Appeared in *Applicable Algebra in Engineering, Communication and Computing*, 16(4):229-270, 2005.

The Size-Change Principle and Dependency Pairs for Termination of Term Rewriting

René Thiemann and Jürgen Giesl

LuFG Informatik II, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany
E-mail: {thiemann|giesl}@informatik.rwth-aachen.de

The date of receipt and acceptance will be inserted by the editor

Abstract In [24], a new *size-change principle* was proposed to verify termination of functional programs automatically. We extend this principle in order to prove termination and innermost termination of arbitrary term rewrite systems (TRSs). Moreover, we compare this approach with existing techniques for termination analysis of TRSs (such as recursive path orders or dependency pairs). It turns out that the size-change principle on its own fails for many examples that can be handled by standard techniques for rewriting, but there are also TRSs where it succeeds whereas existing rewriting techniques fail. Moreover, we also compare the complexity of the respective methods. To this end, we develop the first complexity analysis for the dependency pair approach. While the size-change principle is PSPACE-complete, we prove that the dependency pair approach (in combination with classical path orders) is only NP-complete. To benefit from their respective advantages, we show how to combine the size-change principle with classical orders and with dependency pairs. In this way, we obtain a new approach for automated termination proofs of TRSs which is more powerful than previous approaches. We also show that the combination with dependency pairs does not increase the complexity of the size-change principle, i.e., the combined approach is still PSPACE-complete.

Keywords: Termination, Term Rewriting, Size-Change Principle, Dependency Pairs

1 Introduction

The size-change principle [24] is a new technique for automated termination analysis of functional programs, which raised great interest in the functional programming and automated reasoning community. Moreover, a similar principle is also used in approaches for termination analysis of logic programs, e.g., [8]. However, up to now the connection between this principle and existing approaches for termination proofs of term rewriting was unclear. We introduce the basics of term rewriting and the size-change principle in Section 2. Then we show in Section 3 which orders may be used in connection with the size-change principle in order to yield a sound method for termination and innermost termination proofs of arbitrary TRSs. This also illustrates how to combine the size-change principle with existing orders from term rewriting.

In Section 4 we compare the size-change principle with classical simplification orders and show that it can simulate a certain form of lexicographic and multiset comparison. Hence, the size-change principle in connection with a very simple order can often prove termination of TRSs where one would otherwise need more complex orders like the lexicographic or the recursive path order. On the other hand, there are also examples where termination can be proved with classical orders from term rewriting, while the size-change principle does not succeed.

In Section 5 we compare the size-change principle and the dependency pair approach [2] for termination of TRSs. Again, the size-change principle can simulate and encompass certain ingredients of the dependency pair method and there are examples where a termination proof with the size-change principle is trivial, whereas dependency pairs do not succeed with any classical order amenable to automation. On the other hand, there are many TRSs where dependency pairs can easily prove termination, whereas the size-change principle fails.

To combine their respective advantages, in Section 6 we develop a technique which integrates the size-change principle and dependency pairs. The resulting technique improves upon both original techniques, since the constraints generated for termination proofs are considerably simplified. We show that in this way, one can handle examples that could not be proved terminating before. Moreover, for examples which could already be handled with dependency pairs, our new combination technique often succeeds in connection with much simpler orders than those required when using dependency pairs.

In contrast to other recent techniques for termination analysis, the complexity of the size-change principle has been formally analyzed in [24]. In Section 7 we show that such a complexity analysis is also possible for both the dependency pair approach and for the new technique from Section 6 which combines dependency pairs with the size-change principle. More precisely, while the size-change principle is PSPACE-complete, we show that the dependency pair approach is NP-complete (provided one uses the dependency pair approach with argument filterings, with base orders in NP, and with standard estimations of the dependency graph). Moreover, our combination of dependency pairs and the size-change principle is still PSPACE-complete (if one again uses argument filterings, base orders in NP or PSPACE, and standard estimations of the dependency graph).

The combined technique has been implemented in the system AProVE [13] resulting in a very efficient and powerful automated method which improves the original dependency pair approach significantly. A description of the implementation and an empirical evaluation can be found in Section 8.

A preliminary version of this paper appeared in [30]. The current article extends [30] by numerous new results and refinements: We prove that our definition of size-change termination corresponds to the one of [24] (Lemma 6). Moreover, we give formal proofs for the comparison of the size-change principle with standard rewrite orders based on lexicographic or multiset comparison (Theorems 14 and 16). We also present a large example to demonstrate the advantages of combining dependency pairs and the size-change principle (Example 26). Another new contribution is Section 7 where we present the first complexity results for the dependency pair approach and examine the complexity of the combination with the size-change principle. Finally, we added a section on the implementation and evaluation of our results (Section 8).

2 Term Rewriting and the Size-Change Principle

We first recapitulate the basics of term rewriting in Section 2.1 and introduce the size-change principle in Section 2.2.

2.1 Term Rewriting

This section briefly introduces the basic notions of term rewriting. For further details and explanations we refer to [4], for example.

A *signature* $\mathcal{F}$ is a set of function symbols and for a set of variables $\mathcal{V}$, $\mathcal{T}(\mathcal{F},\mathcal{V})$ denotes the *terms* built from $\mathcal{F}$ and $\mathcal{V}$. For a term t, $\mathcal{V}(t)$ is the set of variables occurring in t and for $t \notin \mathcal{V}$, the *root symbol* $\mathrm{root}(t)$ denotes the topmost symbol of t (i.e., $\mathrm{root}(f(s_1,\ldots,s_n)) = f$). As usual, a *ground term* is a term without variables and $\rhd$ denotes the *proper subterm relation.*

A *term rewrite system* (TRS) over a signature $\mathcal{F}$ is a set of *rules* $l \to r$, where $l, r \in \mathcal{T}(\mathcal{F},\mathcal{V})$, $l \notin \mathcal{V}$, and $\mathcal{V}(r) \subseteq \mathcal{V}(l)$. Throughout the paper, we restrict ourselves to finite signatures and TRSs.

For a TRS $\mathcal{R}$, one can define the corresponding *reduction relation* $\to_\mathcal{R}$ on terms (i.e., $\to_\mathcal{R} \subseteq \mathcal{T}(\mathcal{F},\mathcal{V}) \times \mathcal{T}(\mathcal{F},\mathcal{V})$): for any $s,t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ we have $s \to_\mathcal{R} t$ iff there exists a position p in s and a rule $l \to r \in \mathcal{R}$ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. In other words, the left-hand side l matches the subterm $s|_p$ with the matching substitution σ and t results from s by replacing this subterm by the right-hand side of the rule instantiated by the matcher σ. Moreover, if no proper subterms of $s|_p$ are reducible, then we speak of an *innermost* reduction step, denoted $s \xrightarrow{\text{i}}_\mathcal{R} t$. A term s is a *normal form* if it cannot be reduced anymore, i.e., if $s \to_\mathcal{R} t$ does not hold for any term t.

We denote the transitive closure of a relation $\to$ by $\to^+$ and the transitive and reflexive closure is denoted by $\to^*$. So $s \to_{\mathcal{R}}^* t$ means that s can be reduced to t in several (possibly zero) steps.

For a TRS $\mathcal{R}$ over a signature $\mathcal{F}$, the *defined symbols* $\mathcal{D}$ are the root symbols of the left-hand sides of rules and the *constructors* are $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$. A TRS is called a *constructor system* if the left-hand sides of its rules are terms of the form $f(s_1, \ldots, s_n)$ where all s_i are *constructor terms* (i.e., $s_i \in \mathcal{T}(\mathcal{C}, \mathcal{V})$). For any signature $\mathcal{F}$ we define the *embedding rules* $\mathit{Emb}_{\mathcal{F}} = \{f(x_1, \ldots, x_n) \to x_i \mid f \in \mathcal{F} \text{ where } n = \text{arity}(f), 1 \le i \le n\}$. A TRS is *non-overlapping* if there are no rules $l \to r$ and $l' \to r'$ such that a non-variable subterm of l unifies with l'; however, if the two rules are identical, then one only needs to consider proper subterms of l.

A transitive and antisymmetric binary relation $\succ$ is an *order* and a transitive and reflexive binary relation is a *quasi-order*. A binary relation $\succ$ is *well founded* iff there exists no infinite decreasing sequence $t_0 \succ t_1 \succ t_2 \succ \ldots$ A binary relation $\succ$ on terms is *closed under substitutions* (also called "*stable*") iff $s \succ t$ implies $s\sigma \succ t\sigma$ for all $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and all substitutions σ. A binary relation $\succ$ on terms is *closed under contexts* (also called "*monotonic*") if $s_i \succ t_i$ implies $f(s_1, \ldots, s_i, \ldots, s_n) \succ f(s_1, \ldots, t_i, \ldots, s_n)$ for all $f \in \mathcal{F}$, all $1 \le i \le \text{arity}(f)$, and all terms s_j and t_j.

A TRS $\mathcal{R}$ is *terminating* iff $\to_{\mathcal{R}}$ is well founded and it is *innermost terminating* iff $\xrightarrow{\text{i}}_{\mathcal{R}}$ is well founded. The traditional method to prove termination of TRSs works by generating a suitable order such that all rules are decreasing: a TRS $\mathcal{R}$ is terminating iff there exists a well-founded order $\succ$ closed under substitutions and contexts such that $l \succ r$ holds for all rules $l \to r \in \mathcal{R}$ [25]. Most of the orders used for automation are *simplification orders*. A simplification order is a well-founded order $\succ$ closed under substitutions and contexts which also satisfies the subterm property $f(x_1, \ldots, x_n) \succ x_i$ for all $f \in \mathcal{F}$, all $1 \le i \le \text{arity}(f)$, and pairwise different variables x_j. Examples for such orders include lexicographic path orders *LPO* [17], recursive path orders (possibly with status) $\mathit{RPO}(S)$ [6], Knuth-Bendix orders *KBO* [18], and many polynomial orders [22].

2.2 The Size-Change Principle

In [24], the size-change principle was presented for a functional programming language with eager evaluation strategy and without pattern matching. Such functional programs are easily transformed into TRSs which are non-overlapping constructor systems whose ground normal forms only contain constructors (i.e., all functions are "completely" defined). In this section we introduce an extension of the original size-change principle which is formulated for arbitrary TRSs.

We call $(\succsim, \succ)$ a *reduction pair* on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ if $\succsim$ is a quasi-order and $\succ$ is a well-founded order on terms where both $\succsim$ and $\succ$ are closed under substitutions and compatible (i.e., $\succsim \circ \succ \subseteq \succ$ and $\succ \circ \succsim \subseteq \succ$, but $\succ \subseteq \succsim$ is not required). In contrast to the definition of reduction pairs in [21], neither $\succsim$ nor $\succ$ have to be closed under contexts. If $\succsim$ is closed under contexts, we speak of a *monotonic* reduction pair. In Section 3 we examine which additional conditions must be imposed on $(\succsim, \succ)$ in order to use the size-change principle for (innermost) termination proofs of TRSs. *Size-change graphs* denote how the size of function parameters changes when going from one function call to another.

