the constraints in Example 8 can for instance be generated by the well-known technique of *polynomial interpretations* [37]. However, when using polynomial interpretations for *direct* termination proofs of TRSs, the polynomials have to be (strongly) monotonic in all their arguments, i.e. $s>t$ implies $f(\ldots s\ldots)>f(\ldots t\ldots)$. But for the approach of this paper, we only need a *weakly* monotonic quasi-ordering satisfying the inequalities. Thus, $s>t$ only implies $f(\ldots s\ldots)\geqslant f(\ldots t\ldots)$. Hence, when using our method it suffices to find a polynomial interpretation with weakly monotonic polynomials, which do not necessarily depend on all their arguments. For example, we may map $\mathsf{minus}(x,y)$ to the polynomial x which does not depend on the second argument y.

Then the inequalities in Example 8 are satisfied by a polynomial ordering where $\mathsf{0}$ is mapped to 0, $\mathsf{s}(x)$ is mapped to $x+1$, and $\mathsf{minus}(x,y)$, $\mathsf{quot}(x,y)$, $\mathsf{M}(x,y)$ and $\mathsf{Q}(x,y)$ are all mapped to x. Methods for the automated synthesis of polynomial orderings have for instance been developed in [23, 43]. In this way, termination of this TRS can

[5] Using the strict part $>$ of the quasi-ordering in Theorem 7 is sometimes too restrictive. In many standard methods based on semantic interpretations, quasi-orderings are lifted from ground terms to non-ground terms by defining $s\geqslant t$ iff $s\sigma\geqslant t\sigma$ for all ground substitutions σ. However, the strict part of such a quasi-ordering is in general not closed under substitution. On the other hand, the irreflexive ordering intuitively associated with such a quasi-ordering is defined as $s>_{\text{lift}} t$ iff $s\sigma>t\sigma$ holds for all ground substitutions σ, which is indeed closed under substitution. The ordering $>_{\text{lift}}$ is *compatible* with $\geqslant$, i.e., $>_{\text{lift}}\circ\geqslant\subseteq>$. From the proof of the theorem we easily see that instead of the strict part we in fact only need such a compatible ordering. Therefore in the following, '$>$' may also be read as this intuitively associated irreflexive ordering $>_{\text{lift}}$.

be proved fully automatically, although a direct termination proof with simplification orderings was not possible.

Instead of polynomial orderings one can also use path orderings, which can easily be generated automatically. However, these path orderings are always strongly monotonic, whereas in our method we only need a weakly monotonic ordering. For that reason, before synthesizing a suitable path ordering some of the arguments of function symbols may be eliminated. For instance, one may eliminate the second argument of the function symbol minus. Then every term $\mathsf{minus}(s,t)$ in the inequalities is replaced by $\mathsf{m}(s)$ (where m is a new unary function symbol). By comparing the terms resulting from this replacement (instead of the original terms) we can take advantage of the fact that minus does not have to be strongly monotonic in its second argument.

Example 9. In this way, the inequalities of Example 8 are transformed into

$$\begin{aligned}
\mathsf{m}(x) &\geqslant x\\
\mathsf{m}(\mathsf{s}(x)) &\geqslant \mathsf{m}(x)\\
\mathsf{quot}(\mathsf{0},\mathsf{s}(y)) &\geqslant \mathsf{0}\\
\mathsf{quot}(\mathsf{s}(x),\mathsf{s}(y)) &\geqslant \mathsf{s}(\mathsf{quot}(\mathsf{m}(x),\mathsf{s}(y)))\\
\\
\mathsf{M}(\mathsf{s}(x),\mathsf{s}(y)) &> \mathsf{M}(x,y)\\
\mathsf{Q}(\mathsf{s}(x),\mathsf{s}(y)) &> \mathsf{M}(x,y)\\
\mathsf{Q}(\mathsf{s}(x),\mathsf{s}(y)) &> \mathsf{Q}(\mathsf{m}(x),\mathsf{s}(y)).
\end{aligned}$$

These inequalities are satisfied by the recursive path ordering using the precedence $\mathsf{quot} \rhd \mathsf{s} \rhd \mathsf{m}$ and $\mathsf{Q} \rhd \mathsf{M}$.

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 minus one could replace all terms $\mathsf{minus}(s,t)$ by minus' first argument s. Then the resulting inequalities are again satisfied by the recursive path ordering. To perform this elimination of arguments resp. of function symbols we introduce the following concept.

Definition 10 (*Argument filtering TRS*). An *argument filtering TRS*[6] for the signature $\mathscr{F}$ (AFS for short) is a TRS whose rewrite rules are of the form

$$f(x_1,\ldots,x_n) \to g(y_1,\ldots y_k)$$

or

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

where $x_1,\ldots,x_n$ are pairwise different variables, $y_1,\ldots,y_k$ are pairwise different variables out of $x_1,\ldots,x_n$, $g \notin \mathscr{F}$, and for every function symbol $f \in \mathscr{F}$ there is at most one f-rule in the AFS.

[6] Argument filtering TRSs are a special form of recursive program schemes [13, 32]

From a rewriting point of view AFSs are quite simple, because every AFS is complete. Hence, for any term t the normal form $t\downarrow_{\mathscr{A}}$ w.r.t. an AFS $\mathscr{A}$ is unique.

The following theorem states that 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. Subsequently, one only has to find a quasi-ordering that satisfies these modified inequalities. Note that for a finite signature there are only finitely many AFSs (up to renaming of the symbols). 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.

Theorem 11 (Preservation under argument filtering). *Let $\mathscr{A}$ be an AFS and let IN be a set of inequalities. If the inequalities*

$$\{s\downarrow_{\mathscr{A}} > t\downarrow_{\mathscr{A}} \mid s > t \in IN\} \cup \{s\downarrow_{\mathscr{A}} \geqslant t\downarrow_{\mathscr{A}} \mid s \geqslant t \in IN\}$$

are satisfied by a well-founded weakly monotonic quasi-ordering (*where both $\geqslant$ and $>$ are closed under substitution*), *then there also exists such a quasi-ordering satisfying the inequalities IN.*

Proof. Assuming that the normalized inequalities are satisfied by a quasi-ordering $\geqslant$, a relation $\geqslant'$ on terms is defined where the terms are first normalized w.r.t. $\mathscr{A}$ and then compared w.r.t. the quasi-ordering $\geqslant$ (i.e. $s\geqslant' t$ iff $s\downarrow_{\mathscr{A}}\geqslant t\downarrow_{\mathscr{A}}$). It is straightforward to see that $\geqslant'$ is a well-founded quasi-ordering satisfying the inequalities IN.

For any substitution σ, let $\sigma\downarrow_{\mathscr{A}}$ denote the substitution which results from σ by normalizing all terms in the range of σ w.r.t. $\mathscr{A}$. Then, for all terms t and all substitutions σ we have $(t\sigma)\downarrow_{\mathscr{A}} = (t\downarrow_{\mathscr{A}})(\sigma\downarrow_{\mathscr{A}})$. Hence, both $\geqslant'$ and $>'$ are closed under substitution. Moreover, $\geqslant'$ is weakly monotonic, because $s\downarrow_{\mathscr{A}}\geqslant t\downarrow_{\mathscr{A}}$ implies $f(\ldots x\ldots)\downarrow_{\mathscr{A}}[x/s\downarrow_{\mathscr{A}}]\geqslant f(\ldots x\ldots)\downarrow_{\mathscr{A}}[x/t\downarrow_{\mathscr{A}}]$ resp. $f(\ldots s\ldots)\downarrow_{\mathscr{A}}\geqslant f(\ldots t\ldots)\downarrow_{\mathscr{A}}$ (here, x is a variable occurring just once in $f(\ldots x\ldots)$). □

By the above theorem in combination with Theorem 7 it is now possible to prove termination of the TRS in Example 2 automatically using the recursive path ordering. After normalizing all inequalities in Example 8 w.r.t. the one-rule AFS

$$\mathsf{minus}(x, y) \to \mathsf{m}(x),$$

one obtains the inequalities in Example 9 which are satisfied by the recursive path ordering.

2.4. Refinement using dependency graphs

While the method of Theorem 7 can be successfully used for automated termination proofs, in this section we introduce a refinement of this approach, i.e., we show how the constraints obtained can be weakened. By this weakening, the (automatic) search for a suitable quasi-ordering satisfying these constraints can be eased significantly.

In order to ensure that every possible infinite chain results in an infinite decreasing sequence of terms, in Theorem 7 we demanded $s > t$ for *all* dependency pairs $\langle s, t\rangle$. However, in many examples several dependency pairs can occur at most once in any chain and therefore they do not have to be considered at all. Moreover, for the other dependency pairs it is often sufficient if just *some* of them are strictly decreasing, whereas others may be weakly decreasing.

Example 12. The dependency pair $\langle \mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{M}(x, y)\rangle$ occurs at most *once* in any chain: Recall that a dependency pair $\langle v, w\rangle$ may only follow a pair $\langle s, t\rangle$ in a chain, if there exists a substitution σ such that $t\sigma \to_{\mathcal{R}}^{*} v\sigma$. As the tuple symbol M is not a defined symbol, $\mathsf{M}(x, y)\sigma$ can only be reduced to terms with the same root symbol M. Hence, the dependency pair (2) can only be succeeded by the dependency pair (1) which in turn can only be succeeded by itself, i.e. (2) can never occur twice in a chain. Therefore, any possible infinite chain has an infinite tail in which the dependency pair $\langle \mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{M}(x, y)\rangle$ does not occur. Therefore, it suffices to show that no infinite chain exists consisting of the other dependency pairs.

For the TRS of Example 2 it is not necessary to reduce the number of constraints in order to prove termination automatically. However, for the following TRS we have to get rid of a constraint in order to use a simplification ordering for satisfying the inequalities.

Example 13. Let us extend the TRS of Example 2 by three additional rules. We now write infix operators for the defined symbols minus and plus to ease readability:

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

The dependency pairs of this TRS are the dependency pairs as given in Example 4 together with the dependency pairs

$$\langle \mathsf{P}(\mathsf{s}(x), y), \mathsf{P}(x, y)\rangle \tag{5}$$

$$\langle \mathsf{M}(x - y, z), \mathsf{P}(y, z)\rangle \tag{6}$$

$$\langle \mathsf{M}(x - y, z), \mathsf{M}(x, y + z)\rangle \tag{7}$$

where P is the tuple symbol for the defined symbol '+'. To prove termination according to Theorem 7 we now obtain the following inequalities:

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

$$\begin{aligned}
\mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) &> \mathsf{M}(x, y)\\
\mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y)) &> \mathsf{M}(x, y)\\
\mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y)) &> \mathsf{Q}(x - y, \mathsf{s}(y))\\
\mathsf{P}(\mathsf{s}(x), y) &> \mathsf{P}(x, y)\\
\mathsf{M}(x - y, z) &> \mathsf{P}(y, z)\\
\mathsf{M}(x - y, z) &> \mathsf{M}(x, y + z)
\end{aligned}$$

Since the inequality $\mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y)) > \mathsf{Q}(x - y, \mathsf{s}(y))$ has an instantiation that is self-embedding, no simplification ordering satisfies these inequalities directly. In order to apply techniques for the automated generation of simplification orderings, therefore Theorem 11 has to be used first. We have to normalize the inequalities w.r.t. an AFS $\mathcal{A}$ that rewrites $x - y$ to $\mathsf{m}(x)$ or to x (this is forced by the inequalities). But thereafter, the inequality

$$\mathsf{M}(x - y, z)\downarrow_{\mathcal{A}} > \mathsf{P}(y, z)\downarrow_{\mathcal{A}}$$

in combination with the other remaining inequalities cannot be satisfied by any well-founded monotonic ordering closed under substitution. (The reason is that y does not occur in $\mathsf{M}(x - y, z)\downarrow_{\mathcal{A}}$ any more, whereas $\mathsf{P}(y, z)\downarrow_{\mathcal{A}}$ still depends on y, as $\mathcal{A}$ must not eliminate the first argument of P.) Hence, an automatic termination proof fails at this point.

Recall that one may delete all dependency pairs which occur at most once in any chain. In the example above, this elimination of constraints results in a set of inequalities for which a suitable quasi-ordering can be generated automatically, whereas this is not possible for the original set of constraints.

Example 14. For the TRS of Example 13, the constraint $\mathsf{M}(\ldots) > \mathsf{P}(\ldots)$ is unnecessary to ensure the absence of infinite chains. The reason is that in any chain the dependency pair (6) can occur at most once, since the only dependency pair following (6) can be (5) and (5) can only be followed by itself.

To determine those dependency pairs which may occur infinitely often in a chain we define a graph of dependency pairs where those dependency pairs that possibly occur consecutive in a chain are connected. In this way, any infinite chain corresponds to a cycle in the graph (as we restricted ourselves to finite TRSs).

Definition 15 (*Dependency graph*). The *dependency graph* of a TRS $\mathscr{R}$ is the directed graph whose nodes are the dependency pairs and there is an arc from $\langle s,t\rangle$ to $\langle v,w\rangle$ if $\langle s,t\rangle\langle v,w\rangle$ is an $\mathscr{R}$-chain.

Thus, the dependency graph connects dependency pairs that form a chain, i.e., for some instantiation the right-hand side of one pair reduces to the left-hand side of the other pair. Every chain corresponds to a path in the dependency graph. Note however that the converse does not hold, i.e., a path in this graph does not necessarily correspond to a chain, since instead of using one 'global' substitution for all dependency pairs in a chain, here one may use different 'local' substitutions for consecutive dependency pairs.

Example 16. As an example consider the TRS with the rules

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

It has the following dependency pairs.

$$\langle \mathsf{F}(x), \mathsf{G}(x,\mathsf{0})\rangle \tag{8}$$

$$\langle \mathsf{G}(\mathsf{1},y), \mathsf{F}(y)\rangle \tag{9}$$

Both (8) (9) and (9) (8) are chains, as can be seen using the 'local' substitutions $\sigma_1(x)=\mathsf{1},\sigma_1(y)=\mathsf{0}$ and $\sigma_2(x)=y$. Hence, in the dependency graph there are arcs from (8) to (9) and back. However, although (8) (9) (8) is on a path in the dependency graph, it is not a chain, because there exists no 'global' substitution σ such that $\mathsf{G}(x,\mathsf{0})\sigma \to^*_{\mathscr{R}} \mathsf{G}(\mathsf{1},y)\sigma$ *and* $\mathsf{F}(y)\sigma \to^*_{\mathscr{R}} \mathsf{F}(x)\sigma$.

Now to prove termination of a TRS it is sufficient if $s>t$ holds for at least one dependency pair $\langle s,t\rangle$ on each cycle of the dependency graph and if $s\geqslant t$ holds for all other dependency pairs on cycles. Dependency pairs that do not occur on a cycle can be ignored.

Example 17. For the TRS of Example 13 we obtain the dependency graph in Fig. 1. Hence, this results in the following set of inequalities:

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

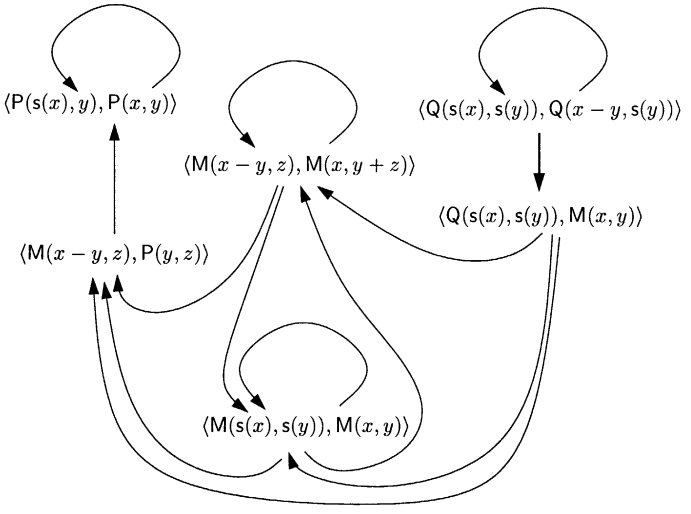

Fig. 1. The dependency graph for the TRS of Example 13.

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

$$\begin{aligned}
\mathsf{M}(\mathsf{s}(x), \mathsf{s}(y)) &> \mathsf{M}(x, y) \\
\mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y)) &> \mathsf{Q}(x - y, \mathsf{s}(y)) \\
\mathsf{P}(\mathsf{s}(x), y) &> \mathsf{P}(x, y) \\
\mathsf{M}(x - y, z) &> \mathsf{M}(x, y + z)
\end{aligned}$$

The inequalities obtained are satisfied by the polynomial ordering where $\mathsf{0}$ is mapped to 0, $\mathsf{s}(x)$ is mapped to $x + 2$, $x - y$ is mapped to $x + 1$, $\mathsf{quot}(x, y)$ is mapped to $2x$, $\mathsf{M}(x, y)$ and $\mathsf{Q}(x, y)$ are mapped to x, and both $+$ and P are mapped to addition. By normalizing the inequalities with respect to the argument filtering TRS

$$\begin{aligned}
x - y &\to \mathsf{m}(x) \\
\mathsf{M}(x, y) &\to x
\end{aligned}$$

the resulting inequalities are also satisfied by the recursive path ordering. Thus, by the following theorem, termination of the TRS is proved.

Theorem 18 (Dependency graph refinement). *A TRS $\mathscr{R}(D, C, R)$ is terminating iff there exists a well-founded weakly monotonic quasi-ordering $\geqslant$, where both $\geqslant$ and $>$ are closed under substitution, such that*

- *$l \geqslant r$ for all rules $l \to r$ in R,*

- $s \geqslant t$ *for all dependency pairs* $\langle s,t\rangle$ *on a cycle of the dependency graph, and*
- $s > t$ *for at least one dependency pair* $\langle s,t\rangle$ *on each cycle of the dependency graph.*

Proof. The proof is similar to the proof of Theorem 7 with the additional observation that any infinite $\mathscr{R}$-chain corresponds to an infinite path in the dependency graph. This infinite path traverses at least one cycle infinitely many times, since there are only finitely many dependency pairs. At least one dependency pair in this cycle corresponds to a strict inequality. Thus, the chain corresponds to a descending sequence of terms containing infinitely many strict inequalities.

Theorem 7 directly implies that the above conditions are also *necessary* for the termination of $\mathscr{R}$. □

However, to perform termination proofs according to Theorem 18, we have to construct the dependency graph automatically. Unfortunately, in general, this is not possible, since for two dependency pairs $\langle s,t\rangle, \langle v,w\rangle$ it is undecidable whether they form a chain (i.e. whether there exists a substitution σ such that $t\sigma \to^*_{\mathscr{R}} v\sigma$).

Therefore, we introduce a technique to approximate the dependency graph, i.e., the technique computes a *superset* of those terms t, v where $t\sigma \to^*_{\mathscr{R}} v\sigma$ holds for some substitution σ. We call terms t, v suggested by our technique *connectable terms*. In this way, (at least) all cycles that occur in the dependency graph and hence all possibly infinite chains can be determined. So by computing a graph *containing* the dependency graph we can indeed apply the method of Theorem 18 for automated termination proofs.

For the computation of connectable terms we use syntactic unification. This unification is not performed on the terms of the dependency pairs directly, but one of the terms is modified first. If t is a term with a constructor root symbol c, then $t\sigma$ can only be reduced to terms which have the same root symbol c. If the root symbol of t is defined, then this does not give us any direct information about those terms $t\sigma$ can be reduced to. For that reason, to determine whether the term t is connectable to v, we replace all subterms in t that have a defined root symbol by a new variable and check whether this modification of t unifies with v.

For example, $\mathsf{P}(\ldots)$ is not connectable to $\mathsf{M}(\ldots)$. On the other hand, the term $\mathsf{Q}(x - y, \mathsf{s}(y))$ is connectable to $\mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y))$, because before unification, the subterm $x - y$ is replaced by a new variable.

In order to ensure that t is connectable to v whenever there exists a substitution σ such that $t\sigma \to^*_{\mathscr{R}} v\sigma$, before unification we also have to *rename* multiple occurrences of the same variable x in t. (The reason is that different occurrences of $x\sigma$ can reduce to different terms.)

Example 19. As an example consider the following TRS of Toyama [46]:

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

The only dependency pair, viz. $\langle \mathsf{F}(\mathsf{0},\mathsf{1},x), \mathsf{F}(x,x,x)\rangle$, is on a cycle of the dependency graph, because $\mathsf{F}(x,x,x)\sigma$ reduces to $\mathsf{F}(\mathsf{0},\mathsf{1},x')\sigma$, if σ replaces x and x' by $\mathsf{g}(\mathsf{0},\mathsf{1})$. Note however that $\mathsf{F}(x,x,x)$ does not unify with $\mathsf{F}(\mathsf{0},\mathsf{1},x')$, i.e., if we would not rename $\mathsf{F}(x,x,x)$ to $\mathsf{F}(x_1,x_2,x_3)$ before the unification, then we could not determine this cycle of the dependency graph and we would falsely conclude termination of this (non-terminating) TRS.

To perform the required modification on the term t, two functions CAP and REN are introduced. For any term t, CAP(t) results from replacing all subterms of t that have a defined root symbol by different new variables and REN(t) results from replacing all variables in t by different fresh variables. In particular, different occurrences of the same variable are also replaced by different new variables.

Definition 20 (*Connectable terms*). Let D be the set of defined symbols. The functions CAP and REN from terms to terms are inductively defined as

$$\begin{aligned}
\mathrm{CAP}(x) &= x \quad \text{for variables } x,\\
\mathrm{CAP}(f(t_1,\ldots,t_n)) &= \begin{cases} y & \text{if } f \in D,\\ f(\mathrm{CAP}(t_1),\ldots,\mathrm{CAP}(t_n)) & \text{if } f \notin D,\end{cases}\\
\mathrm{REN}(x) &= y \quad \text{for variables } x,\\
\mathrm{REN}(f(t_1,\ldots,t_n)) &= f(\mathrm{REN}(t_1),\ldots,\mathrm{REN}(t_n)),
\end{aligned}$$

where y is the next variable in an infinite list of fresh variables $y_1, y_2, \ldots$.

For any terms t and v, the term t is *connectable* to v if REN(CAP(t)) and v are unifiable.

Strictly speaking, neither CAP nor REN are proper functions, because one time we have $\mathrm{REN}(x) = y_1$ and the next time we obtain $\mathrm{REN}(x) = y_2$. Of course, CAP and REN can easily be transformed into proper functions by giving CAP and REN a second argument which contains the next fresh variable that has not yet been used. However, we omitted this second argument to ease readability.

For example, we have

$$\mathrm{REN}(\mathrm{CAP}(\mathsf{Q}(x-y,\mathsf{s}(y)))) = \mathrm{REN}(\mathsf{Q}(y_1,\mathsf{s}(y))) = \mathsf{Q}(y_2,\mathsf{s}(y_3))$$

and

$$\mathrm{REN}(\mathrm{CAP}(\mathsf{Q}(x,x))) = \mathrm{REN}(\mathsf{Q}(x,x)) = \mathsf{Q}(y_4,y_5).$$

As REN(t) is always a linear term, to check whether two terms are connectable we can even use a unification algorithm without occur check.

To approximate the dependency graph, we draw an arc from a dependency pair $\langle s,t\rangle$ to $\langle v,w\rangle$ whenever t is connectable to v. In this way, for our example the dependency graph of Fig. 1 is constructed automatically. So termination of the TRS in Example 13 can be proved automatically.

The following theorem proves the soundness of this approach: by computing connectable terms we in fact obtain a supergraph of the dependency graph. Using this supergraph, we can now prove termination according to Theorem 18.

Theorem 21 (Computing dependency graphs). *Let $\mathcal{R}$ be a TRS and let $\langle s,t\rangle, \langle v,w\rangle$ be dependency pairs. If there is an arc from $\langle s,t\rangle$ to $\langle v,w\rangle$ in the dependency graph, then t is connectable to v.*

Proof. By induction on the structure of t we prove that if there exists a substitution σ with $t\sigma \to_{\mathcal{R}}^* u$ for some term u, then $\text{REN}(\text{CAP}(t))$ matches u. So in particular, if $t\sigma \to_{\mathcal{R}}^* v\sigma$, then $\text{REN}(\text{CAP}(t))$ matches $v\sigma$. As $\text{REN}(\text{CAP}(t))$ only contains new variables, this implies that $\text{REN}(\text{CAP}(t))$ and v are unifiable.

Assume that $t\sigma \to_{\mathcal{R}}^* u$ for some term u. If t is a variable or if $t = f(t_1,\ldots,t_k)$ for a defined symbol f, then $\text{REN}(\text{CAP}(t))$ is a variable, hence it matches u.

If $t = c(t_1,\ldots,t_k)$ for some constructor c, then

$$\text{REN}(\text{CAP}(t)) = c(\text{REN}(\text{CAP}(t_1)),\ldots,\text{REN}(\text{CAP}(t_k))).$$

In this case, u has to be of the form $c(u_1,\ldots,u_k)$ and $t_j\sigma \to_{\mathcal{R}}^* u_j$ holds for all j. By the induction hypothesis we obtain that $\text{REN}(\text{CAP}(t_j))$ matches u_j. Since the variables in $\text{REN}(\text{CAP}(t_j))$ are disjoint from the variables in $\text{REN}(\text{CAP}(t_i))$ for all $i \neq j$, $\text{REN}(\text{CAP}(t))$ also matches u. □

2.5. Refined termination proofs by narrowing dependency pairs

By the refinement of dependency graphs, Theorem 18 provides us with a powerful technique to prove that there exists no infinite chain of dependency pairs. However, there are still examples where the automation of our method fails.

Example 22. For instance, let us replace the last rule of the TRS in Example 13 by a 'commutativity' rule (here, $\mathsf{s0}$ abbreviates $\mathsf{s}(\mathsf{0})$, etc.):

$$\begin{aligned}
x - \mathsf{0} &\to x\\
\mathsf{s}(x) - \mathsf{s}(y) &\to x - y\\
\mathsf{quot}(\mathsf{0}, \mathsf{s}(y)) &\to \mathsf{0}\\
\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{quot}(x - y, \mathsf{s}(y)))\\
\mathsf{0} + y &\to y\\
\mathsf{s}(x) + y &\to \mathsf{s}(x + y)\\
(x - \mathsf{s0}) + (y - \mathsf{ss}z) &\to (y - \mathsf{ss}z) + (x - \mathsf{s0})
\end{aligned}$$

One of the new dependency pairs, viz.

$$\langle \mathsf{P}(x - \mathsf{s0}, y - \mathsf{ss}z), \mathsf{P}(y - \mathsf{ss}z, x - \mathsf{s0})\rangle, \tag{10}$$

forms a cycle of the dependency graph. Hence, due to Theorem 18 we have to find an ordering such that the dependency pair (10) is strictly decreasing, i.e.

$$\mathsf{P}(x - \mathsf{s0}, y - \mathsf{ss}z) > \mathsf{P}(y - \mathsf{ss}z, x - \mathsf{s0}).$$

In order to apply techniques for the synthesis of simplification orderings, we have to normalize the inequalities w.r.t. an AFS again which rewrites $x - y$ to $\mathsf{m}(x)$ (or to x). However, the resulting constraint

$$\mathsf{P}(\mathsf{m}(x), \mathsf{m}(y)) > \mathsf{P}(\mathsf{m}(y), \mathsf{m}(x))$$

is not satisfied by any well-founded ordering closed under substitution. Hence, in this way termination of the TRS cannot be proved automatically.

Up to now we demanded constraints which ensure that in any sequence of dependency pairs $\langle s_1, t_1\rangle \langle s_2, t_2\rangle \ldots$ and for all substitutions σ with $t_j\sigma \to_{\mathscr{R}}^* s_{j+1}\sigma$ we have [7]

$$s_1\sigma > t_1\sigma \geqslant s_2\sigma > t_2\sigma \geqslant \cdots . \tag{4}$$

So we demanded $s > t$ for the dependency pairs $\langle s, t\rangle$. But instead of the requirement that there should be a strict decrease *in* dependency pairs, it would also be sufficient if the ordering is strict *between* two dependency pairs. Thus, if $\langle s, t\rangle$ and $\langle v, w\rangle$ are consecutive in a chain, then instead of $s\sigma > t\sigma \geqslant v\sigma$ one could demand $s\sigma \geqslant t\sigma$ and $t\sigma > v\sigma$ for all substitutions σ with $t\sigma \to_{\mathscr{R}}^* v\sigma$.