Definition 1 (Size-Change Graph) *Let $(\succsim, \succ)$ be a reduction pair. For every rule $f(s_1, \ldots, s_n) \to r$ of a TRS $\mathcal{R}$ and every subterm $g(t_1, \ldots, t_m)$ of r where $g \in \mathcal{D}$, we define a* size-change graph*. The graph has n* output nodes *marked with $\{1_f, \ldots, n_f\}$ and m* input nodes *marked with $\{1_g, \ldots, m_g\}$. Output nodes are nodes where one may have outgoing edges and input nodes may have incoming edges. If $s_i \succ t_j$, then there is a directed edge marked with "$\succ$" from output node i_f to input node j_g. Otherwise, if $s_i \succsim t_j$, then there is an edge marked with "$\succsim$" from i_f to j_g. If f and g are clear from the context, then we often omit the subscripts from the nodes. So a size-change graph is a bipartite graph $G = (V, W, E)$ where $V = \{1_f, \ldots, n_f\}$ and $W = \{1_g, \ldots, m_g\}$ are the labels of the output and input nodes, respectively, and we have edges $E \subseteq V \times W \times \{\succsim, \succ\}$.*

Example 2 Let $\mathcal{R}$ consist of the following rules.

$$\mathsf{f}(\mathsf{s}(x), y) \to \mathsf{f}(x, \mathsf{s}(x)) \qquad (1) \qquad\qquad \mathsf{f}(x, \mathsf{s}(y)) \to \mathsf{f}(y, x) \qquad (2)$$

$\mathcal{R}$ has two size-change graphs $G_{(1)}$ and $G_{(2)}$ resulting from the rules (1) and (2). Here, we use the embedding order on constructors $\mathcal{C}$, i.e., $(\succsim, \succ) = (\to^*_{Emb_\mathcal{C}}, \to^+_{Emb_\mathcal{C}})$.

$$G_{(1)}: \quad \begin{tikzcd} 1_\mathsf{f} \arrow[r, "\succ"] \arrow[dr, "\succsim"] & 1_\mathsf{f} \\ 2_\mathsf{f} & 2_\mathsf{f} \end{tikzcd} \qquad\qquad G_{(2)}: \quad \begin{tikzcd} 1_\mathsf{f} \arrow[dr, "\succsim"] & 1_\mathsf{f} \\ 2_\mathsf{f} \arrow[ur, "\succ"'] & 2_\mathsf{f} \end{tikzcd}$$

To trace the sizes of parameters along subsequent function calls, size-change graphs (V_1, W_1, E_1) and (V_2, W_2, E_2) can be concatenated to *multigraphs* if $W_1 = V_2$, i.e., if they correspond to arguments $\{1_g, \ldots, m_g\}$ of the same function g.

Definition 3 (Multigraph and Concatenation) *For a TRS $\mathcal{R}$, every size-change graph of $\mathcal{R}$ is a* multigraph *of $\mathcal{R}$ and if $G = (\{1_f, \ldots, n_f\}, \{1_g, \ldots, m_g\}, E_1)$ and $H = (\{1_g, \ldots, m_g\}, \{1_h, \ldots, l_h\}, E_2)$ are multigraphs w.r.t. the same reduction pair $(\succsim, \succ)$, then the* concatenation *$G \bullet H = (\{1_f, \ldots, n_f\}, \{1_h, \ldots, l_h\}, E)$ is also a multigraph of $\mathcal{R}$. For $1 \le i \le n$ and $1 \le k \le l$, E contains an edge from i_f to k_h iff E_1 contains an edge from i_f to some j_g and E_2 contains an edge from j_g to k_h. If there is such a j_g where the edge of E_1 or E_2 is labelled with "$\succ$", then the edge in E is labelled with "$\succ$" as well. Otherwise, it is labelled with "$\succsim$". A multigraph G is called* maximal *if its input and output nodes are both labelled with $\{1_f, \ldots, n_f\}$ for some f and if it is idempotent, i.e., $G = G \bullet G$.*

Example 4 In Example 2 there are three maximal multigraphs (note that $G_{(1)} \bullet G_{(1)} = G_{(1)} \bullet G_{(2)}$):

$$G_{(1)} \bullet G_{(2)} : \begin{tikzcd} 1_{\mathsf{f}} \arrow[r, "\succ"] \arrow[dr, "\succ"] & 1_{\mathsf{f}} \\ 2_{\mathsf{f}} & 2_{\mathsf{f}} \end{tikzcd} \qquad G_{(2)} \bullet G_{(1)} : \begin{tikzcd} 1_{\mathsf{f}} & 1_{\mathsf{f}} \\ 2_{\mathsf{f}} \arrow[ur, "\succ"] \arrow[r, "\succ"'] & 2_{\mathsf{f}} \end{tikzcd} \qquad G_{(2)} \bullet G_{(2)} : \begin{tikzcd} 1_{\mathsf{f}} \arrow[r, "\succ"] & 1_{\mathsf{f}} \\ 2_{\mathsf{f}} \arrow[r, "\succ"] & 2_{\mathsf{f}} \end{tikzcd}$$

For termination, in every maximal multigraph some parameter must be decreasing.[1]

Definition 5 (Size-Change Termination) *A TRS $\mathcal{R}$ over the signature $\mathcal{F}$ is* size-change terminating *w.r.t. a reduction pair $(\succsim, \succ)$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ iff every maximal multigraph contains an edge of the form $i \xrightarrow{\succ} i$.*

In Example 4, each maximal multigraph contains the edge $1_{\mathsf{f}} \xrightarrow{\succ} 1_{\mathsf{f}}$ or $2_{\mathsf{f}} \xrightarrow{\succ} 2_{\mathsf{f}}$. So the TRS is size-change terminating w.r.t. the embedding order. Note that classical path orders from term rewriting fail on this example (see Section 4).

Since there are only finitely many possible multigraphs, they can be constructed automatically. So for a given reduction pair $(\succsim, \succ)$ where $\succsim$ and $\succ$ are decidable, size-change termination is decidable as well. While the formulation of size-change termination in Definition 5 is suitable for automation and for a comparison with techniques from term rewriting, size-change termination was defined in a slightly different way in [24]. Here, instead of concatenating size-change graphs $G_1, \ldots, G_n$, one builds (possibly infinite) graphs by identifying the input nodes of a size-change graph G_i with the output nodes of the next size-change graph G_{i+1}. Then a program is called *size-change terminating* iff there exists an infinite path in this graph which contains infinitely many edges labelled with "$\succ$". The following lemma (whose proof uses the same ideas as the proof of [24, Theorem 4]) states that our definition is equivalent to the one of [24]. Here, for two size-change graphs or multigraphs G and H where G's input nodes have the same labels as H's output nodes, let $G \circ H$ be the graph resulting from identifying G's input and H's output nodes. So $G \circ H$ differs from $G \bullet H$ in that these nodes are not dropped.

Lemma 6 (Infinite Graphs Correspond to Multigraphs) *Let Γ be a finite set of size-change graphs. The following statements are equivalent.*

(1) Every graph $G_1 \circ G_2 \circ \ldots$ with an infinite sequence $G_1, G_2, \ldots \in \Gamma$ has an infinite path containing infinitely many edges labelled with "$\succ$".

[1] Our definition of *size-change termination* generalizes the original one of [24] by permitting arbitrary reduction pairs $(\succsim, \succ)$. If one is restricted to the reduction pairs used in [24], then size-change termination implies termination for functional programs. However, if one may use arbitrary reduction pairs, then size-change termination is no longer sufficient for termination. Therefore, in Section 3 we identify classes of reduction pairs where size-change termination indeed implies (innermost) termination of TRSs.

(2) *Every maximal multigraph* $G_1 \bullet G_2 \bullet \ldots \bullet G_n$ *with* $G_1, \ldots, G_n \in \Gamma$ *has an edge of the form* $i \xrightarrow{\succsim} i$.

Proof We first prove "(1) $\Rightarrow$ (2)". Assume that there exists a maximal multigraph $G = G_1 \bullet \ldots \bullet G_n$ which has no edge of the form $i \xrightarrow{\succsim} i$. On the other hand, the graph $G_1 \circ \ldots \circ G_n \circ G_1 \circ \ldots \circ G_n \circ \ldots$ must have a path containing infinitely many edges labelled with "$\succ$". Thus, this also holds for the infinite graph $G \circ G \circ \ldots$ Obviously, for some $i \in \mathbb{N}$ and $f \in \mathcal{F}$, a node labelled with i_f must occur more than once in this path such that an edge between these two occurrences is labelled with "$\succ$". Let k be the length of the subpath from the first occurrence of i_f to the next occurrence of i_f such that an $\xrightarrow{\succ}$-edge is on this subpath. Thus, there is a path from i_f to i_f in the graph $G \circ G \circ \ldots \circ G$ (where G is combined with itself k times) and at least one edge of the path is labelled with "$\succ$". This means that the multigraph $G \bullet G \bullet \ldots \bullet G$ (where G is concatenated with itself k times), contains an edge $i \xrightarrow{\succ} i$. Since G is idempotent, we have $G \bullet G \bullet \ldots \bullet G = G$ and thus, this contradicts the assumption that G does not have such edges.

Now we show "(2) $\Rightarrow$ (1)". Assume that there is an infinite graph $G_1 \circ G_2 \circ \ldots$ that does not contain an infinite path with infinitely many $\xrightarrow{\succ}$-edges. For all pairs of numbers (n, m) with $n < m$ let $G_{n,m}$ be the multigraph resulting from the concatenation of $G_n, \ldots, G_{m-1}$, i.e., $G_{n,m} = G_n \bullet \ldots \bullet G_{m-1}$. As there are only finitely many possible multigraphs, by Ramsey's theorem there is an infinite set $I \subseteq \mathbb{N}$ such that $G_{n,m}$ is always the same graph for all $n, m \in I$ with $n < m$. We call this graph G. Note that G is a maximal multigraph: for $n_1 < n_2 < n_3$ with $n_i \in I$, we have $G_{n_1,n_3} = G_{n_1} \bullet \ldots \bullet G_{n_2-1} \bullet G_{n_2} \bullet \ldots \bullet G_{n_3-1} = G_{n_1,n_2} \bullet G_{n_2,n_3}$, and thus $G = G \bullet G$.

Let $I = \{n_1, n_2, \ldots\}$ with $n_1 < n_2 < \ldots$ Thus, for our original infinite graph, we have

$$G_1 \circ G_2 \circ \ldots \;=\; G_1 \circ \ldots \circ G_{n_1-1} \circ G_{n_1} \circ \ldots \circ G_{n_2-1} \circ G_{n_2} \circ \ldots \circ G_{n_3-1} \circ \ldots$$

Since by assumption, this graph did not contain an infinite path with infinitely many $\xrightarrow{\succ}$-edges, this also holds for the graph

$$G_{n_1} \bullet \ldots \bullet G_{n_2-1} \circ G_{n_2} \bullet \ldots \bullet G_{n_3-1} \circ \ldots \;=\; G_{n_1,n_2} \circ G_{n_2,n_3} \circ \ldots \;=\; G \circ G \circ \ldots$$

But since G is a maximal multigraph, G contains an edge $i \xrightarrow{\succ} i$. Thus, the above infinite graph does contain an infinite path labelled with infinitely many $\xrightarrow{\succ}$-edges, which contradicts the assumption. $\square$

3 Size-Change Termination and Termination of TRSs

In [24], the authors use reduction pairs $(\succsim, \succ)$ where $\succsim$ and $\succ$ are relations on constructor terms[2] and where $\succsim$ is the reflexive closure of $\succ$. Then indeed,

[2] More precisely, they use an underlying well-founded order $>$ on "values" (i.e., constructor ground terms) and do not focus on the problem of comparing terms with defined symbols.

size-change termination implies termination of the corresponding functional program.

However in general, one also has to compare terms containing defined symbols when building size-change graphs. In particular, when regarding TRSs instead of functional programs, defined symbols may occur both in the input arguments and in the recursive arguments (i.e., in a rule $f(s_1, \ldots, s_n) \to \ldots f(t_1, \ldots, t_n) \ldots$, any terms s_i or t_i may contain defined symbols).

Therefore, in this section we investigate which reduction pairs may be used in order to apply the size-change principle to TRSs. Unfortunately, in general size-change termination does not imply termination if one may use arbitrary reduction pairs in Definition 5.

Example 7 Consider the TRS with the rules $\mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b})$ and $\mathsf{b} \to \mathsf{a}$. If we use the lexicographic path order $\succ_{LPO}$ [17] with the precedence $\mathsf{a} > \mathsf{b}$, then the only maximal multigraph is $1_\mathsf{f} \xrightarrow{\succ_{LPO}} 1_\mathsf{f}$. So size-change termination can be proved, although the TRS is obviously not terminating.

In this section we develop conditions on the reduction pair in Definition 5 which ensure that size-change termination indeed implies (innermost) termination. Then the size-change principle can be combined with classical orders from term rewriting and becomes a sound (innermost) termination criterion for TRSs.

Innermost termination is interesting, since then there are no infinite reductions w.r.t. eager evaluation strategies. Moreover, for non-overlapping TRSs, innermost termination already implies termination. Example 7 demonstrates that size-change termination w.r.t. an arbitrary reduction pair does not imply innermost termination. Therefore, to obtain a sufficient criterion for innermost termination, we will only use the *restriction* of the reduction pair to the constructors $\mathcal{C}$ when building size-change graphs.

Definition 8 ($\mathcal{G}$-restriction) *For a subset $\mathcal{G} \subseteq \mathcal{F}$ and a relation $\succ$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$, its $\mathcal{G}$-*restriction *$\succ'$ is defined as $s \succ' t$ iff $s \succ t$ and $t \in \mathcal{T}(\mathcal{G}, \mathcal{V})$. For a reduction pair $(\succsim, \succ)$ its $\mathcal{G}$-restriction $(\succsim', \succ')$ consists of the $\mathcal{G}$-restrictions of $\succsim$ and $\succ$, respectively.*

Strictly speaking, the $\mathcal{G}$-restriction $(\succsim', \succ')$ is not a reduction pair since it is only closed under substitutions with terms from $\mathcal{T}(\mathcal{G}, \mathcal{V})$. Nevertheless, the definitions of size-change graphs and size-change termination can of course be extended to any pairs $(\succsim', \succ')$ of relations in a straightforward way. This leads to the following theorem which shows how to use the size-change principle for innermost termination proofs of TRSs.

Theorem 9 (Innermost Termination Proofs) *Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$ with constructors $\mathcal{C}$ and let $(\succsim, \succ)$ be a reduction pair on $\mathcal{T}(\mathcal{F}, \mathcal{V})$. If $\mathcal{R}$ is size-change terminating w.r.t. the $\mathcal{C}$-restriction of $(\succsim, \succ)$, then $\mathcal{R}$ is innermost terminating.*

Proof If $\mathcal{R}$ is not innermost terminating, then there is a minimal non-innermost terminating term v_0, i.e., all proper subterms of v_0 are innermost terminating. Let $\xrightarrow{\mathsf{i}}_\varepsilon$ denote root reductions and let $\xrightarrow{\mathsf{i}}_{>\varepsilon}$ denote reductions below the root. Then v_0's infinite innermost reduction starts with $v_0 \xrightarrow{\mathsf{i}}{}^*_{>\varepsilon} u_1 \xrightarrow{\mathsf{i}}_\varepsilon w_1$ where all proper subterms of u_1 are in normal form. Since w_1 is not innermost terminating, it has a minimal non-innermost terminating subterm v_1.

The infinite reduction continues in the same way. So for $i \geq 1$, we have $v_{i-1} \xrightarrow{\mathsf{i}}{}^*_{>\varepsilon} u_i = l_i\sigma$ and $v_i = r'_i\sigma$ for a rule $l_i \to r_i$, a subterm r'_i of r_i with defined root, and a substitution σ which instantiates l_i's variables with normal forms. To ease readability we assume that different (occurrences of) rules $l_i \to r_i$ are variable disjoint. Then we can use the same substitution σ for all $i \geq 1$.

For each step from u_i to v_i there is a corresponding size-change graph G_i. If $\mathcal{R}$ is size-change terminating, by Lemma 6 the graph $G_1 \circ G_2 \circ \ldots$ contains an infinite path where infinitely many edges are labelled with "$\succ$". Without loss of generality we assume that this path already starts in G_1. For every i, let a_i be the output node in G_i which is on this path. So we have $l_i|_{a_i} \succ r'_i|_{a_{i+1}}$ for all i from an infinite set $I \subseteq \mathbb{N}$ and $l_i|_{a_i} \succsim r'_i|_{a_{i+1}}$ for $i \in \mathbb{N} \setminus I$. As usual, "$l_i|_{a_i}$" denotes the subterm of l_i at the position a_i. Note that $l_i|_{a_i}\sigma = u_i|_{a_i}$ and $r'_i|_{a_{i+1}}\sigma = v_i|_{a_{i+1}} \xrightarrow{\mathsf{i}}{}^* u_{i+1}|_{a_{i+1}}$. Recall that $r'_i|_{a_{i+1}} \in \mathcal{T}(\mathcal{C},\mathcal{V})$, as one only uses the $\mathcal{C}$-restriction for the construction of size-change graphs. Since σ instantiates l_i's variables with normal forms, since $\mathcal{V}(r'_i) \subseteq \mathcal{V}(l_i)$, and since $r'_i|_{a_{i+1}}$ is a constructor term, we conclude that $r'_i|_{a_{i+1}}\sigma$ is a normal form. Hence, in the above reduction we can replace "$\xrightarrow{\mathsf{i}}{}^*$" by "$=$" and obtain $r'_i|_{a_{i+1}}\sigma = v_i|_{a_{i+1}} = u_{i+1}|_{a_{i+1}}$.

Hence, we have $u_i|_{a_i} \succ u_{i+1}|_{a_{i+1}}$ for all $i \in I$ and $u_i|_{a_i} \succsim u_{i+1}|_{a_{i+1}}$ for all $i \in \mathbb{N} \setminus I$. This is a contradiction to the well-foundedness of $\succ$. $\square$

For the TRS in Example 2, when using the $\mathcal{C}$-restriction of the reduction pair $(\to^*_{Emb_\mathcal{C}}, \to^+_{Emb_\mathcal{C}})$, we obtain the size-change graphs $G_{(1)}$ and $G_{(2)}$. Example 4 shows that the TRS is size-change terminating w.r.t. this reduction pair and hence, by Theorem 9, this proves innermost termination. However, a variant of Toyama's example [32] shows that Theorem 9 is not sufficient to prove full (non-innermost) termination.

Example 10 Let $\mathcal{R} = \{\mathsf{f}(\mathsf{c}(\mathsf{a},\mathsf{b},x)) \to \mathsf{f}(\mathsf{c}(x,x,x)),\ \mathsf{g}(x,y) \to x,\ \mathsf{g}(x,y) \to y\}$. We define $\succsim\, = \,\to^*_\mathcal{S}$ and $\succ\, =\, \to^+_\mathcal{S}$, where $\mathcal{S}$ is the terminating TRS with the rule $\mathsf{c}(\mathsf{a},\mathsf{b},x) \to \mathsf{c}(x,x,x)$. The only size-change graph is $1_\mathsf{f} \xrightarrow{\succ} 1_\mathsf{f}$ (since the input argument $\mathsf{c}(\mathsf{a},\mathsf{b},x)$ is greater than the argument $\mathsf{c}(x,x,x)$ in the recursive argument when compared w.r.t. the $\mathcal{C}$-restriction of $\to^+_\mathcal{S}$). Moreover, concatenating this size-change graph with itself yields $1_\mathsf{f} \xrightarrow{\succ} 1_\mathsf{f}$ again, i.e., this is the only maximal multigraph. Thus, $\mathcal{R}$ is size-change terminating and by Theorem 9 it is innermost terminating. However, $\mathcal{R}$ does not terminate as can be seen by the following cyclic reduction: $\mathsf{f}(\mathsf{c}(\mathsf{a},\mathsf{b},\mathsf{g}(\mathsf{a},\mathsf{b}))) \to \mathsf{f}(\mathsf{c}(\mathsf{g}(\mathsf{a},\mathsf{b}),\mathsf{g}(\mathsf{a},\mathsf{b}),\mathsf{g}(\mathsf{a},\mathsf{b}))) \to^* \mathsf{f}(\mathsf{c}(\mathsf{a},\mathsf{b},\mathsf{g}(\mathsf{a},\mathsf{b})))$.

As in Example 10, reduction pairs $(\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$ can be defined using a terminating TRS $\mathcal{S}$. The following theorem shows that if $\mathcal{S}$ is a non-duplicating TRS over $\mathcal{C}$, then we may use the relation $\to_{\mathcal{S}}$ also on terms with defined symbols and size-change termination even implies full termination. A TRS is *non-duplicating* iff every variable occurs on the left-hand side of a rule at least as often as on the corresponding right-hand side. So size-change termination of the TRS in Examples 2 and 4 using the reduction pair $(\to_{Emb_{\mathcal{C}}}^*, \to_{Emb_{\mathcal{C}}}^+)$ implies that the TRS is indeed terminating.

In order to prove the theorem, we need a preliminary lemma which states that minimal non-terminating terms w.r.t. $\mathcal{R} \cup \mathcal{S}$ cannot start with constructors of $\mathcal{R}$. Again, here $\mathcal{S}$ must be non-duplicating. Otherwise, Example 10 would be a counterexample, since $\mathsf{c}(\mathsf{a}, \mathsf{b}, \mathsf{g}(\mathsf{a}, \mathsf{b}))$ is a minimal non-terminating term w.r.t. $\mathcal{R} \cup \mathcal{S}$ that starts with a constructor of $\mathcal{R}$.

Lemma 11 *Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$ with constructors $\mathcal{C}$ and let $\mathcal{S}$ be a terminating non-duplicating TRS over the signature $\mathcal{C}$.*

If $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ are terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$ and $c \in \mathcal{C}$, then the term $c(t_1, \ldots, t_n)$ is also terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$.

Proof For any term $s \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, let M_s be the multiset of the maximal subterms of s whose root is defined, i.e., $M_s = \{s|_p \mid root(s|_p) \in \mathcal{D}$ and for all p' above p we have $root(s|_{p'}) \in \mathcal{C}\}$. Let $\twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}}$ be the extension of $(\to_{\mathcal{R}\cup\mathcal{S}} \cup \rhd)$ to multisets where $M \twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}} M'$ iff $M = N \cup \{s\}$ and $M' = N \cup \{t_1, \ldots, t_n\}$ with $n \geq 0$ and with $s\,(\to_{\mathcal{R}\cup\mathcal{S}} \cup \rhd)\,t_i$ for all i. We prove the following conjecture.