To achieve this effect we replace the original dependency pairs by new pairs of terms. Subsequently, we demand that these *new* pairs of terms are *strictly* decreasing. Note that if the reduction from $t\sigma$ to $v\sigma$ is always of the form

$$t\sigma \to_{\mathscr{R}} t'\sigma \to_{\mathscr{R}}^* v\sigma,$$

then instead of $s\sigma > t\sigma \geqslant v\sigma$ we may also require $s\sigma > t'\sigma \geqslant v\sigma$. To compute the terms t' we use *narrowing* (cf. e.g. [30]).

Definition 23 (*Narrowing*). Let $\mathscr{R}$ be a TRS. A term t *narrows* to a term t' via the substitution μ (denoted by $t \rightsquigarrow_{\mathscr{R}} t'$), if there exists a non-variable position p in t, μ is the most general unifier of $t|_p$ and l for some rewrite rule $l \to r$ of $\mathscr{R}$, and $t' = t\mu[r\mu]_p$. (Here, the variables of $l \to r$ must have been renamed to fresh variables.)

If a dependency pair $\langle s, t\rangle$ is followed by another dependency pair $\langle v, w\rangle$ in a chain, and if t is not already unifiable with v (i.e. at least one rule of $\mathscr{R}$ is needed to reduce $t\sigma$ to $v\sigma$), then we may perform all possible narrowing steps on t (resulting in new terms $t_1, \ldots, t_n$) in order to examine the reduction from $t\sigma$ to $v\sigma$.

However, instead of only narrowing right-hand sides of dependency pairs $\langle s, t\rangle$, the substitutions derived from narrowing the term t should also be applied on the left-hand

[7] By taking the dependency graph into account, this requirement has been weakened, i.e., it is sufficient if just a certain subset of dependency pairs is strictly decreasing in any possibly infinite chain.

side s of the pair $\langle s,t\rangle$. Thus, if $t \rightsquigarrow_{\mathscr{R}} t_1, \ldots, t \rightsquigarrow_{\mathscr{R}} t_n$ are *all* possible narrowings of t (via the substitutions $\mu_1, \ldots, \mu_n$), then instead of

$$s\sigma > t\sigma \geqslant v\sigma \quad \text{for all } \sigma \text{ with } t\sigma \to^*_{\mathscr{R}} v\sigma$$

it is sufficient to demand

$$\begin{aligned} s\mu_1\sigma > t_1\sigma \geqslant v\sigma \quad &\text{for all } \sigma \text{ with } t_1\sigma \to^*_{\mathscr{R}} v\sigma, \\ &\vdots \\ s\mu_n\sigma > t_n\sigma \geqslant v\sigma \quad &\text{for all } \sigma \text{ with } t_n\sigma \to^*_{\mathscr{R}} v\sigma. \end{aligned}$$

Hence, we may replace the dependency pair $\langle s,t\rangle$ by the n new pairs $\langle s\mu_1, t_1\rangle, \ldots, \langle s\mu_n, t_n\rangle$. For that purpose instead of narrowing *terms* we introduce the concept of narrowing *pairs* of terms.

Definition 24 (*Narrowing pairs*). Let $\mathscr{R}$ be a TRS. If a term t narrows to a term t' via the substitution μ, then we say that the *pair* of terms $\langle s,t\rangle$ narrows to the pair $\langle s\mu, t'\rangle$.

Example 25. For Example 22, the instantiated right-hand side

$$\mathsf{P}(y - \mathsf{s}\mathsf{s}z, x - \mathsf{s}0)\sigma$$

of dependency pair (10) can only reduce to an instantiation of a left-hand side of a dependency pair if one of the minus-rules is applied to $(y - \mathsf{s}\mathsf{s}z)\sigma$ or $(x - \mathsf{s}0)\sigma$. So instead of the dependency pair (10) we may regard its two narrowings

$$\langle \mathsf{P}(x - \mathsf{s}0, \mathsf{s}y - \mathsf{s}\mathsf{s}z), \mathsf{P}(y - \mathsf{s}z, x - \mathsf{s}0)\rangle \tag{11}$$

$$\langle \mathsf{P}(\mathsf{s}x - \mathsf{s}0, y - \mathsf{s}\mathsf{s}z), \mathsf{P}(y - \mathsf{s}\mathsf{s}z, x - 0)\rangle. \tag{12}$$

Now the constraints that the left-hand sides of the new pairs (11) and (12) should be greater than their right-hand sides (together with the remaining constraints for this system) are again satisfied by the orderings mentioned in Example 17. Hence, in this way termination of the TRS can be proved automatically.

If $\mathscr{P}$ is the set of all dependency pairs of $\mathscr{R}$, then instead of checking whether there exists an infinite $\mathscr{R}$-chain of pairs from $\mathscr{P}$ now it suffices to show that there is no infinite $\mathscr{R}$-chain of pairs from $\mathscr{P}\backslash\{\langle s,t\rangle\} \cup \{\langle s\mu_1, t_1\rangle, \ldots, \langle s\mu_n, t_n\rangle\}$, where $\langle s\mu_1, t_1\rangle, \ldots, \langle s\mu_n, t_n\rangle$ are all narrowings of $\langle s,t\rangle$. (So with this refinement we have to regard chains of pairs of terms which are no dependency pairs any more.) Note that any pair $\langle s,t\rangle$ can only be narrowed (in one step) to *finitely many* pairs $\langle s',t'\rangle$ (up to variable renaming) and these pairs $\langle s',t'\rangle$ can easily be computed automatically. In particular, if a dependency pair $\langle s,t\rangle$ has *no* narrowings, then it does not have to be considered any more for the termination proof.

However, the following two examples demonstrate that a pair $\langle s,t\rangle$ in $\mathscr{P}$ may only be replaced by its narrowings, if t does not unify with any left-hand side of a pair in $\mathscr{P}$ and if t is a *linear* term.

Example 26. The following non-terminating TRS,

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

has one cycle in the dependency graph formed by an arc from the dependency pair $\langle \mathsf{F}(\mathsf{0}), \mathsf{F}(\mathsf{0})\rangle$ to itself. Narrowing this pair, although its right-hand side unifies with its left-hand side, results in $\langle \mathsf{F}(\mathsf{0}), \mathsf{F}(\mathsf{1})\rangle$. Now the new right-hand side $\mathsf{F}(\mathsf{1})$ is not connectable to $\mathsf{F}(\mathsf{0})$ any more. Hence, by ignoring the unification condition, the only cycle in the dependency graph would be erroneously removed and therefore termination of this TRS could be falsely concluded.

Similarly, the linearity of the right-hand side plays a crucial role, as can be seen from the non-terminating TRS

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

where $\langle \mathsf{F}(\mathsf{s}(x)), \mathsf{F}(\mathsf{g}(x,x))\rangle$ forms the only cycle of the dependency graph. However, by ignoring the linearity condition, this dependency pair could be deleted, as the term $\mathsf{F}(\mathsf{g}(x,x))$ cannot be narrowed. Hence, no cycle exists in the new dependency graph and therefore termination of the TRS would be falsely concluded.[8]

The following theorem proves that under the above conditions the replacement of dependency pairs by their narrowings maintains the sufficiency and necessity of our termination criterion.

Theorem 27 (Narrowing refinement for termination). *Let $\mathscr{R}$ be a TRS and let $\mathscr{P}$ be a set of pairs of terms. Let $\langle s,t\rangle \in \mathscr{P}$ such that t is linear and for all $\langle v,w\rangle \in \mathscr{P}$ the terms t and v are not unifiable* (*after renaming the variables*). *Let*

$$\mathscr{P}' = \mathscr{P}\backslash\{\langle s,t\rangle\} \cup \{\langle s',t'\rangle \mid \langle s',t'\rangle \text{ is a narrowing of } \langle s,t\rangle\}.$$

There exists an infinite $\mathscr{R}$-chain of pairs from $\mathscr{P}$ iff there exists an infinite $\mathscr{R}$-chain of pairs from $\mathscr{P}'$.

[8] The problem is that the first reduction step from $\mathsf{F}(\mathsf{g}(x,x))\sigma$ to $\mathsf{F}(\mathsf{s}(x'))\sigma$ takes place 'in σ' and therefore it cannot be captured by narrowing. For linear terms, this effect could be simulated by choosing another suitable σ', but in the above example this is not possible, because here two different occurrences of $x\sigma$ are reduced to different terms.

Proof. It suffices to prove that for every $\langle s,t\rangle \in \mathscr{P}$ the sequence

$$\ldots\langle v_1,w_1\rangle\langle s,t\rangle\langle v_2,w_2\rangle\ldots$$

(of pairs from $\mathscr{P}$ or $\mathscr{P}'$) is an $\mathscr{R}$-chain iff there exists a narrowing $\langle s',t'\rangle$ of $\langle s,t\rangle$ such that $\ldots\langle v_1,w_1\rangle\langle s',t'\rangle\langle v_2,w_2\rangle\ldots$ is an $\mathscr{R}$-chain. Here, $\langle s,t\rangle$ resp. $\langle s',t'\rangle$ may also be the first pair in the chain (i.e. $\langle v_1,w_1\rangle$ may be missing).

If this has been proved then all occurrences of $\langle s,t\rangle$ in an infinite chain may be replaced by pairs from $\mathscr{P}'$. In an analogous way, every infinite chain of pairs from $\mathscr{P}'$ can also be transformed into an infinite chain of pairs from $\mathscr{P}$.

For the first direction, let $\ldots\langle v_1,w_1\rangle\langle s,t\rangle\langle v_2,w_2\rangle\ldots$ be an $\mathscr{R}$-chain. Hence, there must be a substitution such that for all pairs, the instantiated right-hand side reduces to the instantiated left-hand side of the next pair in the chain. Let σ be such a substitution where the length of the reduction

$$t\sigma \to_{\mathscr{R}}^* v_2\sigma$$

is *minimal*. Note that the length of this reduction cannot be zero, as t and v_2 do not unify. Hence, we have $t\sigma \to_{\mathscr{R}} q \to_{\mathscr{R}}^* v_2\sigma$ for some term q.

There are two possibilities for the reduction $t\sigma \to_{\mathscr{R}} q$. Let us first assume that this reduction takes place 'in σ'. Hence, there is a variable x in t (i.e. $t|_p = x$ for some position p) such that $\sigma(x) \to_{\mathscr{R}} r$ and $q = t[r]_p$. The variable x only occurs *once* in t (as t is linear) and therefore, we have $q = t\sigma'$, where σ' is the substitution with $\sigma'(x) = r$ and $\sigma'(y) = \sigma(y)$ for all $y \neq x$. As all (occurrences of) dependency pairs are variable disjoint, σ' behaves like σ for all pairs except $\langle s,t\rangle$. For this pair, we have

$$w_1\sigma' = w_1\sigma \to_{\mathscr{R}}^* s\sigma \to_{\mathscr{R}}^* s\sigma'$$

and

$$t\sigma' = q \to_{\mathscr{R}}^* v_2\sigma = v_2\sigma'.$$

Hence, σ' is also a substitution where each instantiated right-hand side reduces to the instantiation of the left-hand side of the following pair in the chain. But as the reduction from $t\sigma'$ to $v_2\sigma'$ is shorter than the reduction from $t\sigma$ to $v_2\sigma$, this is a contradiction to the minimality of σ.

So the reduction $t\sigma \to_{\mathscr{R}} q$ cannot take place 'in σ'. Hence, t contains some subterm $f(\vec{u})$ such that a rule $l \to r$ has been applied to $f(\vec{u})\sigma$. In other words, l matches $f(\vec{u})\sigma$ (i.e. $l\rho = f(\vec{u})\sigma$). Hence, the reduction has the following form:

$$t\sigma = t\sigma[f(\vec{u})\sigma]_p = t\sigma[l\rho]_p \to_{\mathscr{R}} t\sigma[r\rho]_p = q.$$

Similar to Definition 23 we assume that the variables of $l \to r$ have been renamed to fresh ones. Therefore we can extend σ to 'behave' like ρ on the variables of l and r (but it still remains the same on the variables of all pairs in the chain). Now σ is a unifier of l and $f(\vec{u})$ and hence, there also exists a most general unifier μ. By the definition of most general unifiers, then there must be a substitution τ such that $\sigma = \mu\tau$.

Let t' be the term $t\mu[r\mu]_p$ and let s' be $s\mu$. Then $\langle s,t\rangle$ narrows to $\langle s',t'\rangle$. As we may assume s' and t' to be variable disjoint from all other pairs, we may extend σ to behave like τ on the variables of s' and t'. Then we have

$$w_1\sigma \to^*_{\mathscr{R}} s\sigma = s\mu\tau = s'\tau = s'\sigma$$

and

$$t'\sigma = t'\tau = t\mu\tau[r\mu\tau]_p = t\sigma[r\sigma]_p = t\sigma[r\rho]_p = q \to^*_{\mathscr{R}} v_2\sigma.$$

Hence, $\ldots\langle v_1,w_1\rangle\langle s',t'\rangle\langle v_2,w_2\rangle\ldots$ is also an $\mathscr{R}$-chain.