> Let $s \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that all terms in M_s are terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$ and let $s \to_{\mathcal{R}\cup\mathcal{S}} t$. Then all terms in M_t are also terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$. Moreover, $M_s \twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}}^+ M_t$ or both $M_s \supseteq M_t$ and $s \to_{\mathcal{S}} t$. (3)

Note that if a term is terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$, then the term does not start any infinite decreasing sequence w.r.t. $\to_{\mathcal{R}\cup\mathcal{S}} \cup \rhd$ either. Hence, $\twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}}^+$ is well founded on multisets like M_s which only contain terminating terms. Termination of $\mathcal{S}$ implies that $\to_{\mathcal{S}}$ is also well founded and the lexicographic combination of two well-founded orders preserves well-foundedness. Hence, (3) implies that if all terms in M_s are terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$, then s is terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$ as well. So the lemma immediately follows from Conjecture (3).

To prove (3), we perform induction on s. If s has a defined root symbol then we have $M_s = \{s\} \twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}} \{t\} \twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}}^* M_t$, where in the step from $\{t\}$ to M_t, t is replaced by its maximal subterms with defined root. Otherwise, we have $s = c(s_1, \ldots, s_n) \to_{\mathcal{R}\cup\mathcal{S}} t$ where c is a constructor. We distinguish two cases: If the reduction is on the root position we must have used a rule of $\mathcal{S}$ and get $s \to_{\mathcal{S}} t$ and $M_s \supseteq M_t$ as $\mathcal{S}$ is non-duplicating. If the reduction is below the root then there must be some s_i such that $s_i \to_{\mathcal{R}\cup\mathcal{S}} t_i$ and $t = c(s_1, \ldots, t_i, \ldots, s_n)$. By induction we conclude that $M_{s_i} \twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}}^+ M_{t_i}$ or both $M_{s_i} \supseteq M_{t_i}$ and $s_i \to_{\mathcal{S}} t_i$. As c is a constructor we know that

$M_s = M_{s_1} \cup \dots \cup M_{s_i} \cup \dots \cup M_{s_n}$ and $M_t = M_{s_1} \cup \dots \cup M_{t_i} \cup \dots M_{s_n}$. In either case we easily obtain $M_s \twoheadrightarrow^{+}_{\mathcal{R}\cup\mathcal{S}} M_t$ or both $M_s \supseteq M_t$ and $s \to_{\mathcal{S}} t$. □

Now we can show the desired theorem.

Theorem 12 (Termination Proofs) *Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$ with constructors $\mathcal{C}$ and let $\mathcal{S}$ be a terminating non-duplicating TRS over $\mathcal{C}$. If $\mathcal{R}$ is size-change terminating w.r.t. the reduction pair $(\to^*_{\mathcal{S}}, \to^+_{\mathcal{S}})$ on $\mathcal{T}(\mathcal{F},\mathcal{V})$, then $\mathcal{R}$ (and even $\mathcal{R}\cup\mathcal{S}$) is terminating.*

Proof We define $\mathcal{R}' := \mathcal{R} \cup \mathcal{S}$. If $\mathcal{R}'$ is not terminating, then as in the proof of Theorem 9 we obtain an infinite sequence of minimal non-terminating terms u_i, v_i with $v_i \to^*_{>\varepsilon,\mathcal{R}'} u_{i+1}$ where the step from u_i to v_i corresponds to a size-change graph of $\mathcal{R}'$. Thus, for all i there is a rule $l_i \to r_i$ in $\mathcal{R}'$ with $u_i = l_i\sigma$ and $v_i = r'_i\sigma$ for a subterm r'_i of r_i and a substitution σ. The reason for $v_i = r'_i\sigma$ is that by the minimality of $u_i = l_i\sigma$, σ instantiates all variables of l_i (and hence, of r_i) by terminating terms. Hence, any non-terminating subterm of $r_i\sigma$ has the form $r'_i\sigma$ for a non-variable subterm r'_i of r_i.

By Lemma 11, the roots of u_i and v_i are defined symbols of $\mathcal{R}$. So all these size-change graphs are from $\mathcal{R}$. As in Theorem 9's proof, there are a_i with $l_i|_{a_i} \to^+_{\mathcal{S}} r'_i|_{a_{i+1}}$ for all i from an infinite set $I \subseteq \mathbb{N}$ and $l_i|_{a_i} \to^*_{\mathcal{S}} r'_i|_{a_{i+1}}$ for $i \in \mathbb{N} \setminus I$ with $i \geq 1$. Since $\to_{\mathcal{S}}$ is closed under substitution we also have $u_i|_{a_i} \to^+_{\mathcal{S}} v_i|_{a_{i+1}}$ or $u_i|_{a_i} \to^*_{\mathcal{S}} v_i|_{a_{i+1}}$, respectively. Recall $v_i|_{a_{i+1}} \to^*_{\mathcal{R}'} u_{i+1}|_{a_{i+1}}$ and $\mathcal{S} \subseteq \mathcal{R}'$. So for $I = \{i_1, i_2, \dots\}$ with $i_1 < i_2 < \dots$ we have $u_{i_1}|_{a_{i_1}} \to^+_{\mathcal{R}'} u_{i_2}|_{a_{i_2}} \to^+_{\mathcal{R}'} \dots$ contradicting the minimality of the terms u_i. □

With Theorems 9 and 12 we have two possibilities for automating the size-change principle. Note that even for innermost termination, Theorem 9 and Theorem 12 do not subsume each other. Innermost termination of Example 10 cannot be shown by Theorem 12 and innermost termination of $\{\mathsf{g}(\mathsf{f}(\mathsf{a})) \to \mathsf{g}(\mathsf{f}(\mathsf{b})), \mathsf{f}(x) \to x\}$ cannot be proved with Theorem 9, since $\mathsf{f}(\mathsf{a}) \not\succ' \mathsf{f}(\mathsf{b})$ for the $\mathcal{C}$-restriction $\succ'$ of any order $\succ$. On the other hand, termination is easily shown with Theorem 12 using $\mathcal{S} = \{\mathsf{a} \to \mathsf{b}\}$. In fact, a variant of Theorem 12 also holds for innermost termination if $\mathcal{S}$ is innermost terminating (and possibly duplicating). However, this variant only proves innermost termination of $\mathcal{R} \cup \mathcal{S}$ and in general, this does not imply innermost termination of $\mathcal{R}$.

So Theorems 9 and 12 are new contributions that show which reduction pairs are admissible in order to use size-change termination for termination or innermost termination proofs of TRSs. In this way, size-change termination can be turned into an automatic technique, since one can use classical techniques from termination analysis of term rewriting to generate suitable reduction pairs automatically.

4 Comparison with Orders from Term Rewriting

Most traditional techniques for termination of TRSs are based on *simplification orders*. A TRS is *simply terminating* iff there is a simplification order $\succ$ such that $l \succ r$ holds for all rules $l \to r$ of the TRS. Equivalently, a TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is simply terminating iff $\mathcal{R} \cup \mathit{Emb}_\mathcal{F}$ terminates. We now show that similar to these traditional techniques, the size-change principle only verifies simple termination.

Theorem 13 (a) states that the size-change principle cannot distinguish between the termination behavior of $\mathcal{R}$ and of $\mathcal{R} \cup \mathit{Emb}_\mathcal{F}$. For this reason, the size-change principle is not suitable for non-simply terminating TRSs (where $\mathcal{R}$ terminates and $\mathcal{R} \cup \mathit{Emb}_\mathcal{F}$ does not). More precisely by Theorem 13 (b), the size-change principle for termination of TRSs from Theorem 12 can only prove simple termination if the underlying base order (i.e., the relation $\to_\mathcal{S}^+$) is a simplification order. In other words, the size-change principle does not succeed when using simplification orders for termination proofs of non-simply terminating systems.

Theorem 13 (Size-Change Principle and Simple Termination)

(a) A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is size-change terminating w.r.t. a reduction pair $(\succsim, \succ)$ iff $\mathcal{R} \cup \mathit{Emb}_\mathcal{F}$ is size-change terminating w.r.t. $(\succsim, \succ)$.

(b) Let $\mathcal{S}$ be a non-duplicating TRS as in Theorem 12. If $\mathcal{S}$ is simply terminating and $\mathcal{R}$ is size-change terminating w.r.t. $(\to_\mathcal{S}^*, \to_\mathcal{S}^+)$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$, then $\mathcal{R} \cup \mathcal{S}$ is simply terminating.

Proof

(a) The “if” direction is obvious. For the “only if” direction, note that $\mathit{Emb}_\mathcal{F}$ yields no new size-change graphs. But due to $\mathit{Emb}_\mathcal{C}$, all constructors are transformed into defined symbols. So from those $\mathcal{R}$-rules with (former) constructors in their right-hand side, we obtain additional size-change graphs whose input nodes are labelled with constructors (i.e., $1_c, \ldots, n_c$ for $c \in \mathcal{C}$). However, since output nodes are never labelled with constructors, this does not yield new maximal multigraphs (since there, output and input nodes must be labelled by the same function). Hence, size-change termination is not affected when adding $\mathit{Emb}_\mathcal{F}$.

(b) As in (a), adding $\mathit{Emb}_\mathcal{D}$ to $\mathcal{R}$ yields no new size-change graphs and thus, $\mathcal{R} \cup \mathit{Emb}_\mathcal{D}$ is also size-change terminating w.r.t. $(\to_\mathcal{S}^*, \to_\mathcal{S}^+)$ and hence, also w.r.t. $(\to_{\mathcal{S} \cup \mathit{Emb}_\mathcal{C}}^*, \to_{\mathcal{S} \cup \mathit{Emb}_\mathcal{C}}^+)$. Note that this is indeed a reduction pair, since $\mathcal{S} \cup \mathit{Emb}_\mathcal{C}$ is terminating by simple termination of $\mathcal{S}$. Now Theorem 12 implies termination of $\mathcal{R} \cup \mathit{Emb}_\mathcal{D} \cup \mathcal{S} \cup \mathit{Emb}_\mathcal{C}$, i.e., simple termination of $\mathcal{R} \cup \mathcal{S}$. $\square$

The restriction to simple termination excludes many practically relevant TRSs. Theorem 13 illustrates that the size-change principle cannot compete with new techniques from term rewriting (e.g., *dependency pairs* [2] or the *monotonic semantic path order* [5]) which can prove termination of non-simply terminating TRSs using a simplification order as underlying base order. However, these new techniques still require methods to generate such base orders. Hence, there is still an urgent need for powerful simplification orders.

In the remainder of the section, we clarify the connection between size-change termination and classical orders and show in Sections 4.1 and 4.2 that size-change termination and classical simplification orders do not subsume each other in general.

4.1 Advantages of the Size-Change Principle

A major advantage of the size-change principle is that it can simulate the basic ingredients of *LPO* and *RPO*(S), i.e., the concepts of *lexicographic* and of *multiset* comparison. Thus, by the size-change principle w.r.t. a very simple reduction pair like the embedding order we obtain an automated method for termination analysis which avoids the search problems of *LPO* and *RPO*(S) and which can still capture the idea of comparing tuples of arguments lexicographically or as multisets. To simplify the presentation, here we restrict ourselves to TRSs without mutual recursion. Thus, we only regard TRSs where there exist no function symbols $f \neq g$ such that f depends on g and g depends on f. Here, every function symbol depends on itself and moreover, a function symbol f depends on g if some symbol h which depends on g occurs in the right-hand side of an f-rule. If there is no mutual recursion, then for size-change termination it is sufficient only to build size-change graphs which compare the left-hand side of a rule $f(s_1, \ldots, s_n) \to r$ with those subterms of r whose root is f.

We first show in Theorem 14 that lexicographic orders can be simulated by the size-change principle. For a reduction pair $(\succsim, \succ)$ and a permutation π of $1, \ldots, n$, let $\succ_{lex}^{\pi}$ be the following order on n-tuples: $(s_1, \ldots, s_n) \succ_{lex}^{\pi} (t_1, \ldots, t_n)$ iff there is an $1 \leq i \leq n$ such that $s_{\pi(i)} \succ t_{\pi(i)}$ and $s_{\pi(j)} \succsim t_{\pi(j)}$ for all $j < i$.

Theorem 14 (Simulating Lexicographic Comparison) *Let $(\succsim, \succ)$ be a reduction pair, let π be a permutation of $1, \ldots, n$. Moreover, let $\mathcal{R}$ be a TRS without mutual recursion such that for every rule $f(s_1, \ldots, s_n) \to r$ and every subterm $f(t_1, \ldots, t_n)$ of r, we have $(s_1, \ldots, s_n) \succ_{lex}^{\pi} (t_1, \ldots, t_n)$. Then $\mathcal{R}$ is size-change terminating w.r.t. $(\succsim, \succ)$.*

Proof All size-change graphs for recursive calls of f have an edge $\pi(i)_f \xrightarrow{\succ} \pi(i)_f$ for some i and $\pi(j)_f \xrightarrow{\succsim} \pi(j)_f$ for all $j < i$. The concatenation of such graphs again yields a graph of this form and thus, all maximal multigraphs are also of this shape. Hence, they all contain an edge of the form $\pi(i)_f \xrightarrow{\succ} \pi(i)_f$ which proves size-change termination. □

Thus, size-change termination w.r.t. the same reduction pair $(\succsim, \succ)$ can simulate $\succ_{lex}^{\pi}$ for *any permutation* π used to compare the components of a tuple.

Example 15 For instance, regard the TRS $\{\mathsf{ack}(0, y) \to \mathsf{s}(y),\ \mathsf{ack}(\mathsf{s}(x), 0) \to \mathsf{ack}(x, \mathsf{s}(0)),\ \mathsf{ack}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{ack}(x, \mathsf{ack}(\mathsf{s}(x), y))\}$ computing the Ackermann function. By Theorem 14, the TRS is size-change terminating w.r.t. the embedding order on constructors, whereas with traditional term rewriting techniques, one would have to use the lexicographic path order.

The next theorem shows that size-change termination can also simulate multiset comparison. For a reduction pair $(\succsim, \succ)$, let $(s_1, \dots, s_n) \succ_{mul} (t_1, \dots, t_n)$ hold iff $\{s_1, \dots, s_n\} \neq \{t_1, \dots, t_n\}$ and for each $t \in \{t_1, \dots, t_n\} \setminus \{s_1, \dots, s_n\}$ there is an $s \in \{s_1, \dots, s_n\} \setminus \{t_1, \dots, t_n\}$ with $s \succ t$. Here, $\{s_1, \dots, s_n\}$ and $\{t_1, \dots, t_n\}$ denote multisets. So $\succ_{mul}$ compares tuples $(s_1, \dots, s_n)$ and $(t_1, \dots, t_n)$ by replacing at least one s_i by zero or more components t_j that are $\succ$-smaller than s_i.

Theorem 16 (Simulating Multiset Comparison) *Let $(\succsim, \succ)$ be a reduction pair and let $\mathcal{R}$ be a TRS without mutual recursion such that for every rule $f(s_1, \dots, s_n) \to r$ and every subterm $f(t_1, \dots, t_n)$ of r, we have $(s_1, \dots, s_n) \succ_{mul} (t_1, \dots, t_n)$. Then $\mathcal{R}$ is size-change terminating w.r.t. $(\succsim, \succ)$.*

Proof In every size-change graph for recursive calls of f, one can select a subset of edges with the following properties: (1) all input nodes have exactly one selected incoming edge, (2) for each output node, if one selects an outgoing edge labelled with "$\succsim$", then no other edge starting in this node may be selected, (3) at least one edge labelled with "$\succ$" is selected. The reason is that for all $t_i \in \{t_1, \dots, t_n\} \setminus \{s_1, \dots, s_n\}$, there is an $\xrightarrow{\succ}$-edge ending in i_f and all other input nodes are reached by an $\xrightarrow{\succsim}$-edge.

It is easy to see that if one concatenates such size-change graphs G_1 and G_2 and selects those edges which result from the concatenation of two selected edges in G_1 and G_2, then the selected edges in the resulting multigraph also satisfy the conditions (1) – (3). Hence, the properties (1) – (3) also hold for the maximal multigraphs. Due to (3), there exists a selected edge $i_f \xrightarrow{\succ} j_f$ in each maximal multigraph. By (1), there is also a selected edge $k_f \to i_f$ reaching the input node marked with i_f. In the concatenation of the multigraph with itself, $k_f \to i_f \xrightarrow{\succ} j_f$ would give rise to a (selected) edge $k_f \xrightarrow{\succ} j_f$. Since maximal multigraphs are idempotent, the multigraph itself must already contain the (selected) edge $k_f \xrightarrow{\succ} j_f$. Then (1) implies that $k_f = i_f$ and hence, we have a selected edge $k_f = i_f \to i_f$. Due to (2), this edge must be labelled with "$\succ$" and thus, size-change termination is proved. $\square$

The construction in the above proof is illustrated in Figure 1, where we only depicted the *selected* edges of the graphs. Thus, every input node is reached by one unique edge (1) and every output node may have at most one outgoing $\xrightarrow{\succsim}$-edge (2). Moreover, there must be at least one $\xrightarrow{\succ}$-edge in each graph (3). The example in Figure 1 demonstrates that the properties (1) – (3) are indeed preserved under concatenation of graphs.

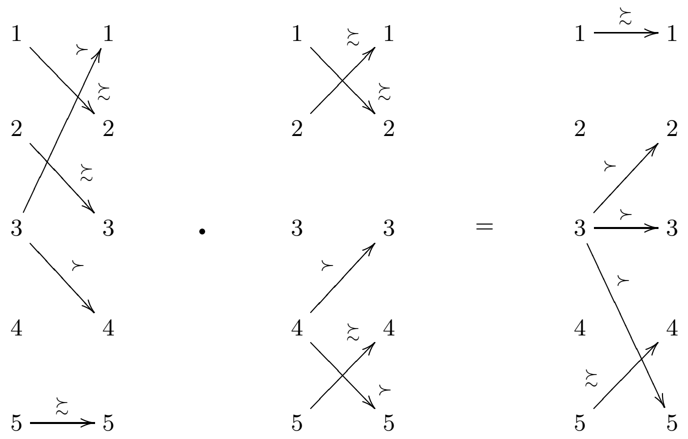


Fig. 1 Multiset Comparison with Size-Change Graphs

For example by Theorem 16, the TRS $\{\mathsf{plus}(\mathsf{0}, y) \to y,\ \mathsf{plus}(\mathsf{s}(x), y) \to \mathsf{s}(\mathsf{plus}(y, x))\}$ where plus permutes its arguments is size-change terminating w.r.t. the embedding order on constructors, whereas in existing rewriting approaches one would have to use the recursive (multiset) path order.

Since both lexicographic and multiset comparison are simulated by the size-change principle using the *same* reduction pair, in this way one can also deal with TRSs where traditional orders like *RPOS*, *KBO*, or polynomial orders fail.

Example 17 We extend the TRS of Example 2 by the rules for the Ackermann function from Example 15.

$$\begin{array}{ll}
\mathsf{f}(\mathsf{s}(x), y) \to \mathsf{f}(x, \mathsf{s}(x)) & \mathsf{ack}(\mathsf{0}, y) \to \mathsf{s}(y) \\
\mathsf{f}(x, \mathsf{s}(y)) \to \mathsf{f}(y, x) & \mathsf{ack}(\mathsf{s}(x), \mathsf{0}) \to \mathsf{ack}(x, \mathsf{s}(\mathsf{0})) \\
 & \mathsf{ack}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{ack}(x, \mathsf{ack}(\mathsf{s}(x), y))
\end{array}$$

Classical path orders like *RPOS* (or *KBO*) cannot prove termination of the f-rules. The reason is that in the first rule $\mathsf{f}(\mathsf{s}(x), y) \to \mathsf{f}(x, \mathsf{s}(x))$ the arguments of f have to be compared lexicographically from left to right and in the second rule $\mathsf{f}(x, \mathsf{s}(y)) \to \mathsf{f}(y, x)$ they have to be compared as multisets. Moreover, due to the rules for the Ackermann function, polynomial orders fail as well. In contrast, the TRS is size-change terminating w.r.t. the embedding order on constructors.

4.2 Disadvantages of the Size-Change Principle

However, compared to classical orders from term rewriting, the size-change principle also has several drawbacks. One problem is that it can only simulate lexicographic and multiset comparison for the arguments of the *root* symbol (provided we again use a simple reduction pair like the embedding order which itself does not feature lexicographic or multiset comparison). Hence, if one adds a new function on top of all terms in the rules, this simulation is no longer possible. For example, the TRS $\{\mathsf{f}(\mathsf{plus}(\mathsf{0}, y)) \to \mathsf{f}(y), \mathsf{f}(\mathsf{plus}(\mathsf{s}(x), y)) \to \mathsf{f}(\mathsf{s}(\mathsf{plus}(y, x)))\}$ is no longer size-change terminating w.r.t. the embedding order, whereas classical path orders can apply lexicographic or multiset comparisons on all levels of the term. Thus, termination would still be easy to prove with RPO.

Perhaps the most serious drawback is that the size-change principle lacks concepts to compare defined function symbols syntactically. For example, consider a TRS with the rule $\mathsf{log}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{log}(\mathsf{s}(\mathsf{half}(x))))$ and rules for half such that $\mathsf{half}(x)$ computes $\lfloor \frac{x}{2} \rfloor$. Whenever a function (like log) calls another defined function (like half) in the arguments of its recursive calls, one has to check whether the argument $\mathsf{half}(x)$ is smaller than the term $\mathsf{s}(x)$ in the corresponding left-hand side. The size-change principle on its own offers no possibility for that and its mechanizable versions (Theorems 9 and 12) fail since they only compare terms w.r.t. the $\mathcal{C}$-restriction of an order or w.r.t. an order given by a TRS over $\mathcal{C}$. In contrast, classical orders like RPO can easily show termination automatically using a *precedence* $\mathsf{log} > \mathsf{s} > \mathsf{half}$ on function symbols.

Finally, the size-change principle has the disadvantage that it cannot *measure* terms by combining the measures of subterms as in polynomial orders or KBO.

Example 18 Term measures (or *weights*) are particularly useful if one parameter is increasing, but the decrease of another parameter is greater than this increase. So termination of the TRS $\{\mathsf{plus}(\mathsf{s}(\mathsf{s}(x)), y) \to \mathsf{s}(\mathsf{plus}(x, \mathsf{s}(y))), \mathsf{plus}(x, \mathsf{s}(\mathsf{s}(y))) \to \mathsf{s}(\mathsf{plus}(\mathsf{s}(x), y)), \mathsf{plus}(\mathsf{s}(\mathsf{0}), y) \to \mathsf{s}(y),\ \mathsf{plus}(\mathsf{0}, y) \to y\}$ is trivial to prove with polynomial orders or KBO, but the TRS is not size-change terminating w.r.t. *any* reduction pair.