For the other direction of the theorem, let $\ldots\langle v_1,w_1\rangle\langle s',t'\rangle\langle v_2,w_2\rangle\ldots$ be an $\mathscr{R}$-chain. Hence, there is a substitution σ such that for all pairs the instantiated right-hand side reduces to the instantiated left-hand side of the next pair in the chain. So in particular we have

$$w_1\sigma \to^*_{\mathscr{R}} s'\sigma \quad \text{and} \quad t'\sigma \to^*_{\mathscr{R}} v_2\sigma.$$

We know that $\langle s,t\rangle$ narrows to $\langle s',t'\rangle$ via some substitution μ. As the variables in $\langle s,t\rangle$ are disjoint from all other occurring variables, we may extend σ to 'behave' like $\mu\sigma$ on the variables of s and t. Then we have $s\sigma = s\mu\sigma = s'\sigma$ and hence,

$$w_1\sigma \to^*_{\mathscr{R}} s\sigma.$$

Moreover, by the definition of narrowing, $t\mu \to_{\mathscr{R}} t'$. This implies $t\mu\sigma \to_{\mathscr{R}} t'\sigma$ and as $t\sigma = t\mu\sigma$, we have

$$t\sigma \to^*_{\mathscr{R}} v_2\sigma.$$

Hence, $\ldots\langle v_1,w_1\rangle\langle s,t\rangle\langle v_2,w_2\rangle\ldots$ is also an $\mathscr{R}$-chain. □

Note that while *dependency pairs* may indeed be replaced by their narrowings, in general a similar replacement of *rules* by their narrowings is unsound, i.e., it does not preserve the termination behaviour. For example, in the TRS with the rules $\mathsf{f}(\mathsf{1}) \to \mathsf{f}(\mathsf{0})$ and $\mathsf{0} \to \mathsf{1}$, the right-hand side $\mathsf{1}$ of the second rule cannot be narrowed. However, deleting this second rule transforms the non-terminating TRS into a terminating one. So narrowing of dependency pairs is different from narrowing of rules, because even if some dependency pairs are eliminated, still *all* rules can be used for the reductions *between* two dependency pairs.

Example 28. Narrowing pairs can be repeated several times if appropriate. So instead of replacing the dependency pair (10) by (11) and (12) we could also apply narrowing again and replace (11) and (12) by those pairs they narrow to. For example, the pair (11) has a linear right-hand side which does not unify with the left-hand side of any

pair. Thus it may be replaced by its narrowings

$$\langle \mathsf{P}(x - \mathsf{s0}, \mathsf{ss}y - \mathsf{ss}z), \mathsf{P}(y - z, x - \mathsf{s0})\rangle$$

$$\langle \mathsf{P}(\mathsf{s}x - \mathsf{s0}, \mathsf{s}y - \mathsf{ss}z), \mathsf{P}(y - \mathsf{s}z, x - 0)\rangle.$$

In general, before application of Theorem 18 one can apply an *arbitrary* number of narrowing steps to the dependency pairs. Subsequently, the resulting set of pairs is considered to be the 'set of dependency pairs' and the techniques presented to approximate the dependency graph and to synthesize the inequalities are applied. Finally, standard techniques are used to find an ordering satisfying the generated inequalities.

By the use of narrowing the automation of our method can be improved significantly. For instance, if in our example we perform at least one narrowing step, then termination can again be verified automatically.

Note that if an ordering can be found that satisfies the set of inequalities obtained without narrowing any of the pairs, then the inequalities obtained after narrowing are also satisfied by the same ordering. (If the ordering satisfies $s > t$ and $l \geqslant r$, then it also satisfies $s\mu > t\mu \geqslant t'$, provided that $t \rightsquigarrow_{\mathcal{R}} t'$ via the substitution μ. Hence, $s > t$ resp. $s \geqslant t$ implies $s' > t'$ resp. $s' \geqslant t'$ for any narrowing $\langle s', t'\rangle$ of $\langle s, t\rangle$. Moreover, if $\langle s', t'\rangle$ and $\langle v', w'\rangle$ are narrowings of $\langle s, t\rangle$ and $\langle v, w\rangle$, respectively, then there can only be an arc from $\langle s', t'\rangle$ to $\langle v', w'\rangle$ in the new dependency graph if there already was an arc from $\langle s, t\rangle$ to $\langle v, w\rangle$ in the original dependency graph. The corresponding statement also holds for our approximation of dependency graphs, i.e., if t' is connectable to v', then t is also connectable to v.) Thus, replacing pairs by their narrowings can only extend the set of TRSs for which termination can be proved automatically.

2.6. Summary

Combining all refinements, we obtain the following technique to prove termination automatically using the dependency pair approach:

- Determine the dependency pairs (this can be automated in a straightforward way).
- Replace some (dependency) pairs by all their narrowings. This step may be repeated several times.
- Approximate the dependency graph by estimating for all (dependency) pairs whether an arc exists between them. For this purpose, the functions CAP and REN are introduced. The pairs that occur on a cycle in the approximated dependency graph are computed by standard graph algorithms. Pairs which are not on a cycle in the approximated dependency graph can be ignored.
- Transform the rules and the (dependency) pairs on cycles into inequalities.
- Find a well-founded weakly monotonic quasi-ordering satisfying the inequalities after normalizing them with respect to one of the possible AFSs.

For finding suitable orderings standard techniques like the recursive path ordering or polynomial interpretations may be used. In this way, standard techniques can now be applied to prove termination of TRSs whose termination could not be proved

automatically before. For a collection of examples to demonstrate the power of our approach see [6].

3. Proving innermost termination

Similar to our approach for termination we now introduce a method to prove *innermost termination* of TRSs. Several ideas and notions can be transferred from the termination case to the innermost termination case. Therefore many theorems in this section look similar to the theorems in the previous section and in their proofs we only indicate the differences to the previous approach.

In Section 3.1 we present a criterion for innermost termination corresponding to the termination criterion of Section 2. We show in Section 3.2 that this criterion is also suitable for automation and that similar refinements for improving the technique can be developed (Section 3.3 and Section 3.4). The automated checking of this criterion enables us to prove innermost termination automatically, even if the TRS is not terminating. Additionally, for several classes of TRSs innermost termination already suffices for termination [27, 28]. Moreover, numerous modularity results exist for innermost termination [2, 7, 27, 35, 36], which do not hold for termination. Therefore, for those classes of TRSs *termination* can be proved by splitting the TRS and proving *innermost termination* of the subsystems separately. The advantage of this approach is that there are several interesting TRSs where a direct termination proof is not possible with the existing automatic techniques (including the technique of Section 2). However in many of these examples, a suitable ordering satisfying the constraints generated by our technique for proving *innermost* termination can nevertheless be synthesized automatically. So for many TRSs proving innermost termination automatically is essentially easier than proving termination. In this way, innermost termination (and thereby, termination) of many also non-simply terminating systems can now be verified automatically. An overview of the technique is given in Section 3.5.

3.1. Innermost termination criterion

In contrast to the approach in the previous section, now our aim is to prove that the length of every *innermost* reduction is finite (where innermost redexes have to be contracted first). Of course, termination implies innermost termination, but in general the converse does not hold.

Example 29. As an example consider the following TRS with the defined symbols f and g and the constructors $\mathsf{0}$ and s:

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

In this example, we have the following infinite (cycling) reduction:

$$\mathsf{f}(\mathsf{gs0},\mathsf{s0},\mathsf{gs0}) \to \mathsf{f}(\mathsf{gs0},\mathsf{gs0},\mathsf{gs0}) \to \mathsf{f}(\mathsf{gs0},\mathsf{sg0},\mathsf{gs0}) \to \mathsf{f}(\mathsf{gs0},\mathsf{s0},\mathsf{gs0}) \to \cdots$$

However, this reduction is not an innermost reduction, because in the first reduction step the subterm $\mathsf{gs0}$ is a redex and would have to be reduced first. Although this TRS is not terminating, it nevertheless turns out to be innermost terminating.

The aim of this section is to develop a criterion for innermost termination similar to our termination criterion of Section 2. In the above example we obtain the following dependency pairs:

$$\langle \mathsf{F}(\mathsf{g}(x),\mathsf{s}(\mathsf{0}),y), \mathsf{G}(x)\rangle \tag{13}$$

$$\langle \mathsf{F}(\mathsf{g}(x),\mathsf{s}(\mathsf{0}),y), \mathsf{F}(y,y,\mathsf{g}(x))\rangle \tag{14}$$

$$\langle \mathsf{G}(\mathsf{s}(x)), \mathsf{G}(x)\rangle \tag{15}$$

Recall that a sequence of dependency pairs $\langle s_1,t_1\rangle\langle s_2,t_2\rangle\ldots$ is a *chain*, if there exists a substitution σ such that $t_j\sigma$ reduces to $s_{j+1}\sigma$ for all j. Here, the right-hand side of each dependency pair corresponds to a newly introduced redex and the reductions $t_j\sigma \to^*_{\mathscr{R}} s_{j+1}\sigma$ are used to contract the arguments of the redex that is traced. However, *chains* correspond to *arbitrary* reductions, whereas now we are only interested in *innermost* reductions. Therefore, we have to restrict the definition of chains in order to obtain a notion which corresponds to the innermost reduction strategy.

The first restriction is motivated by the fact that when regarding innermost reductions, arguments of a redex should be in normal form before the redex is contracted. Therefore, we demand that all $s_j\sigma$ should be normal forms. Additionally, when concentrating on innermost reductions, the reductions of the arguments to normal form should also be innermost reductions. This results in the following restricted notion of a chain (where innermost reductions are denoted by '$\xrightarrow{\mathsf{i}}$').

Definition 30 (*Innermost chain*). Let $\mathscr{R}(D,C,R)$ be a TRS. A sequence of dependency pairs $\langle s_1,t_1\rangle\langle s_2,t_2\rangle\ldots$ is an *innermost $\mathscr{R}$-chain* if there exists a substitution σ such that all $s_j\sigma$ are normal forms and such that $t_j\sigma \xrightarrow{\mathsf{i}}{}^*_{\mathscr{R}} s_{j+1}\sigma$ holds for every two consecutive pairs $\langle s_j,t_j\rangle$ and $\langle s_{j+1},t_{j+1}\rangle$ in the sequence.

In our example we have the innermost chain

$$\langle \mathsf{G}(\mathsf{s}(x_1)), \mathsf{G}(x_1)\rangle\langle \mathsf{G}(\mathsf{s}(x_2)), \mathsf{G}(x_2)\rangle\langle \mathsf{G}(\mathsf{s}(x_3)), \mathsf{G}(x_3)\rangle$$

because $\mathsf{G}(x_1)\sigma \xrightarrow{\mathsf{i}}{}^*_{\mathscr{R}} \mathsf{G}(\mathsf{s}(x_2))\sigma$ and $\mathsf{G}(x_2)\sigma \xrightarrow{\mathsf{i}}{}^*_{\mathscr{R}} \mathsf{G}(\mathsf{s}(x_3))\sigma$ hold for the substitution σ that replaces x_1 by $\mathsf{s}(\mathsf{s}(x_3))$ and x_2 by $\mathsf{s}(x_3)$.

Of course, every innermost chain is also a chain, but not vice versa. For instance, the infinite sequence consisting of the second dependency pair (14) only is an infinite

chain, because

$$\mathsf{F}(y_1, y_1, \mathsf{g}(x_1))\sigma \to^*_{\mathscr{R}} \mathsf{F}(\mathsf{g}(x_2), \mathsf{s}(\mathsf{0}), y_2)\sigma \tag{16}$$

holds if $\sigma(x_j) = \mathsf{s}(\mathsf{0})$ and $\sigma(y_j) = \mathsf{g}(\mathsf{s}(\mathsf{0}))$. However, this infinite chain is not an innermost chain, because for every substitution σ satisfying (16), the term $\mathsf{F}(\mathsf{g}(x_2), \mathsf{s}(\mathsf{0}), y_2)\sigma$ is not a normal form. The following theorem proves that the absence of infinite innermost chains is a sufficient and necessary criterion for innermost termination. (Hence, the restriction of *chains* to *innermost chains* in fact corresponds to the restriction of *reductions* to *innermost reductions*.)

Theorem 31 (Innermost termination criterion). *A TRS $\mathscr{R}(D, C, R)$ is innermost terminating if and only if no infinite innermost $\mathscr{R}$-chain exists.*

Proof. The proof of this theorem is very similar to the proof of Theorem 6. In the same way as in the latter proof, an infinite sequence of dependency pairs can be constructed, whenever an infinite innermost reduction exists. The difference, however, is that now the arguments of the terms are innermost reduced to normal form before building the next dependency pair, whereas in the proof of Theorem 6 the arguments were reduced an arbitrary number of steps. The sequence constructed in this way is in fact an innermost chain.

For the other direction, similar to the corresponding proof of Theorem 6 one can show that any infinite innermost chain corresponds to an infinite innermost reduction. □

3.2. Checking the innermost termination criterion automatically

In this section we present an automatic approach for innermost termination proofs using the criterion of Theorem 31, i.e., we develop a method to prove the absence of infinite innermost chains automatically.

Assume that there is a sequence $\langle s_1, t_1\rangle\langle s_2, t_2\rangle\langle s_3, t_3\rangle\ldots$ of dependency pairs and a substitution σ such that all terms $s_j\sigma$ are in normal form and such that $t_j\sigma$ reduces innermost to $s_{j+1}\sigma$ for all j. Then to prove that this sequence is finite, it suffices again to find a well-founded quasi-ordering $\gtrsim$ such that the following inequalities are satisfied:

$$s_1\sigma > t_1\sigma \gtrsim s_2\sigma > t_2\sigma \gtrsim s_3\sigma > t_3\sigma \gtrsim \cdots . \tag{4}$$

To ensure that all dependency pairs are decreasing, we again demand $s > t$ for all dependency pairs $\langle s, t\rangle$. In our example this results in the following constraints, cf. (13)–(15):

$$\mathsf{F}(\mathsf{g}(x), \mathsf{s}(\mathsf{0}), y) > \mathsf{G}(x) \tag{17}$$

$$\mathsf{F}(\mathsf{g}(x), \mathsf{s}(\mathsf{0}), y) > \mathsf{F}(y, y, \mathsf{g}(x)) \tag{18}$$

$$\mathsf{G}(\mathsf{s}(x)) > \mathsf{G}(x). \tag{19}$$

Moreover, we have to ensure $t_j\sigma \geqslant s_{j+1}\sigma$ whenever $t_j\sigma \xrightarrow{i}{}^*_{\mathscr{R}} s_{j+1}\sigma$ holds. Recall that to prove termination we demanded that *all* rules were weakly decreasing. This was necessary, because in chains, σ may be an arbitrary substitution and hence, every rule can be used in the reduction from $t_j\sigma$ to $s_{j+1}\sigma$.[9] However, in contrast to the situation for chains, in an innermost chain only a subset of the rewrite rules of the TRS can be applied in the reduction in between the dependency pairs. Therefore, to prove innermost termination we only demand the constraints $l \geqslant r$ for those rules $l \to r$ that can be used in an innermost reduction of $t_j\sigma$. Note that as all terms $s_j\sigma$ are normal, σ is a *normal substitution* (i.e., it instantiates all variables with normal forms). Hence, for the dependency pairs (13) and (15) we directly obtain that *no* rule can ever be used to reduce a normal instantiation of $\mathsf{G}(x)$ (because G is no defined symbol). The only dependency pair whose right-hand side can be reduced if its variables are instantiated with normal forms is (14), because this is a dependency pair with a *defined symbol* on the right-hand side. As the only defined symbol in $\mathsf{F}(y, y, \mathsf{g}(x))$ is g, the only rules that may be applied on normal instantiations of this term are the two g-rules of the TRS. Since these g-rules can never introduce a new redex with root symbol f, the two g-rules are the only rules that can be used to reduce any normal instantiation of $\mathsf{F}(y, y, \mathsf{g}(x))$. Hence, in this example we only have to demand that these rules should be weakly decreasing:

$$\mathsf{g}(\mathsf{s}(x)) \geqslant \mathsf{s}(\mathsf{g}(x)), \quad \mathsf{g}(\mathsf{0}) \geqslant \mathsf{0}. \tag{20}$$

In general, to determine the *usable rules*, i.e. (a superset of) those rules that may possibly be used in an innermost reduction of a normal instantiation of a term t, we proceed as follows. If t contains a defined symbol f, then all f-rules are *usable* and furthermore, all rules that are *usable* for right-hand sides of f-rules are also *usable* for t. However, if one of these rules contains a redex as a proper subterm of the left-hand side, then we do not have to include it in the usable rules, since this rule can never be applied in any innermost reduction. (Of course, such rules could also be directly removed from the TRS before the innermost termination proof.)