This drawback of being unable to measure terms is partly solved in a recent improvement of the size-change principle [1]. Up to now, size-change graphs and multigraphs state whether a parameter is strictly or weakly decreasing, but they cannot express how big this decrease is. In the new *affine-based size-change principle (AB-SCP)* of [1], size-change graphs and multigraphs are annotated with Presburger constraints which give more detailed information on the amount of the decrease. To illustrate this, consider the TRS $\{\mathsf{f}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{g}(x),\ \mathsf{g}(x) \to \mathsf{f}(\mathsf{s}(x))\}$. The size-change principle cannot capture that the big decrease of the parameter in the first rule (from f to g) compensates the small increase of the parameter in the second rule (from g to f). In contrast, termination can be proved easily with the AB-SCP. However, the AB-SCP still fails for Example 18: one obtains a maximal multigraph with Presburger constraints which state that the sum of the two parameters of plus is strictly decreasing. But since no single parameter is guaranteed to decrease, the graph contains no edges. (Both in the ordinary size-change principle and in the AB-SCP, the output and input nodes of multigraphs correspond to the parameters of the functions and an edge from one parameter to another parameter indicates a (weak or strict) decrease.) Since the maximal multigraph does not contain any edges (and hence, no edge of the form $i \xrightarrow{\succsim} i$), the AB-SCP cannot prove termination.

Another advantage of the AB-SCP over the size-change principle is an analysis that extracts and uses information about built-in arithmetic functions and predicates. But the AB-SCP still has the drawback of the size-change principle that it does not compare terms containing defined function symbols like half automatically. However, this would be needed in order to prove termination of algorithms like log above.

5 Comparison with Dependency Pairs

Now we compare the size-change principle with *dependency pairs*. In contrast to other recent techniques from term rewriting (e.g., [5,9]), the dependency pair approach has a direct relationship to the size-change principle. The reason is that both dependency pairs and size-change graphs are built from function calls (i.e., from left-hand sides of rules and subterms of right-hand sides which have a defined root symbol). This suggests to compare and to combine these approaches to benefit from their respective advantages. In Section 5.1 we briefly recapitulate the dependency pair approach and show in Section 5.2 that there are examples where the size-change principle is advantageous to the dependency pair approach. Then in Section 5.3 we discuss the drawbacks of the size-change principle compared to dependency pairs.

5.1 Dependency Pairs

We now introduce the dependency pair approach and refer to [2,10–12, 16,31] for refinements and motivations and to [12,15] for efficient algorithms to automate the approach. Let $\mathcal{F}^\sharp = \{f^\sharp \mid f \in \mathcal{D}\}$ be a set of *tuple symbols*, where $f^\sharp$ has the same arity as f and we often write F for $f^\sharp$, etc. If $t = g(t_1, \ldots, t_m)$ with $g \in \mathcal{D}$, we write $t^\sharp$ for $g^\sharp(t_1, \ldots, t_m)$. If $l \to r \in \mathcal{R}$ and t is a subterm of r with defined root symbol, then the rewrite rule $l^\sharp \to t^\sharp$ is called a *dependency pair* of $\mathcal{R}$. So the TRS $\mathcal{R} = \{\mathsf{f}(\mathsf{s}(x), y) \to \mathsf{f}(x, \mathsf{s}(x)),\ \mathsf{f}(x, \mathsf{s}(y)) \to \mathsf{f}(y, x)\}$ from Example 2 has the following dependency pairs.

$$\mathsf{F}(\mathsf{s}(x), y) \to \mathsf{F}(x, \mathsf{s}(x)) \qquad (4) \qquad\qquad \mathsf{F}(x, \mathsf{s}(y)) \to \mathsf{F}(y, x) \qquad (5)$$

We always assume that different occurrences of dependency pairs are variable disjoint. Then a TRS is (innermost) terminating iff there is no infinite (innermost) *chain* of dependency pairs. A sequence $s_1 \to t_1, s_2 \to t_2, \ldots$ of dependency pairs is a *chain* iff $t_i\sigma \to_{\mathcal{R}}^* s_{i+1}\sigma$ for all i and a suitable substitution σ. The sequence is an *innermost* chain iff $t_i\sigma \xrightarrow{\mathrm{i}}_{\mathcal{R}}^* s_{i+1}\sigma$ and all $s_i\sigma$ are in normal form.

To estimate which dependency pairs may occur consecutively in (innermost) chains, one builds a so-called *(innermost) dependency graph* whose nodes are the dependency pairs and there is an edge from $s \to t$ to $v \to w$ iff $s \to t$, $v \to w$ is an (innermost) chain. Since it is undecidable whether two dependency pairs form an (innermost) chain, for automation one constructs *estimated* graphs such that all edges in the real graph are also edges in the estimated graph (see, e.g., [2,14,26]). For the TRS of Example 2 we obtain the following dependency graph.

$$\begin{tikzcd} \boxed{\mathsf{F}(\mathsf{s}(x), y) \to \mathsf{F}(x, \mathsf{s}(x))} \arrow[loop above] \arrow[rr, leftrightarrow] & & \boxed{\mathsf{F}(x, \mathsf{s}(y)) \to \mathsf{F}(y, x)} \arrow[loop above] \end{tikzcd}$$

A non-empty set $\mathcal{P}$ of dependency pairs is a *cycle* iff for any pairs $s \to t$ and $v \to w$ in $\mathcal{P}$ there is a non-empty path from $s \to t$ to $v \to w$ in the graph which only traverses pairs from $\mathcal{P}$. In our example we have the cycles $\{(4)\}$, $\{(5)\}$, and $\{(4), (5)\}$. If a cycle only contains dependency pairs resulting from the rules $\mathcal{R}' \subseteq \mathcal{R}$ we speak of an $\mathcal{R}'$*-cycle* of the dependency graph of $\mathcal{R}$. For every cycle of the graph, we generate a set of inequality constraints such that the existence of a reduction pair $(\succsim, \succ)$ satisfying the constraints guarantees that there are no infinite (innermost) chains of dependency pairs from the cycle. Since we only regard finite TRSs, every infinite (innermost) chain would correspond to a cycle and hence, in this way (innermost) termination is proved. Essentially, the constraints require that at least one dependency pair from each cycle must be strictly decreasing (w.r.t. $\succ$) and the remaining ones must be weakly decreasing (w.r.t. $\succsim$). Moreover, when going from the right-hand side of a dependency pair to the left-hand side of the next dependency pair in a chain, we need a weak decrease. Therefore, we restrict ourselves to monotonic quasi-orders $\succsim$ and require $l \succsim r$ for all rules $l \to r$.

When proving innermost termination, we do not have to demand $l \succsim r$ for all rules $l \to r$, but only for those rules that are *usable* to reduce right-hand sides of dependency pairs if their variables are instantiated by normal forms. For $f \in \mathcal{D}$ we define its *usable rules* $\mathcal{U}(f)$ as the smallest set containing all f-rules and all rules that are usable for function symbols occurring in right-hand sides of f-rules. In our example, the usable rules for f are (1) and (2). For $\mathcal{D}' \subseteq \mathcal{D}$ let $\mathcal{U}(\mathcal{D}') = \bigcup_{f \in \mathcal{D}'} \mathcal{U}(f)$. Moreover, for a set of dependency pairs $\mathcal{P}$, we define $\mathcal{U}(\mathcal{P}) = \mathcal{U}(\{f \mid f \in \mathcal{D}$ occurs in t for some $s \to t \in \mathcal{P}\})$.

Theorem 19 (Dependency Pair Approach [11]) *A TRS $\mathcal{R}$ is terminating iff for each cycle $\mathcal{P}$ in the (estimated) dependency graph there is a monotonic reduction pair $(\succsim, \succ)$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ such that*

(a) $s \succ t$ for at least one $s \to t \in \mathcal{P}$ and $s \succsim t$ for all remaining $s \to t \in \mathcal{P}$
(b) $l \succsim r$ for all $l \to r \in \mathcal{R}$

$\mathcal{R}$ is innermost terminating if for each cycle $\mathcal{P}$ in the (estimated) innermost dependency graph there is a monotonic reduction pair $(\succsim, \succ)$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ such that

(c) $s \succ t$ *for at least one* $s \to t \in \mathcal{P}$ *and* $s \succsim t$ *for all remaining* $s \to t \in \mathcal{P}$
(d) $l \succsim r$ *for all* $l \to r \in \mathcal{U}(\mathcal{P})$

For the TRS in Example 2, in the cycle $\mathcal{P} = \{(4), (5)\}$ we have to find a reduction pair such that one dependency pair is weakly decreasing and the other one is strictly decreasing. Of course, we want to use standard techniques to synthesize reduction pairs $(\succsim, \succ)$ satisfying the constraints of the dependency pair approach. Most existing techniques generate monotonic *orders* $\succ$. However, for the dependency pair approach only the *quasi-order* $\succsim$ must be monotonic, whereas $\succ$ does not have to be monotonic. For that reason, before synthesizing a suitable order, some of the arguments of function symbols can be eliminated. To perform this elimination of arguments resp. of function symbols, the concept of *argument filtering* was introduced in [2] (here we use the notation of [21]).

Definition 20 (Argument Filtering) *An* argument filtering π *for a signature* $\mathcal{F}$ *maps every* n*-ary function symbol to an argument position* $i \in \{1, \ldots, n\}$ *or to a (possibly empty) list* $[i_1, \ldots, i_m]$ *of argument positions with* $1 \le i_1 < \ldots < i_m \le n$*. The signature* $\mathcal{F}_\pi$ *consists of all function symbols* f *with* $\pi(f) = [i_1, \ldots, i_m]$*, where in* $\mathcal{F}_\pi$ *the arity of* f *is* m*. Every argument filtering* π *induces a mapping from* $\mathcal{T}(\mathcal{F}, \mathcal{V})$ *to* $\mathcal{T}(\mathcal{F}_\pi, \mathcal{V})$*, also denoted by* π*, which is defined as:*

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

An argument filtering with $\pi(f) = i$ *for some* $f \in \mathcal{F}$ *is called* collapsing.

For an argument filtering on $\mathcal{F} \cup \mathcal{F}^\sharp$ and a relation $\succ$ on $\mathcal{T}(\mathcal{F}_\pi \cup \mathcal{F}^\sharp_\pi, \mathcal{V})$, let $\succ_\pi$ denote the relation on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ with $s \succ_\pi t$ iff $\pi(s) \succ \pi(t)$. As observed in [2], if $(\succsim, \succ)$ is a monotonic reduction pair on $\mathcal{T}(\mathcal{F}_\pi \cup \mathcal{F}^\sharp_\pi, \mathcal{V})$, then $(\succsim_\pi, \succ_\pi)$ is a monotonic reduction pair on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$. However, while $\succsim_\pi$ is monotonic, $\succ_\pi$ is usually not monotonic, even if $\succ$ is monotonic. Thus, in order to find monotonic reduction pairs in Theorem 19 one may first preprocess the terms in the inequalities by an argument filtering and afterwards use standard techniques to search for a reduction pair satisfying the filtered constraints.

5.2 Advantages of the Size-Change Principle

To continue the termination proof of the TRS from Example 2 using Theorem 19, we may eliminate the second argument position of F by choosing an argument filtering with $\pi(\mathsf{F}) = [1]$ (and $\pi(\mathsf{s}) = [1]$). In this way, F becomes unary and every term $\mathsf{F}(s, t)$ is replaced by $\mathsf{F}(\pi(s))$. Then the constraint $\mathsf{F}(\mathsf{s}(x)) \succ \mathsf{F}(x)$ resulting from the dependency pair (4) is easily satisfied but there is no reduction pair satisfying the constraint $\mathsf{F}(x) \succsim \mathsf{F}(y)$ from the second dependency pair (5). Indeed, there exists no argument filtering such that the constraints resulting from the dependency pair approach would be satisfied by a standard path order like *RPOS* or *KBO*. Moreover, if one adds the rules $\mathsf{f}(x,y) \to \mathsf{ack}(x,y)$, $\mathsf{ack}(\mathsf{s}(x),y) \to \mathsf{f}(x,x)$, and the rules for the Ackermann function ack from Example 15, then the dependency pair constraints are not satisfied by any polynomial order either.

Thus, termination cannot be proved with dependency pairs in combination with classical orders amenable to automation, whereas the proof is very easy with the size-change principle and a simple reduction pair like the embedding order on constructors. While the examples in [24] are easily handled by dependency pairs and *RPOS*, this shows that there exist TRSs where the size-change principle is preferable to dependency pairs and standard rewrite orders.

In fact, size-change termination encompasses the concept of argument filtering for root symbols, since it concentrates on certain arguments of (root) function symbols while ignoring others. This is an advantage compared to dependency pairs where finding the argument filtering is a major search problem. Moreover, the size-change principle examines sequences of function calls in a more sophisticated way. Depending on the different "paths" from one function call to another, it can choose different arguments to be (strictly) decreasing. In contrast, in the dependency pair approach such choices remain fixed for the whole cycle.

5.3 Disadvantages of the Size-Change Principle

However, the size-change principle also has severe drawbacks compared to dependency pairs. In addition to the drawbacks mentioned in Section 4, a disadvantage of the size-change principle is that it is not modular, i.e., one has to use the same reduction pair for the whole termination proof whereas the dependency pair approach permits different orders for different cycles. The size-change principle also does not analyze arguments of terms to check whether two function calls can follow each other, whereas such a check is performed in the construction of (innermost) dependency graphs. Again, the most severe drawback is that the size-change principle offers no technique to compare terms with defined symbols, whereas dependency pairs use inequalities of the form $l \succsim r$ for this purpose. For that reason, only very restricted reduction pairs may be used for the size-change principle in Theorems 9 and 12, whereas one may use arbitrary monotonic reduction pairs for the dependency pair approach. In fact, dependency pairs are a *complete* technique which can prove termination of every terminating TRS, which is not the case for the size-change principle (see e.g., Example 18).

6 Combination with Dependency Pairs

After having analyzed their respective advantages and drawbacks, we now introduce a new technique to combine dependency pairs and the size-change principle. A straightforward approach would be to use dependency pairs as a preprocessing step and size-change termination as the "base order" when trying to satisfy the constraints resulting from the dependency pair approach. However, this would be very weak due to the restrictions on the reduction pairs in Theorems 9 and 12.

Instead, we incorporate the size-change principle into the dependency pair approach and use it when generating the constraints. The resulting technique is stronger than both previous approaches: If (innermost) termination can be proved by the size-change principle or by dependency pairs using certain reduction pairs, then it can also be proved with our new technique using the *same* reduction pairs. Moreover, there are many examples which cannot be proved by the size-change principle and where dependency pairs would require complicated reduction pairs (that can hardly be generated automatically), whereas with our combined technique the (automatic) proof works with very simple reduction pairs. Of course, if one uses more advanced reduction pairs in the combined method one obtains an even more powerful approach for automated termination proofs, cf. Section 8.

Size-change graphs and dependency pairs have a close correspondence, since both represent a call of a defined symbol g in the right-hand side of a rule $f(s_1, \ldots, s_n) \to \ldots g(t_1, \ldots, t_m) \ldots$ Since we only need to concatenate size-change graphs which correspond to cycles in the (innermost) dependency graph, we now label size-change graphs by the corresponding dependency pair and multigraphs are labelled by the corresponding sequence of dependency pairs. Then two size-change graphs or multigraphs labelled with $(\ldots, D)$ and $(D', \ldots)$ may only be concatenated (for termination proofs) if there is an arc from D to D' in the (estimated) dependency graph. If one proves innermost termination instead of termination, then concatenation is allowed whenever there is an arc from D to D' in the (estimated) innermost dependency graph. Another problem is that in size-change graphs one only has output nodes $1_f, \ldots, n_f$ and input nodes $1_g, \ldots, m_g$ to compare the *arguments* of f and g. Therefore, the size-change principle cannot deal with TRSs like Example 18 where one has to regard the *whole* term in order to show termination. For that reason we add another output node ε_f and input node ε_g which correspond to the whole terms (or more precisely, to the terms $F(s_1, \ldots, s_n)$ and $G(t_1, \ldots, t_m)$ of the corresponding dependency pair).

Definition 21 (Extended Size-Change Graphs) *Let $(\succsim, \succ)$ be a reduction pair on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$. For every rule $f(s_1, \ldots, s_n) \to r$ of a TRS $\mathcal{R}$ and every subterm $g(t_1, \ldots, t_m)$ of r with $g \in \mathcal{D}$, the* extended size-change graph *has $n{+}1$ output nodes i_f and $m{+}1$ input nodes j_g where $i \in \{\varepsilon, 1, \ldots, n\}$, $j \in \{\varepsilon, 1, \ldots, m\}$. Let $s = F(s_1, \ldots, s_n)$ and $t = G(t_1, \ldots, t_m)$. Then there is an edge $i_f \xrightarrow{\succ} j_g$ iff $s|_i \succ t|_j$ and otherwise, there is an edge $i_f \xrightarrow{\succsim} j_g$ iff $s|_i \succsim t|_j$. Moreover, every extended size-change graph is labelled by a one-element sequence $(F(s_1, \ldots, s_n) \to G(t_1, \ldots, t_m))$.*

Concatenation of extended size-change graphs to extended multigraphs works as in Definition 3. However, if G is a multigraph labelled with the se-

quence $(D_1, \ldots, D_k)$ and H is labelled with $(D'_1, \ldots, D'_{k'})$, then they can only be concatenated if there is an arc from D_k to D'_1 in the (estimated) dependency graph or in the (estimated) innermost dependency graph, respectively. The concatenation $G \bullet H$ is labelled with $(D_1, \ldots, D_k, D'_1, \ldots, D'_{k'})$.

As an example, reconsider the TRS for the Ackermann function from Example 15. The rule $\mathsf{ack}(\mathsf{s}(x), \mathsf{0}) \to \mathsf{ack}(x, \mathsf{s}(\mathsf{0}))$ gives rise to the following extended size-change graph if we use the embedding order on constructors and tuple symbols.

$$\begin{tikzcd} \varepsilon_{\mathsf{ack}} \arrow[dr, "\succ"] & \varepsilon_{\mathsf{ack}} \\ 1_{\mathsf{ack}} \arrow[r, "\succ"'] & 1_{\mathsf{ack}} \\ 2_{\mathsf{ack}} & 2_{\mathsf{ack}} \end{tikzcd}$$

This graph is labelled with the singleton sequence consisting of the dependency pair $\mathsf{ACK}(\mathsf{s}(x), \mathsf{0}) \to \mathsf{ACK}(x, \mathsf{s}(\mathsf{0}))$. Thus, it cannot be concatenated with itself, since there is no arc from this dependency pair to itself in the estimation of the (innermost) dependency graph.

In the remainder, when we speak of size-change graphs or multigraphs, we always mean *extended* graphs. Obviously, there may exist infinitely many multigraphs due to the labelling with a sequence of dependency pairs. However, two multigraphs with labels $(D_1, \ldots, D_k)$ and $(D'_1, \ldots, D'_{k'})$ are identified if their nodes and edges are identical and if $D_1 = D'_1$, $D_k = D'_{k'}$, and $\{D_1, \ldots, D_k\} = \{D'_1, \ldots, D'_{k'}\}$. Thus, for the label only the set of dependency pairs and the first and last dependency pair of the sequences is important. Then, there are again only finitely many different multigraphs.

To combine dependency pairs and the size-change principle, now we only regard multigraphs labelled with a cycle $\mathcal{P}$ of the (estimated) dependency or innermost dependency graph, respectively (i.e., they are labelled with $(D_1, \ldots, D_k)$ such that $\mathcal{P} = \{D_1, \ldots, D_k\}$). Moreover, one may use different reduction pairs for the multigraphs resulting from different cycles. To benefit from the advantages of the size-change principle (i.e., combining lexicographic and multiset comparison and using different argument filterings and strict inequalities within one cycle), we do not build inequalities but size-change graphs out of the dependency pairs.

The following theorem combines dependency pairs and the size-change principle (Theorem 12) for full termination. In contrast to Theorem 12 we now allow arbitrary reduction pairs. However, to handle defined symbols properly, then one has to require that all rules are weakly decreasing (like in the dependency pair approach). Alternatively, as in Theorem 12 one may also use reduction pairs $(\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$ for a terminating non-duplicating TRS $\mathcal{S}$ over the constructors $\mathcal{C}$ of $\mathcal{R}$ and the tuple symbols $\mathcal{F}^\sharp$ without requiring that $\mathcal{R}$'s rules are weakly decreasing. For example, in this way one can prove termination of the Ackermann TRS with the embedding order (i.e., $\mathcal{S} = \mathit{Emb}_{\mathcal{C} \cup \mathcal{F}^\sharp}$). However, in order to use $(\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$ for some cycles and other reduction pairs $(\succsim, \succ)$ for other cycles, one has to prove termination of $\mathcal{R} \cup \mathcal{S}$ instead of just $\mathcal{R}$.

Example 22 To illustrate this, let $\mathcal{R} = \{\mathsf{g}(\mathsf{f}(\mathsf{a})) \to \mathsf{g}(\mathsf{f}(\mathsf{b})), \mathsf{f}(\mathsf{b}) \to \mathsf{f}(\mathsf{a})\}$ and $\mathcal{S} = \{\mathsf{a} \to \mathsf{b}\}$. The only cycle of $\mathcal{R}$'s dependency graph is $\{\mathsf{G}(\mathsf{f}(\mathsf{a})) \to \mathsf{G}(\mathsf{f}(\mathsf{b}))\}$ and for this cycle, size-change termination can be shown using $(\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$. Thus, if one only regards $\mathcal{R}$ instead of $\mathcal{R} \cup \mathcal{S}$, one could falsely "prove" termination of $\mathcal{R}$. Instead, $\{\mathsf{F}(\mathsf{b}) \to \mathsf{F}(\mathsf{a})\}$ must also be regarded, since it is an $\mathcal{R}$-cycle of the dependency graph of $\mathcal{R} \cup \mathcal{S}$ (because in $\mathcal{R} \cup \mathcal{S}$, a is a defined symbol). Moreover, for reduction pairs $(\succsim, \succ) \neq (\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$, one has to demand $l \succsim r$ not only for the rules $l \to r$ of $\mathcal{R}$, but for those of $\mathcal{S}$ as well. Otherwise, the constraints for the cycle $\{\mathsf{F}(\mathsf{b}) \to \mathsf{F}(\mathsf{a})\}$ would falsely be satisfiable.