Definition 32 (*Usable rules*). Let $\mathscr{R}(D, C, R)$ be a TRS. For any symbol f let $Rules(R, f) = \{l \to r \text{ in } R \mid root(l) = f,\ l \text{ has no redex as proper subterm}\}$. For any term t, the set of usable rules $\mathbf{U}(R, t)$ is inductively defined as

$$\mathbf{U}(R, x) = \emptyset$$

$$\mathbf{U}(R, f(t_1, \ldots, t_n)) = Rules(R, f) \cup \bigcup_{j=1}^{n} \mathbf{U}(R', t_j) \cup \bigcup_{l \to r \in Rules(R,f)} \mathbf{U}(R', r),$$

where[10] $R' = R \backslash Rules(R, f)$.

[9] Provided that a variable occurs in t_j, but termination is decidable for TRSs with ground right-hand sides [14].

[10] $\mathbf{U}(R, t)$ is well-defined, because its first argument R is decreasing.

Hence, in our example we have

$$\begin{aligned}\mathbf{U}(R,\mathsf{F}(y,y,\mathsf{g}(x))) &= \mathit{Rules}(R,\mathsf{F}) \cup \mathbf{U}(R,y) \cup \mathbf{U}(R,\mathsf{g}(x)) \cup \emptyset \\ &= \mathbf{U}(R,\mathsf{g}(x)) \\ &= \mathit{Rules}(R,\mathsf{g}) \cup \\ &\quad \mathbf{U}(\{\mathsf{f}(\ldots) \to \mathsf{f}(\ldots)\},x) \cup \\ &\quad \mathbf{U}(\{\mathsf{f}(\ldots) \to \mathsf{f}(\ldots)\},\mathsf{s}(\mathsf{g}(x))) \cup \\ &\quad \mathbf{U}(\{\mathsf{f}(\ldots) \to \mathsf{f}(\ldots)\},\mathsf{0}) \\ &= \{\mathsf{g}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{g}(x)), \mathsf{g}(\mathsf{0}) \to \mathsf{0}\}.\end{aligned}$$

Observe that by the above definition $\mathit{Rules}(R,f) = \emptyset$ for any constructor f.

When proving termination we had to search for a *weakly monotonic* quasi-ordering satisfying the constraints obtained. The reason for demanding weak monotonicity was that $l \geqslant r$ for all rules had to ensure $t_j\sigma \geqslant s_{j+1}\sigma$ whenever $t_j\sigma$ could be reduced to $s_{j+1}\sigma$. However, now for the tuple symbols we do not need weak monotonicity on all positions any more. For example, for the tuple symbol F we only have to ensure that all reductions starting from $\mathsf{F}(y,y,\mathsf{g}(x))\sigma$ are weakly decreasing (where σ is a normal substitution). Obviously, such reductions can never take place in the first two arguments of F and hence, F does not have to be weakly monotonic in these arguments.

The constraints (20) already ensure that during reductions of $\mathsf{F}(y,y,\mathsf{g}(x))\sigma$ the value of the *subterm* $\mathsf{g}(x)\sigma$ can only be decreased. Of course, we have to guarantee that the value of the *whole* term $\mathsf{F}(y,y,\mathsf{g}(x))$ is weakly decreasing if an instantiation of $\mathsf{g}(x)$ is replaced by a smaller term. For that purpose, we demand that $\mathsf{F}(y,y,\mathsf{g}(x))$ must be weakly monotonic on the position of its subterm $\mathsf{g}(x)$, i.e., for the tuple symbol F we only have to demand the following monotonicity constraint:

$$x_1 \geqslant x_2 \Rightarrow \mathsf{F}(y,y,x_1) \geqslant \mathsf{F}(y,y,x_2). \tag{21}$$

We only compute such monotonicity constraints for the tuple symbols and for all other (lower-case) symbols we demand weak monotonicity in all of their arguments. In general, we obtain the following procedure for the generation of constraints.

Theorem 33 (Proving innermost termination). *Let $\mathcal{R}(D,C,R)$ be a TRS and let $\geqslant$ be a well-founded quasi-ordering where both $\geqslant$ and $>$ are closed under substitution. If $\geqslant$ is weakly monotonic on all symbols apart from the tuple symbols and if $\geqslant$ satisfies the following constraints for all dependency pairs $\langle s,t\rangle$*

- *$l \geqslant r$ for all usable rules $l \to r$ in $\mathbf{U}(R,t)$,*
- *$s > t$,*
- *$x_1 \geqslant y_1 \wedge \cdots \wedge x_n \geqslant y_n \Rightarrow C[x_1,\ldots,x_n] \geqslant C[y_1,\ldots,y_n]$, if $t = C[f_1(\vec{u}_1),\ldots,f_n(\vec{u}_n)]$,*

where C is a context without defined symbols and $f_1,\ldots,f_n$ are defined symbols, then $\mathcal{R}$ is innermost terminating.

Proof. The proof of this theorem corresponds to the proof of Theorem 7. Suppose that $\langle s_1, t_1\rangle\langle s_2, t_2\rangle \ldots$ is an infinite innermost $\mathscr{R}$-chain. Then there exists a substitution σ such that $s_j\sigma$ is a normal form and $t_j\sigma$ reduces innermost to $s_{j+1}\sigma$ for all j. Hence, σ replaces all variables by normal forms and therefore, the only rules that can be applied in this reduction are the usable rules $\mathbf{U}(R, t_j)$. All usable rules are weakly decreasing and the terms t_j are weakly monotonic on all positions where reductions are applied. (The reason is that lower-case symbols are weakly monotonic and without loss of generality we can assume that σ does not introduce any tuple symbols, i.e., the only tuple symbol in $t_j\sigma$ is on the root position.) Hence, we have $t_j\sigma \geqslant s_{j+1}\sigma$. This results in an infinite decreasing sequence $s_1\sigma > t_1\sigma \geqslant s_2\sigma > t_2\sigma \geqslant \cdots$ which is a contradiction to the well-foundedness of $\geqslant$. Thus, no infinite innermost $\mathscr{R}$-chain exists and by Theorem 31, the TRS is innermost terminating. □

So there are two main differences between the termination approach and the approach for innermost termination. The first difference is in the set of inequalities that is generated. As we restrict ourselves to innermost reductions and to terms $s_j\sigma$ that are normal forms, several inequalities that have to be demanded when proving termination are unnecessary when proving innermost termination (i.e., we do not have to demand $l \geqslant r$ for all rules any more, but it suffices if just the usable rules are weakly decreasing). After generating the inequalities, the second difference is that the quasi-ordering satisfying the inequalities does not have to be weakly monotonic for all function symbols (i.e., tuple symbols only have to satisfy the monotonicity constraints that are stated explicitly).

Hence, in Example 29, to prove innermost termination it is sufficient to find a well-founded quasi-ordering satisfying the constraints in (17)–(21). For the synthesis of suitable quasi-orderings we proceed in the same way as it has been done for termination (Section 2.3) where for polynomial interpretations the difference is that the polynomials do not have to be weakly monotonic in all arguments.

For example, these constraints are fulfilled by the polynomial ordering where the constant $\mathsf{0}$ is mapped to the number 0, $\mathsf{s}(x)$ is mapped to $x + 1$, $\mathsf{g}(x)$ is mapped to $x + 2$, $\mathsf{F}(x, y, z)$ is mapped to $(x - y)^2 + 1$, and $\mathsf{G}(x)$ is mapped to x. Note that this quasi-ordering is not weakly monotonic on the tuple symbol F. The only monotonicity constraint in our example is (21), which is obviously satisfied as $\mathsf{F}(x, y, z)$ is mapped to a polynomial which is weakly monotonic [11] in its third argument z. However, this polynomial is not weakly monotonic in x or y.

Unlike Theorem 7 for termination proofs, the existence of a quasi-ordering satisfying the conditions of Theorem 33 is sufficient, but not necessary for innermost termination.

[11] When using polynomial interpretations, monotonicity constraints like (21) can also be represented as inequalities. For instance, if F is mapped to some polynomial $[\mathsf{F}]$, then instead of (21) one could demand that the *partial derivative* of $[\mathsf{F}](y, y, x)$ with respect to x should be non-negative, i.e. $\partial[\mathsf{F}](y, y, x)/\partial x \geqslant 0$, cf. [23]. If one uses other techniques (e.g. path orderings) which can only generate monotonic orderings, then of course one may drop monotonicity constraints like (21).

The reason is that demanding the constraints of Theorem 33 for *all* instantiations may be too strong, since for *innermost* chains sometimes it would be sufficient to regard *certain* instantiations only.

Example 34. For example, consider the innermost terminating TRS

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

$$\mathsf{g}(\mathsf{h}(x)) \to \mathsf{g}(x)$$

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

$$\mathsf{g}(\mathsf{0}) \to \mathsf{s}(\mathsf{0})$$

$$\mathsf{h}(\mathsf{0}) \to \mathsf{a}.$$

In this example there are no infinite innermost chains. However, the constraints according to Theorem 33 include the inequalities

$$\mathsf{F}(\mathsf{s}(x)) > \mathsf{F}(\mathsf{g}(\mathsf{h}(x)))$$

$$\mathsf{g}(\mathsf{h}(x)) \geqslant \mathsf{g}(x)$$

$$\mathsf{g}(\mathsf{0}) \geqslant \mathsf{s}(\mathsf{0})$$

$$x_1 \geqslant x_2 \Rightarrow \mathsf{F}(x_1) \geqslant \mathsf{F}(x_2).$$

These constraints imply $\mathsf{F}(\mathsf{s}(\mathsf{0})) > \mathsf{F}(\mathsf{g}(\mathsf{h}(\mathsf{0}))) \geqslant \mathsf{F}(\mathsf{g}(\mathsf{0})) \geqslant \mathsf{F}(\mathsf{s}(\mathsf{0}))$. Therefore they cannot be satisfied by any well-founded quasi-ordering closed under substitution.

However, the approach of Theorem 33 suffices to prove *innermost termination* of numerous important examples and challenge problems (including the TRS in Example 29) automatically, i.e., this technique allows the application of standard techniques for innermost termination proofs, even if the TRS is not terminating. Moreover, using the results of Gramlich [27, 28], Theorem 33 can also be applied to prove *termination* of TRSs that are non-overlapping (or for locally confluent overlay systems).

Example 35. As an example regard the following TRS by Kolbe [34] where $\mathsf{quot}(x, y, z)$ is used to compute $1 + \lfloor (x - y)/z \rfloor$, if $x \geqslant y$ and $z \neq 0$ (i.e. $\mathsf{quot}(x, y, y)$ computes $\lfloor x/y \rfloor$):

$$\mathsf{quot}(\mathsf{0}, \mathsf{s}(y), \mathsf{s}(z)) \to \mathsf{0}$$

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

$$\mathsf{quot}(x, \mathsf{0}, \mathsf{s}(z)) \to \mathsf{s}(\mathsf{quot}(x, \mathsf{s}(z), \mathsf{s}(z)))$$

The above system is not simply terminating (the left-hand side of the last rule is embedded in the right-hand side if z is instantiated with $\mathsf{0}$) and therefore most automatic approaches for termination proofs (which are restricted to simplification orderings) fail.

Nevertheless, with our technique we can prove innermost termination and therefore termination of this system automatically. As quot is the only defined symbol of this system, we obtain the following dependency pairs:

$$\langle \mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y), z), \mathsf{Q}(x, y, z) \rangle \tag{22}$$

$$\langle \mathsf{Q}(x, \mathsf{0}, \mathsf{s}(z)), \mathsf{Q}(x, \mathsf{s}(z), \mathsf{s}(z)) \rangle. \tag{23}$$

In this example there are no usable rules, as on the right-hand sides of the dependency pairs no defined symbols occur. Hence, due to Theorem 33 we only have to find a well-founded ordering such that both dependency pairs are decreasing. These constraints are for instance satisfied by the polynomial ordering where $\mathsf{0}$ is mapped to the number 0, $\mathsf{s}(x)$ is mapped to $x + 1$, and $\mathsf{Q}(x, y, z)$ is mapped to $x + (x - y + z)^2$. Hence, innermost termination and thereby also termination of this TRS is proved (as it is non-overlapping).

Note that again we benefit from the fact that the tuple symbol Q need not be weakly monotonic in its arguments. Therefore, an interpretation like the polynomial $x + (x - y + z)^2$ may be used, which is not weakly monotonic in any of its arguments. In fact, if the set of usable rules is empty, then the quasi-ordering need not even be weakly monotonic for any symbol. The termination approach of Section 2 cannot be used to prove termination of this TRS automatically, since the generated inequalities are not satisfied by any well-founded weakly monotonic total quasi-ordering or any quasi-simplification ordering (not even after normalization by a suitable AFS).

3.3. *Refinement using innermost dependency graphs*

To prove innermost termination of a TRS according to Theorem 33 we have to find an ordering such that $s > t$ holds for *all* dependency pairs $\langle s, t \rangle$. However, similar to the refinement for termination proofs in Section 2.4, for certain rewrite systems this requirement can be weakened, i.e., it is sufficient to demand $s > t$ for *some* dependency pairs only.

For instance, in the quot example (Example 35) up to now we demanded that both dependency pairs (22) and (23) had to be decreasing. However, two occurrences of the dependency pair (23) can never follow each other in an innermost chain, because $\mathsf{Q}(x_1, \mathsf{s}(z_1), \mathsf{s}(z_1))\sigma$ can never reduce to any instantiation of $\mathsf{Q}(x_2, \mathsf{0}, \mathsf{s}(z_2))$. The reason is that the second arguments $\mathsf{s}(z_1)$ resp. $\mathsf{0}$ of these two terms have different constructor root symbols. Hence, any possible infinite innermost chain would contain infinitely many occurrences of the other dependency pair (22). Therefore, it is sufficient if (22) is decreasing and if (23) is just weakly decreasing. In this way, we obtain the following (weakened) constraints:

$$\mathsf{Q}(\mathsf{s}(x), \mathsf{s}(y), z) > \mathsf{Q}(x, y, z) \tag{24}$$

$$\mathsf{Q}(x, \mathsf{0}, \mathsf{s}(z)) \geqslant \mathsf{Q}(x, \mathsf{s}(z), \mathsf{s}(z)). \tag{25}$$

In general, to determine those dependency pairs which may possibly follow each other in innermost chains, we define the following graph.

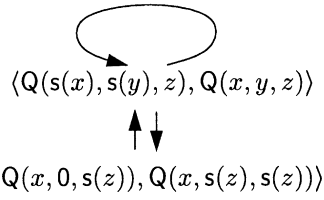

Fig. 2. The innermost dependency graph for the quot TRS (Example 35).

Definition 36 (*Innermost dependency graph*). The *innermost dependency graph* of a TRS $\mathscr{R}$ is the directed graph whose nodes are the dependency pairs and there is an arc from $\langle s,t\rangle$ to $\langle v,w\rangle$ if $\langle s,t\rangle\langle v,w\rangle$ is an innermost $\mathscr{R}$-chain.

For instance, in the innermost dependency graph for the quot example (Fig. 2) there are arcs from (22) to itself and to (23), and there is an arc from (23) to (22) (but *not* to itself).

Of course, the fact that innermost chains are restricted chains cause innermost dependency graphs to be subgraphs of dependency graphs. Now any infinite innermost chain corresponds to a cycle in the innermost dependency graph. Hence, it is sufficient if $s > t$ holds for at least one dependency pair $\langle s,t\rangle$ on every cycle and if $s \geqslant t$ holds for the other dependency pairs on cycles. So, similar to Theorem 18 (for termination) we obtain the following refined theorem for automated innermost termination proofs.

Theorem 37 (Innermost dependency graph refinement). *Let $\mathscr{R}(D,C,R)$ be a TRS and let $\geqslant$ be a well-founded quasi-ordering where both $\geqslant$ and $>$ are closed under substitution. If $\geqslant$ is weakly monotonic on all symbols apart from the tuple symbols and if $\geqslant$ satisfies the following constraints for all dependency pairs $\langle s,t\rangle$ on a cycle of the innermost dependency graph*

- *$l \geqslant r$ for all usable rules $l \to r$ in $\mathbf{U}(R,t)$,*
- *$s \geqslant t$,*
- *$x_1 \geqslant y_1 \wedge \cdots \wedge x_n \geqslant y_n \Rightarrow C[x_1,\ldots,x_n] \geqslant C[y_1,\ldots,y_n]$, if $t = C[f_1(\vec{u}_1),\ldots,f_n(\vec{u}_n)]$,*

where C is a context without defined symbols and $f_1,\ldots,f_n$ are defined symbols, and if $s > t$ holds for at least one dependency pair $\langle s,t\rangle$ on each cycle of the innermost dependency graph, then $\mathscr{R}$ is innermost terminating.

Proof. The proof that Theorem 37 is a consequence of Theorem 33 is completely analogous to the proof that Theorem 18 is a consequence of Theorem 7. □

Hence, in the quot example the constraints (24) and (25) are in fact sufficient for innermost termination. A suitable quasi-ordering satisfying these weakened constraints can easily be synthesized using the technique of Section 2.3. (For instance, one could use the polynomial interpretation where 0 and s are interpreted as usual and where $\mathsf{Q}(x,y,z)$ is mapped to x. If the constraints (24) and (25) are normalized w.r.t. an AFS

which drops the second argument of Q, then they are also satisfied by the recursive path ordering.) This example demonstrates that the weakening of the constraints by using innermost dependency graphs often enables the application of much simpler orderings (e.g., now we can use the recursive path ordering or a linear weakly monotonic polynomial ordering whereas for the original constraints of Section 3.2 we needed a non-monotonic polynomial of degree 2).

However, for an automation of Theorem 37 we have to construct the innermost dependency graph. Again, this cannot be done automatically, since for two pairs $\langle s,t\rangle$ and $\langle v,w\rangle$ it is undecidable whether there exists a substitution σ such that $t\sigma$ reduces innermost to $v\sigma$ and such that $s\sigma$ and $v\sigma$ are normal forms. Hence, similar to the dependency graph, we can only approximate this graph by computing a supergraph containing the innermost dependency graph. Note that $t\sigma$ may only reduce to $v\sigma$ for some substitution σ, if either t has a defined root symbol or if both t and v have the same constructor root symbol.

Recall that $\mathrm{CAP}(t)$ denotes the result of replacing all subterms in t with a defined root symbol by different fresh variables. Then $t\sigma$ can only reduce to $v\sigma$ if $\mathrm{CAP}(t)$ and v are unifiable.

However, this replacement of subterms of t must only be done for terms which are not equal to subterms of s. The reason is that such subterms are already in normal form when instantiated with σ. For example, if we modify the first rule of the TRS in Example 29 to $\mathsf{f}(\mathsf{g}(x),\mathsf{s}(\mathsf{0}))\to\mathsf{f}(\mathsf{g}(x),\mathsf{g}(x))$, then to determine whether there is an arc from the resulting dependency pair

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

to itself, the subterms $\mathsf{g}(x)$ in the right-hand side do not have to be replaced by new variables. As both sides of this dependency pair do not unify after variable renaming, one can immediately see that this pair is not on a cycle of the innermost dependency graph (whereas $\mathrm{CAP}(\mathsf{F}(\mathsf{g}(x),\mathsf{g}(x)))=\mathsf{F}(x_1,x_2)$ would unify with the left-hand side).

Let $\mathrm{CAP}_s(t)$ only replace those subterms of t by different fresh variables which have a defined root symbol and which are not equal to subterms of s. Then to refine the approximation of innermost dependency graphs instead of $\mathrm{CAP}(t)$ we check whether $\mathrm{CAP}_s(t)$ unifies with v. Moreover, if μ is the most general unifier (mgu) of $\mathrm{CAP}_s(t)$ and v, then there can only be an arc from $\langle s,t\rangle$ to $\langle v,w\rangle$ in the innermost dependency graph, if both $s\mu$ and $v\mu$ are in normal form.

So there are three differences between the approximation of the dependency graph and the approximation of the innermost dependency graph. First, for the innermost dependency graph we only replace subterms of t which do not occur in s, i.e., we use $\mathrm{CAP}_s(t)$ instead of $\mathrm{CAP}(t)$. Second, to approximate the dependency graph, multiple occurrences of the same variable in $\mathrm{CAP}(t)$ are replaced by fresh variables (using the function REN), whereas the variables in $\mathrm{CAP}_s(t)$ are left unchanged for the approximation of the innermost dependency graph. The reason is that any substitution used for instantiating the dependency pairs of an innermost chain is a normal substitution. Thus, variables are always instantiated by normal forms, and hence these instantiations are

not reduced. Multiple occurrences of the same variable in a term result in multiple occurrences of the same subterm after reduction of the instantiated term. In contrast, for an arbitrary substitution, instantiated multiple occurrences of the same variable may result in different subterms after reduction of the instantiated term.

The third difference is that for innermost dependency graphs we only draw an arc from $\langle s,t\rangle$ to $\langle v,w\rangle$, if the mgu of $\mathrm{CAP}_s(t)$ and v instantiates s and v to normal forms. This condition is due to the restriction to *innermost* chains.

Similar to the notion of *connectable terms* in Section 2.4, we call two dependency pairs *innermost connectable* if they should be connected by an arc in our approximation of the innermost dependency graph.

Definition 38 (*Innermost connectable pairs*). For any dependency pairs $\langle s,t\rangle$ and $\langle v,w\rangle$, the pair $\langle s,t\rangle$ is *innermost connectable* to $\langle v,w\rangle$ if $\mathrm{CAP}_s(t)$ and v are unifiable by some mgu μ such that $s\mu$ and $v\mu$ are in normal form.

The following theorem proves the soundness of our approximation.

Theorem 39 (Computing innermost dependency graphs). *Let $\mathcal{R}$ be a TRS and let $\langle s,t\rangle$ and $\langle v,w\rangle$ be dependency pairs. If there is an arc from $\langle s,t\rangle$ to $\langle v,w\rangle$ in the innermost dependency graph, then $\langle s,t\rangle$ is innermost connectable to $\langle v,w\rangle$.*

Proof. Due to the additional conditions in the definition of innermost chains and the definition of *innermost* connectable pairs, the proof is slightly different from the proof of Theorem 21.

By induction on the structure of t we show that if there exists a substitution σ such that $s\sigma$ is a normal form and $t\sigma \to_{\mathcal{R}}^* u$ for some term u, then there exists a substitution τ (whose domain only includes variables that are introduced in the construction of $\mathrm{CAP}_s(t)$) with $\mathrm{CAP}_s(t)\sigma\tau = u$. Thus, in particular, if there exists a substitution σ such that $s\sigma$ and $v\sigma$ are normal forms and $t\sigma \to_{\mathcal{R}}^* v\sigma$, then $\mathrm{CAP}_s(t)\sigma\tau = v\sigma$ $(= v\sigma\tau$, since the variables of $v\sigma$ do not occur in the domain of $\tau)$. Hence, $\mathrm{CAP}_s(t)$ and v unify and the most general unifier μ is such that $s\mu$ and $v\mu$ are normal forms. (There exist instantiations of these two terms that are normal forms (viz. $s\sigma\tau = s\sigma$ and $v\sigma\tau = v\sigma$), hence the terms $s\mu$ and $v\mu$ are normal forms themselves.)

If t equals a subterm of s, then $t\sigma$ is in normal form, hence $t\sigma$ equals u. Moreover, we have $\mathrm{CAP}_s(t) = t$. So $\mathrm{CAP}_s(t)\sigma = u$, i.e., in this case τ is the empty substitution.

If t is not equal to a subterm of s and $root(t)$ is defined, then $\mathrm{CAP}_s(t)$ is a fresh variable. Let τ replace $\mathrm{CAP}_s(t)$ by u. Then we have $\mathrm{CAP}_s(t)\sigma\tau = \mathrm{CAP}_s(t)\tau = u$.

Otherwise, $t = c(t_1,\ldots,t_n)$ for some constructor c and we have

$$\mathrm{CAP}_s(t) = c(\mathrm{CAP}_s(t_1),\ldots,\mathrm{CAP}_s(t_n)).$$

In this case u is of the form $c(u_1,\ldots,u_n)$ and $t_j\sigma \to_{\mathcal{R}}^* u_j$ for all j. By the induction hypothesis there exist substitutions τ_j such that $\mathrm{CAP}_s(t_j)\sigma\tau_j = u_j$. Note that the variables newly introduced in $\mathrm{CAP}_s(t_j)$ are disjoint from those variables newly introduced in

$\text{CAP}_s(t_i)$ for $i \neq j$. Hence, if $\tau = \tau_1 \circ \cdots \circ \tau_n$, then for all j we have $\text{CAP}_s(t_j)\sigma\tau = u_j$, and thus, $\text{CAP}_s(t)\sigma\tau = c(u_1, \ldots, u_n)$. □

Using the approximation of Theorem 39, we can now compute the innermost dependency graph for the quot example in Fig. 2 automatically.

Example 40. There are also examples where the innermost dependency graph does not contain any cycles:

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

In this example, the dependency pair $\langle \mathsf{F}(x, \mathsf{g}(x)), \mathsf{F}(\mathsf{1}, \mathsf{g}(x)) \rangle$ is not on a cycle of the innermost dependency graph, although $\text{CAP}_{\mathsf{F}(x_1, \mathsf{g}(x_1))}(\mathsf{F}(\mathsf{1}, \mathsf{g}(x_1))) = \mathsf{F}(\mathsf{1}, \mathsf{g}(x_1))$ unifies with $\mathsf{F}(x_2, \mathsf{g}(x_2))$ using a mgu that replaces x_1 and x_2 by 1. However, the instantiated left-hand side $\mathsf{F}(\mathsf{1}, \mathsf{g}(\mathsf{1}))$ is not a normal form, since it contains the redex $\mathsf{g}(\mathsf{1})$. The other dependency pairs $\langle \mathsf{F}(x, \mathsf{g}(x)), \mathsf{G}(x) \rangle$ and $\langle \mathsf{G}(\mathsf{1}), \mathsf{G}(\mathsf{0}) \rangle$ cannot occur on cycles either, since $\mathsf{G}(\ldots)$ does not unify with $\mathsf{F}(\ldots)$ and $\mathsf{G}(\mathsf{0})$ does not unify with $\mathsf{G}(\mathsf{1})$. Hence, using the refined techniques of Theorems 39 and 37 we obtain *no* constraint at all, i.e., innermost termination can be proved by only computing the (approximation of) the innermost dependency graph.

3.4. Refined innermost termination proofs by narrowing dependency pairs

Similar to the termination technique of Section 2, the power of our technique can be increased if we consider narrowings of the dependency pairs.

Example 41. For an illustration regard the following TRS:

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

Here, a 'conditional' program for minus has been encoded into an unconditional TRS. The dependency pairs on cycles of the innermost dependency graph are

$$\langle \mathsf{LE}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{LE}(x, y) \rangle \tag{26}$$
$$\langle \mathsf{M}(x, y), \mathsf{IF}(\mathsf{le}(x, y), x, y) \rangle \tag{27}$$
$$\langle \mathsf{IF}(\mathsf{false}, x, y), \mathsf{M}(\mathsf{p}(x), y) \rangle. \tag{28}$$

However, the constraints resulting from application of Theorem 37 would imply $\mathsf{M}(\mathsf{s}(x),\mathsf{0}) > \mathsf{M}(\mathsf{p}(\mathsf{s}(x)),\mathsf{0})$. Therefore, an automatic innermost termination proof using quasi-simplification orderings fails.

The only dependency pair whose right-hand side does not unify with any left-hand side of a dependency pair is (27). Hence, in any innermost chain at least one rule of the TRS must be applied in order to reduce an instantiation of $\mathsf{IF}(\mathsf{le}(x,y),x,y)$ to an instantiation of a left-hand side. So instead of examining the dependency pair (27) we may first perform all possible narrowing steps and replace (27) by

$$\langle \mathsf{M}(\mathsf{0},y), \mathsf{IF}(\mathsf{true},\mathsf{0},y)\rangle \tag{29}$$

$$\langle \mathsf{M}(\mathsf{s}(x),\mathsf{0}), \mathsf{IF}(\mathsf{false},\mathsf{s}(x),\mathsf{0})\rangle \tag{30}$$

$$\langle \mathsf{M}(\mathsf{s}(x),\mathsf{s}(y)), \mathsf{IF}(\mathsf{le}(x,y),\mathsf{s}(x),\mathsf{s}(y))\rangle. \tag{31}$$

Note that while the right-hand side of (28) unifies with the left-hand side of the original dependency pair (27), after this replacement the right-hand side of (28) does not unify with left-hand sides any more. Hence, the first narrowing of (27) now enables a subsequent narrowing of (28). So (28) is replaced by

$$\langle \mathsf{IF}(\mathsf{false},\mathsf{0},y), \mathsf{M}(\mathsf{0},y)\rangle \tag{32}$$

$$\langle \mathsf{IF}(\mathsf{false},\mathsf{s}(x),y), \mathsf{M}(x,y)\rangle. \tag{33}$$

In this way, the original set of dependency pairs (26)–(28) is transformed into (26) and (29)–(33). The pairs (29) and (32) are not on cycles of the innermost dependency graph and can therefore be ignored in the innermost termination proof. Thus, our method determines that instead of the original dependency pair (27) one only has to regard instantiations where x is instantiated with a term of the form $\mathsf{s}(\ldots)$. But for those terms, p is decreasing and hence, the call of M on the right-hand side of (33) is applied to smaller arguments than the call of M on the left-hand side of (30) or (31).

Now innermost termination (and thereby termination) of the system can be proved by the technique of Theorem 37. This results in the following constraints.

$$\begin{aligned}
\mathsf{le}(\mathsf{0},y) &\geqslant \mathsf{true}\\
\mathsf{le}(\mathsf{s}(x),\mathsf{0}) &\geqslant \mathsf{false}\\
\mathsf{le}(\mathsf{s}(x),\mathsf{s}(y)) &\geqslant \mathsf{le}(x,y)\\
\\
\mathsf{LE}(\mathsf{s}(x),\mathsf{s}(y)) &> \mathsf{LE}(x,y)\\
\mathsf{M}(\mathsf{s}(x),\mathsf{0}) &\geqslant \mathsf{IF}(\mathsf{false},\mathsf{s}(x),\mathsf{0})\\
\mathsf{M}(\mathsf{s}(x),\mathsf{s}(y)) &\geqslant \mathsf{IF}(\mathsf{le}(x,y),\mathsf{s}(x),\mathsf{s}(y))\\
\mathsf{IF}(\mathsf{false},\mathsf{s}(x),y) &> \mathsf{M}(x,y)\\
x_1 \geqslant x_2 \Rightarrow \mathsf{IF}(x_1,\mathsf{s}(x),\mathsf{s}(y)) &\geqslant \mathsf{IF}(x_2,\mathsf{s}(x),\mathsf{s}(y))
\end{aligned}$$

These constraints are satisfied by a polynomial interpretation where $\mathsf{0}$, true and false are mapped to 0, $\mathsf{s}(x)$ is mapped to $x+1$, $\mathsf{le}(x,y)$, $\mathsf{LE}(x,y)$, and $\mathsf{M}(x,y)$ are mapped to

x, and $\mathsf{IF}(x,y,z)$ is mapped to y. They are also satisfied by the recursive path ordering if an AFS is used to eliminate the first argument of IF.

Narrowing pairs for the innermost termination technique has the side-effect that one may also drop some inequalities $l \geqslant r$ corresponding to the rules $l \to r$, since after narrowing the pairs some rules may not be usable any more. For example, for the original dependency pairs, the p-rules were usable, since (28) contains an occurrence of p on its right-hand side. But after narrowing this dependency pair, the occurrence of p is deleted and hence we do not have to demand that the p-rules are weakly decreasing.

So similar to the approach in Section 2.5 we may replace a dependency pair $\langle s,t\rangle$ by all its narrowings provided that the right-hand side t does not unify with any left-hand side of a dependency pair. In fact, due to the restriction to *innermost* chains we may even perform such a replacement if t unifies with the left-hand side v of a dependency pair, as long as their mgu does not instantiate both s and v to normal forms. Note that in contrast to the termination case, for innermost termination proofs we do not have to demand that t must be a linear term. Hence, we can indeed narrow the dependency pair (27) in the above example, although its right-hand side is not linear. However, this step would not have been possible with the method of Section 2. Therefore, for the TRS in Example 41 the constraints generated by the approach of Section 2 are not satisfied by any quasi-simplification ordering.

Theorem 42 (Narrowing refinement for innermost termination). *Let $\mathscr{R}$ be a TRS and let $\mathscr{P}$ be a set of pairs of terms. Let $\langle s,t\rangle \in \mathscr{P}$ such that all variables of t also occur in s and such that for all $\langle v,w\rangle \in \mathscr{P}$ where t and v are unifiable by some mgu μ (after renaming the variables), one of the terms $s\mu$ or $v\mu$ is no normal form. Let*

$$\mathscr{P}' = \mathscr{P}\backslash\{\langle s,t\rangle\} \cup \{\langle s',t'\rangle \mid \langle s',t'\rangle \text{ is a narrowing of } \langle s,t\rangle\}.$$

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

Proof. The proof is analogous to the proof of Theorem 27. The only difference is that the right-hand side t of the dependency pair does not have to be linear any more. The reason is that in innermost chains we restrict ourselves to normal substitutions σ and therefore, reductions of $t\sigma$ can never take place 'in σ' (as all variables of t also occur in s). □

Note that unlike Theorem 27 for termination, the replacement of dependency pairs by their narrowings can destroy the necessity of our innermost termination criterion. The reason is that narrowing does not respect the *innermost* reduction strategy.

Example 43. The TRS in Example 34 was innermost terminating. Hence, there does not exist an infinite innermost chain of dependency pairs. However, if we replace the

dependency pair $\langle \mathsf{F}(\mathsf{s}(x)), \mathsf{F}(\mathsf{g}(\mathsf{h}(x)))\rangle$ by its narrowings

$$\langle \mathsf{F}(\mathsf{s}(\mathsf{0})), \mathsf{F}(\mathsf{g}(\mathsf{a}))\rangle \tag{34}$$

$$\langle \mathsf{F}(\mathsf{s}(x)), \mathsf{F}(\mathsf{g}(x))\rangle, \tag{35}$$

then there exists an infinite innermost chain consisting of the new dependency pair (35), because $\mathsf{F}(\mathsf{g}(x_1))\sigma \xrightarrow{\mathrm{i}}_{\mathscr{R}} \mathsf{F}(\mathsf{s}(x_2))\sigma$ holds if σ instantiates x_1 and x_2 by $\mathsf{0}$. (In particular, if (35) is again replaced by its narrowings, then we obtain the new pair $\langle \mathsf{F}(\mathsf{s}(\mathsf{0})), \mathsf{F}(\mathsf{s}(\mathsf{0}))\rangle$ which obviously forms an infinite innermost chain.) Thus, although $\mathsf{g}(\mathsf{h}(x))$ has no redex as a proper subterm, narrowing this term leads to a failure of the innermost termination proof.

So there are examples where narrowing transforms a set of dependency pairs without infinite innermost chains into a new set of pairs which form an infinite innermost chain. However, this can only happen for examples where the *automation* of our method would have failed anyway, i.e. where the constraints generated without using narrowing would already have been unsatisfiable (as in Example 34). More precisely, if we use the approach of Theorem 37 and if we approximate innermost dependency graphs by computing the innermost connectable pairs (Theorem 39), then every ordering satisfying the constraints generated without narrowing also satisfies the constraints generated after narrowing dependency pairs. In fact, every constraint obtained when using narrowing is implied by the constraints that one would obtain without narrowing. (The reason is that if $\langle s', t'\rangle$ and $\langle v', w'\rangle$ are narrowings of $\langle s, t\rangle$ and $\langle v, w\rangle$ respectively, then $\langle s, t\rangle$ is innermost connectable to $\langle v, w\rangle$ whenever $\langle s', t'\rangle$ is innermost connectable to $\langle v', w'\rangle$.) Hence, the application of narrowing can only extend the number of systems where innermost termination can be proved automatically.

3.5. Summary

Combining all refinements, our technique to prove innermost termination automatically using the dependency pair approach works as follows:

- Determine the dependency pairs.
- Replace some (dependency) pairs by all their narrowings. Again, this step could be repeated several times.
- Approximate the innermost dependency graph by estimating for all (dependency) pairs whether an arc exists between two of them. For that purpose we introduced the function CAP_s.
- Compute the usable rules $\mathbf{U}$, i.e. (a superset of) those rules that can be used for the reductions between two (dependency) pairs.
- Transform the usable rules and the (dependency) pairs on cycles into inequalities.
- Find a well-founded quasi-ordering satisfying the inequalities after normalizing them with respect to one of the possible AFSs.

As for the termination approach, standard techniques like the recursive path ordering or polynomial interpretations can be used to find these orderings. However, since the

ordering need not be weakly monotonic for tuple symbols, we may also search for different kinds of orderings, such as polynomial interpretations where some polynomials have negative coefficients.

Our approach is the first automatic method which can also prove innermost termination of TRSs that are not terminating. Moreover, for those classes of TRSs where innermost termination already implies termination, the technique described in this section can also be used for termination proofs. In particular, this holds for non-overlapping or at least locally confluent overlay systems. The difference to the termination technique is that we only need to prove absence of infinite *innermost* chains. For that reason several steps in the technique are different to the technique of Section 2:

- Right-hand sides of narrowed (dependency) pairs do not have to be linear and they may unify with left-hand sides as long as their mgu does not instantiate the left-hand sides to normal forms.
- For computing the innermost dependency graph instead of the functions REN and CAP we use the function CAP_s.
- We restrict ourselves to the usable rules when transforming the rules into inequalities.
- The quasi-ordering that has to be found in the end need not be weakly monotonic on tuple symbols (unless explicitly demanded).

As long as the system is non-overlapping it is always advantageous to prove innermost termination only (instead of termination). The reason is that every ordering satisfying the constraints of the termination technique in Section 2 also satisfies the constraints of our innermost termination technique, but not vice versa. For instance, termination of the systems in Examples 35 and 41 can easily be proved with the technique introduced in this section, whereas the constraints generated by the method of Section 2 are not satisfied by any quasi-simplification ordering. A collection of examples demonstrating the power of our technique to prove innermost termination can be found in [6].

4. Conclusion and related work

We have introduced techniques to prove termination and innermost termination of term rewriting systems automatically. For that purpose we have developed sufficient and necessary criteria for both termination and innermost termination. To automate the checking of these criteria, a set of constraints is synthesized for each TRS and standard techniques developed for termination proofs can be used to generate a well-founded ordering satisfying these constraints. If such an ordering can be found, then termination resp. innermost termination of the system is proved.

Most other methods for automated termination proofs are restricted to *simplification orderings*. Compared to proving termination directly, our approach has the advantage that the constraints generated by our method are often satisfied by standard (simplification) orderings, even if termination of the original TRS cannot be proved with these orderings. Moreover, for all those TRSs where termination can be proved with a simplification ordering directly, this simplification ordering also satisfies the inequalities resulting from our technique. Therefore, instead of using simplification orderings for

direct termination proofs, it is always advantageous to combine them with the technique presented in this paper.

We implemented our technique for the generation of constraints and in this way termination could be proved automatically for many challenge problems from literature as well as for practically relevant TRSs from different areas of computer science. See [6] for a collection of numerous such examples, including arithmetical operations (e.g. mod, gcd, logarithm, average), sorting algorithms (such as selection sort, minimum sort, and quicksort), algorithms on graphs and trees, and several other well-known non-simply terminating TRSs (e.g. from [16, 17, 44]).

Our termination criteria are based on the notion of *dependency pairs*. The concept of dependency pairs was introduced in [1] and a first method for its automation was proposed in [3]. For that purpose, we transferred the *estimation technique* [24, 25], which was originally developed for termination proofs of functional programs, to rewrite systems. However, this first method was restricted to non-overlapping constructor systems without nested recursion. In this approach, the dependency pair technique was based on a special form of *semantic labelling* (cf. [48]), called self-labelling (similar to the notion of self-labelling in [40]). Self-labelling determines unique labels for the terms and a dependency pair can be regarded as a combination of the label for the left-hand side with the labels for the right-hand side of a rule.

In [4] we developed a refined framework for dependency pairs which is independent from semantic labelling. Therefore, this framework is better suited for automation (as one does not have to construct an appropriate semantic interpretation any more) and its soundness can be proved in a much easier and shorter way. Moreover, in this framework we could show that our technique is applicable to arbitrary TRSs and we proved that the formulated criterion (Theorem 6) is not only sufficient, but also necessary for termination.

The present paper extends the approach of [4] by the introduction of argument filtering TRSs, the addition of narrowing dependency pairs, and by proving that the whole approach up to the search for suitable quasi-orderings is sound and *complete*, i.e., the inequalities for which an ordering should be found by standard techniques are satisfiable if and only if the TRS is terminating. This result suggests that the transformation described in this paper should always be applied before using any of the standard techniques for termination proofs.

In [5] we presented a modification of the framework, in which the notion of chains was restricted to innermost chains and we showed that a TRS is innermost terminating if and only if no infinite innermost chains exist for the TRS. This approach is the first automatic method which can also prove innermost termination of systems that are not terminating. Moreover, our technique can very successfully be used for termination proofs of non-overlapping systems, because for those systems innermost termination is already sufficient for termination.

In the present paper we extended the technique described in [5] by a refined definition of innermost dependency graphs, a method to compute better approximations of these graphs, and a more powerful approach for narrowing dependency pairs. In [6]

we give a collection of several examples which can now be proved terminating resp. innermost terminating automatically, but where automatic proofs using the techniques in [4, 5] failed.

We have presented a sound and complete termination criterion. In contrast to most other complete approaches (semantic path ordering [31], general path ordering [17], semantic labelling [48], etc.) our method is particularly well suited for automation as has been demonstrated in this paper. The only other complete criterion that has been used for automatic termination proofs (by Steinbach [44]) is the approach of *transformation orderings* [9, 10]. It turns out that the termination of several examples where the automation of Steinbach failed can be proved by our technique automatically, cf. [6].

At first sight there seem to be some similarities between our method and *forward closures* [17, 38]. The idea of forward closures is to restrict the application of rules to that part of a term created by previous rewrites. Similar to our notion of chains, this notion also results in a sequence of terms, but these sequences have completely different semantics. For example, forward closures are reductions whereas in general the terms in a chain do not form a reduction. The reason is that in the dependency pair approach we do not restrict the *application of rules*, but we restrict the examination of *terms* to those subterms that can possibly be reduced further. Compared to the forward closure approach, the dependency pair technique has the advantage that it can be used for *arbitrary* TRSs, whereas the absence of infinite forward closures only implies termination for right-linear [14] or non-overlapping [22] TRSs. Moreover, in contrast to the dependency pair method, we do not know of any attempt to automate the forward closure approach.

The framework of dependency pairs, as introduced in this paper, is very general and is therefore well suited to be used for more general rewriting problems, too. For example, the framework of dependency pairs can easily be extended for termination modulo associativity and commutativity [39]. Moreover, several well-known and new modularity results can be derived in this framework [2, 7, 26].

Acknowledgements

We would like to thank Hans Zantema, Aart Middeldorp, Thomas Kolbe, and Bernhard Gramlich for constructive criticism and many helpful comments. This work was partially supported by the Deutsche Forschungsgemeinschaft under grants no. Wa 652/7-1,2 as part of the focus program ‘Deduktion’.

References

[1] T. Arts, Termination by absence of infinite chains of dependency pairs, in: H. Kirchner (Ed.), Proc. 21st Internat. Colloquium on Trees in Algebra and Programming, CAAP ’96, Lecture Notes in Computer Science, vol. 1059, Linköping, Sweden, April, Springer, Berlin, 1986, pp. 196–210.

[2] T. Arts, Automatically proving termination and innermost normalisation of term rewriting systems, Ph.D. Thesis, Utrecht University, The Netherlands, May 1997.

[3] T. Arts, J. Giesl, Termination of constructor systems, in: H. Ganzinger (Ed.), Proc. 7th Internat. Conf. on Rewriting Techniques and Applications, RTA-96, Lecture Notes in Computer Science, vol. 1103, New Brunswick, NJ, USA, July, Springer, Berlin, 1996, pp. 63–77.

[4] T. Arts, J. Giesl, Automatically proving termination where simplification orderings fail, in: M. Dauchet (Ed.), Proc. 7th Internat. Joint Conf. on the Theory and Practice of Software Development, TAPSOFT '97, Lecture Notes in Computer Science, vol. 1214, Lille, France, April, Springer, Berlin, 1997, pp. 261–272.

[5] T. Arts, J. Giesl, Proving innermost normalisation automatically, in: H. Comon (Ed.), Proc. 8th Internat. Conf. on Rewriting Techniques and Applications, RTA-97, Lecture Notes in Computer Science, vol. 1232, Sitges, Spain, June, Springer, Berlin, 1997, pp. 157–171.

[6] T. Arts, J. Giesl, Termination of term rewriting using dependency pairs, Tech. Report IBN 97/46, Darmstadt University of Technology, Germany, September, 1997. http://www.inferenzsysteme.informatik.tu-darmstadt.de.

[7] T. Arts, J. Giesl, Modularity of termination using dependency pairs, in: T. Nipkow (Ed.), Proc. 9th Internat. Conf. Rewriting Techniques and Applications, RTA-98, Lecture Notes in Computer Science, vol. 1379, Tsukuba, Japan, March/April, Springer, Berlin, 1988, pp. 226–240.

[8] T. Arts, H. Zantema, Termination of logic programs using semantic unification, in: M. Proietti (Ed.), Proc. 5th Internat. Workshop on Logic Program Synthesis and Transformation, LoPSTr '95, Lecture Notes in Computer Science, vol. 1048, Utrecht, The Netherlands, September, Springer, Berlin, 1995, pp. 219–233.

[9] L. Bachmair, N. Dershowitz, Commutation, transformation and termination, in: J.H. Siekmann (Ed.), Proc. 8th Internat. Conf. on Automated Deduction, CADE-8, Lecture Notes in Computer Science, vol. 230, Oxford, England, July, Springer, Berlin, 1986, pp. 5–20.

[10] F. Bellegarde, P. Lescanne, Termination by completion, Appl. Algebra Eng. Comm. Comput. 1 (1990) 79–96.

[11] A. Ben Cherifa, P. Lescanne, Termination of rewriting systems by polynomial interpretations and its implementation, Sci. Comput. Programm. 9 (1987) 137–159.

[12] E. Bevers, J. Lewi, Proving termination of (conditional) rewrite systems, Acta Inform. 30 (1993) 537–568.

[13] B. Courcelle, Recursive applicative program schemes, in: J. van Leeuwen (Ed.), Formal Models and Semantics, Handbook of Theoretical Computer Science, vol. B, North-Holland, Amsterdam, 1990, pp. 459–492.

[14] N. Dershowitz, Termination of linear rewriting systems, in: S. Even, O. Kariv (Eds.), Proc. 8th Internat. Coll. on Automata, Languages and Programming, ICALP '81, Lecture Notes in Computer Science, vol. 115, Acre, Israel, July, Springer, Berlin, 1981, pp. 448–458.

[15] N. Dershowitz, Orderings for term-rewriting systems, Theoret. Comput. Sci. 17 (1982) 279–301.

[16] N. Dershowitz, Termination of rewriting, J. Symbolic Comput. 3 (1 and 2) (1987) 69–116.

[17] N. Dershowitz, C. Hoot, Natural termination, Theoret. Comput. Sci. 142(2) (1995) 179–207.

[18] N. Dershowitz, J.-P. Jouannaud, Rewrite systems, in: J. van Leeuwen (Ed.), Formal Models and Semantics, Handbook of Theoretical Computer Science, vol. B, North-Holland, Amsterdam, 1990, pp. 243–320.

[19] J. Dick, J. Kalmus, U. Martin, Automating the Knuth Bendix ordering, Acta Inform. 28 (1990) 95–119.

[20] M. Ferreira, H. Zantema, Syntactical analysis of total termination, in: G. Levi, M. Rodríguez-Artalejo (Eds.), Proc. 4th Internat. Conf. on Algebraic and Logic Programming, ALP '94, Lecture Notes in Computer Science, vol. 850, Madrid, Spain, September, Springer, Berlin, 1994, pp. 204–222.

[21] M. Ferreira, H. Zantema, Dummy Elimination: making termination easier, in: H. Reichel (Ed.), Proc. 10th Internat. Conf. on Fundamentals of Computation Theory, FCT '95, Lecture Notes in Computer Science, Dresden, Germany, August, Springer, Berlin, 1995, pp. 243–252.

[22] O. Geupel, Overlap closures and termination of term rewriting systems, Tech. Report MIP-8922 283, Universität Passau, Passau, Germany, 1989.

[23] J. Giesl, Generating polynomial orderings for termination proofs, in: J. Hsiang (Ed.), Proc. 6th Internat. Conf. on Rewriting Techniques and Applications, RTA-95, Lecture Notes in Computer Science, vol. 914, Kaiserslautern, Germany, April, Springer, Berlin, 1995, pp. 426–431.

[24] J. Giesl, Termination analysis for functional programs using term orderings, in: A. Mycroft (Ed.), Proc. 2nd Internat. Static Analysis Symp., SAS '95, Lecture Notes in Computer Science, vol. 983, Glasgow, UK, September, Springer, Berlin, 1995, pp. 154–171.
[25] J. Giesl, Termination of nested and mutually recursive algorithms, J. Automat. Reason. 19 (1997) 1–29.
[26] J. Giesl, E. Ohlebusch, Pushing the frontiers of combining rewrite systems farther outwards, in: Proc. 2nd Internat. Workshop on Frontiers of Combining Systems, FroCoS '98, Logic and Computation Series, Amsterdam, The Netherlands, October, Research Studies Press, John Wiley & Sons, 1999.
[27] B. Gramlich, Abstract relations between restricted termination and confluence properties of rewrite systems, Fund. Inform. 24 (1995) 3–23.
[28] B. Gramlich, On proving termination by innermost termination, in: H. Ganzinger (Ed.), Proc. 7th Internat. Conf. on Rewriting Techniques and Applications, RTA-96, Lecture Notes in Computer Science, vol. 1103, New Brunswick, NJ, USA, July, Springer, Berlin, 1996, pp. 93–107.
[29] G. Huet, D. Lankford, On the uniform halting problem for term rewriting systems, Tech. Report 283, INRIA, Le Chesnay, France, 1978.
[30] J.M. Hullot, Canonical forms and unification, in: W. Bibel, R. Kowalski (Eds.), Proc. 5th Internat. Conf. on Automated Deduction, CADE-5, Lecture Notes in Computer Science, vol. 87, Les Arcs, France, July, Springer, Berlin, 1980, pp. 318–334.
[31] S. Kamin, J.-J. Lévy, Two generalizations of the recursive path ordering, Department of Computer Science, University of Illinois, IL, 1980.
[32] J.W. Klop, Term rewriting systems, in: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (Eds.), Background: Computational Structures, Handbook of Logic in Computer Science, vol. 2, Oxford University Press, New York, 1992, pp. 1–116.
[33] D.E. Knuth, P.B. Bendix, Simple word problems in universal algebras, in: J. Leech (Ed.), Computational Problems in Abstract Algebra, Pergamon Press, Oxford, 1970, pp. 263–297.
[34] T. Kolbe, Challenge problems for automated termination proofs of term rewriting systems, Tech. Report IBN 96/42, Darmstadt University of Technology, Germany, 1996.
[35] M.R.K. Krishna Rao, Modular proofs for completeness of hierarchical term rewriting systems, Theoret. Comput. Sci. 151 (1995) 487–512.
[36] M.R.K. Krishna Rao, Some characteristics of strong innermost normalization, in: M. Wirsing, M. Nivat (Eds.), Proc. 5th Internat. Conf. on Algebraic Methodology and Software Technology, AMAST '96, Lecture Notes in Computer Science, vol. 1101, Munich, Germany, July, Springer, Berlin, 1996, pp. 406–420.
[37] D.S. Lankford, On proving term rewriting systems are Noetherian, Tech. Report Memo MTP-3, Louisiana Technical University, Ruston, LA, 1979.
[38] D.S. Lankford, D.R. Musser, A finite termination criterion, 1978.
[39] C. Marché, X. Urbain, Termination of associative-commutative rewriting by dependency pairs, in: T. Nipkow (Ed.), Proc. 9th Internat. Conf. on Rewriting Techniques and Applications, RTA-98, Lecture Notes in Computer Science, vol. 1397, Tsukuba, Japan, March/April, Springer, Berlin, 1998, pp. 241–255.
[40] A. Middeldorp, H. Ohsaki, H. Zantema, Transforming termination by self-labelling, in: M.A. McRobbie, J.K. Slaney (Eds.), Proc. 13th Internat. Conf. on Automated Deduction, CADE-13, Lecture Notes in Artificial Intelligence, vol. 1104, New Brunswick, NJ, USA, July/August, Springer, Berlin, 1996, pp. 373–387.
[41] A. Middeldorp, H. Zantema, Simple termination of rewrite systems, Theoret. Comput. Sci. 175 (1997) 127–158.
[42] D.A. Plaisted, A recursively defined ordering for proving termination of term rewriting systems, Tech. Report R-78-943, Department of Computer Science, University of Illinois, Urbana-Champaign, IL, 1978.
[43] J. Steinbach, Generating polynomial orderings, Inform. Process. Lett. 49 (1994) 85–93.
[44] J. Steinbach, Automatic termination proofs with transformation orderings, in: J. Hsiang (Ed.), Proc. 6th Internat. Conf. on Rewriting Techniques and Applications, RTA-95, Lectures Notes in Computer Science, vol. 914, Kaiserslautern, Germany, April, Springer, Berlin, 1995, pp. 11–25. Full Version appeared as Tech. Report SR-92-23, Universität Kaiserslautern, Germany, 1992.
[45] J. Steinbach, Simplification orderings: history of results, Fund. Inform. 24 (1995) 47–87.

[46] Y. Toyama, Counterexamples to the termination for the direct sum of term rewriting systems, Inform. Process. Lett. 25 (1987) 141–143.
[47] H. Zantema, Termination of term rewriting: interpretation and type elimination, J. Symbolic Comput. 17 (1994) 23–50.
[48] H. Zantema, Termination of term rewriting by semantic labelling, Fund. Inform. 24 (1995) 89–105.