By Theorem 23, the resulting termination criterion is sound, complete, and more powerful than the size-change principle or dependency pairs on their own.

Theorem 23 (Termination Proofs) *Let $\mathcal{R}$ be a TRS over $\mathcal{F}$ with constructors $\mathcal{C}$ and let $\mathcal{S}$ be a terminating non-duplicating TRS over $\mathcal{C} \cup \mathcal{F}^\sharp$. $\mathcal{R} \cup \mathcal{S}$ is terminating iff for each $\mathcal{R}$-cycle $\mathcal{P}$ in the (estimated) dependency graph of $\mathcal{R} \cup \mathcal{S}$ there is a monotonic reduction pair $(\succsim, \succ)$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ such that*

(a′) all maximal multigraphs w.r.t. $(\succsim, \succ)$ labelled with $\mathcal{P}$ contain an edge $i \xrightarrow{\succ} i$

(b′) $\succsim \, = \, \to_{\mathcal{S}}^*$ and $\succ \, = \, \to_{\mathcal{S}}^+$ or $l \succsim r$ for all $l \to r \in \mathcal{R} \cup \mathcal{S}$

The termination criterion with the conditions (a′) and (b′) encompasses both size-change termination and the dependency pair approach: If $\mathcal{R}$ is size-change terminating w.r.t. a reduction pair $(\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$ as in Theorem 12, then the reduction pair $(\succsim, \succ) := (\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ also satisfies the conditions (a′) and (b′) above. Moreover, if a reduction pair $(\succsim, \succ)$ satisfies Conditions (a) and (b) of Theorem 19 for termination with dependency pairs, then $(\succsim, \succ)$ also satisfies the conditions (a′) and (b′) above for $\mathcal{S} = \varnothing$.

Proof The above criterion can simulate size-change termination (Theorem 12): if every maximal multigraph contains an edge $i \xrightarrow{\succ} i$ then this also holds for those maximal multigraphs that are labelled with $\mathcal{P}$. It can also simulate dependency pairs by choosing $\mathcal{S} = \varnothing$: Condition (a) in Theorem 19 implies that every multigraph labelled with $\mathcal{P}$ must contain the edge $\varepsilon \xrightarrow{\succ} \varepsilon$. Since the dependency pair approach is *complete* for termination (even with estimated or no dependency graphs), this also proves the "only if" direction.

For the "if" direction, suppose that $\mathcal{R} \cup \mathcal{S}$ is not terminating. Since $\mathcal{S}$ terminates, by Lemma 11 and the soundness of dependency pairs, there is an infinite chain $s_1 \to t_1, s_2 \to t_2, \ldots$ of $\mathcal{R}$-dependency pairs such that $t_i\sigma \to_{\mathcal{R}\cup\mathcal{S}}^* s_{i+1}\sigma$ for all i and a substitution σ, and each s_i has the form $l_i^\sharp$ for a minimal non-terminating term $l_i\sigma$ w.r.t. $\mathcal{R} \cup \mathcal{S}$. Those dependency pairs which occur infinitely often in this chain must form a cycle $\mathcal{P}$ of the dependency graph of $\mathcal{R} \cup \mathcal{S}$ and since the chain only contains $\mathcal{R}$-dependency pairs, $\mathcal{P}$ is an $\mathcal{R}$-cycle. Let $i_1 < i_2 < \ldots$ such that $\{s_{i_j} \to t_{i_j}, \ldots, s_{i_{j+1}-1} \to t_{i_{j+1}-1}\} = \mathcal{P}$ for all $j \geq 1$, i.e., we partition the sequence into parts where all dependency pairs of $\mathcal{P}$ occur. For all j, let G_j be the multigraph resulting from the concatenation of the size-change graphs corresponding to $s_{i_j} \to t_{i_j}, \ldots, s_{i_{j+1}-1} \to t_{i_{j+1}-1}$. Note that all G_j are labelled with $\mathcal{P}$.

Due to (a′), every multigraph H resulting from concatenation of size-change graphs contains an edge of the form $i \xrightarrow{\succsim} i$, provided that $H = H \bullet H$ and that H is labelled with $\mathcal{P}$. Hence, every idempotent multigraph $H = H \bullet H$ resulting from concatenating graphs from $G_1, G_2, \ldots$ also contains an edge $i \xrightarrow{\succsim} i$. The reason is that since all G_j are labelled with $\mathcal{P}$, H is also labelled with $\mathcal{P}$.

Let $\Gamma = \{G_1, G_2, \ldots\}$. Obviously, Γ is finite since there can only be finitely many multigraphs (as $\mathcal{F}$ is finite). Since every maximal multigraph $G_1 \bullet G_2 \bullet \ldots \bullet G_n$ with $G_1, \ldots, G_n \in \Gamma$ has an edge of the form $i \xrightarrow{\succsim} i$, Lemma 6 implies that there is an infinite path with infinitely many $\xrightarrow{\succ}$-edges in the graph $G_1 \circ G_2 \circ \ldots$ Hence, there is also such an infinite path in the infinite graph resulting from the size-change graphs corresponding to $s_1 \to t_1, s_2 \to t_2, \ldots$ Without loss of generality, we assume that the infinite path already starts in the size-change graph corresponding to $s_1 \to t_1$. For every i, let a_i be the output node in the size-change graph of $s_i \to t_i$ which is on this path. For infinitely many i we have $s_i|_{a_i}\sigma \succ t_i|_{a_{i+1}}\sigma$ and otherwise, we have $s_i|_{a_i}\sigma \succsim t_i|_{a_{i+1}}\sigma$, since $\succsim$ and $\succ$ are closed under substitutions.

If the reduction pair $(\succsim, \succ)$ is $(\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$, then we proceed as in the proof of Theorem 12: We have $s_i|_{a_i}\sigma \succ t_i|_{a_{i+1}}\sigma \to_{\mathcal{R}\cup\mathcal{S}}^* s_{i+1}|_{a_{i+1}}\sigma$ (i.e., $s_i|_{a_i}\sigma \to_{\mathcal{S}}^+ t_i|_{a_{i+1}}\sigma \to_{\mathcal{R}\cup\mathcal{S}}^* s_{i+1}|_{a_{i+1}}\sigma$) for infinitely many i and $s_i|_{a_i}\sigma \succsim t_i|_{a_{i+1}}\sigma \to_{\mathcal{R}\cup\mathcal{S}}^* s_{i+1}|_{a_{i+1}}\sigma$ (i.e., $s_i|_{a_i}\sigma \to_{\mathcal{S}}^* t_i|_{a_{i+1}}\sigma \to_{\mathcal{R}\cup\mathcal{S}}^* s_{i+1}|_{a_{i+1}}\sigma$) for all other i. Hence, $s_1|_{a_1}\sigma = l_1^\sharp|_{a_1}\sigma$ does not terminate w.r.t. $\mathcal{R} \cup \mathcal{S}$. If $a_1 \neq \varepsilon$, then $l_1^\sharp|_{a_1}\sigma$ is a subterm of $l_1\sigma$ which contradicts the minimality of the latter term. If $a_1 = \varepsilon$, then Lemma 11 implies that $l_1^\sharp\sigma$ must have a proper subterm which is non-terminating as well. This again contradicts the minimality of $l_1\sigma$. Otherwise if $(\succsim, \succ) \neq (\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$, we have $t_i|_{a_{i+1}}\sigma \succsim s_{i+1}|_{a_{i+1}}\sigma$ due to (b′) since $t_i|_{a_{i+1}}\sigma \to_{\mathcal{R}\cup\mathcal{S}}^* s_{i+1}|_{a_{i+1}}\sigma$. Hence, we obtain an infinite decreasing sequence w.r.t. $\succ$ which contradicts its well-foundedness. □

For the automation of Theorem 23 we choose $\mathcal{S} = \varnothing$ which results in the following corollary. Of course, this corollary does not encompass the termination criterion with the size-change principle of Theorem 12 anymore. For example, in contrast to Theorem 12, this corollary can no longer prove termination of the Ackermann TRS with the embedding order. Nevertheless, our experiments in Section 8 show that this corollary (which only replaces Condition (a) of Theorem 19 by (a′)) already increases performance significantly.

Corollary 24 (Automated Termination Proofs) *A TRS $\mathcal{R}$ is terminating iff for each cycle $\mathcal{P}$ in the (estimated) dependency graph there is a monotonic reduction pair $(\succsim, \succ)$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ such that*

(a′) all maximal multigraphs w.r.t. $(\succsim, \succ)$ labelled with $\mathcal{P}$ contain an edge $i \xrightarrow{\succ} i$
(b) $l \succsim r$ for all $l \to r \in \mathcal{R}$

In the corresponding approach for innermost termination, we integrate the technique of Theorem 9 with dependency pairs. In the dependency pair approach for innermost termination, only the *usable* rules for defined symbols in right-hand sides t of dependency pairs $s \to t$ have to be weakly decreasing. Here, one can benefit from the size-change principle, which restricts the comparison of terms to certain arguments. Function symbols of t which do not occur in the arguments being compared do not have to be regarded as being "usable". More precisely, if one uses the restriction of a reduction pair where one can only compare s and t if t's symbols come from a subset $\mathcal{D}' \subseteq \mathcal{D}$, then one only has to require weak decreasingness of $\mathcal{U}(\mathcal{D}')$. Thus, here the size-change principle has the important advantage that one can reduce the set of usable rules.

For example, the TRS for the Ackermann function from Example 15 has the rule $\mathsf{ack}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{ack}(x, \mathsf{ack}(\mathsf{s}(x), y))$ and therefore, we obtain the dependency pair $\mathsf{ACK}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{ACK}(x, \mathsf{ack}(\mathsf{s}(x), y))$. Since ack occurs in the right-hand side of this dependency pair, in the dependency pair approach we would have to require $l \succsim r$ for all ack-rules since they would be regarded as being usable. For this reason, we would need a lexicographic comparison. However, in our new technique, the ACK-dependency pairs are transformed into size-change graphs and size-change termination can easily be shown using the embedding order on constructor terms (i.e., $\mathcal{D}' = \varnothing$). In other words, the second argument of $\mathsf{ACK}(x, \mathsf{ack}(\mathsf{s}(x), y))$ is never regarded in this comparison and therefore, the ack-rules are no longer usable. So instead of *LPO* we only need the embedding order to satisfy the resulting constraints. Hence, in the combined technique one can often use much simpler reduction pairs than the reduction pairs needed with dependency pairs.

Theorem 25 (Innermost Termination Proofs) *A TRS $\mathcal{R}$ is innermost terminating if for each cycle $\mathcal{P}$ in the (estimated) innermost dependency graph there is a subset $\mathcal{D}' \subseteq \mathcal{D}$ and a reduction pair on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ which is monotonic if $\mathcal{D}' \neq \varnothing$, such that for its $(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp)$-restriction $(\succsim, \succ)$ we have*

(c′) all maximal multigraphs w.r.t. $(\succsim, \succ)$ labelled with $\mathcal{P}$ contain an edge $i \xrightarrow{\succ} i$
(d′) $l \succsim r$ for all $l \to r \in \mathcal{U}(\mathcal{D}')$

The innermost termination criterion with the conditions (c′) and (d′) encompasses both size-change termination and the dependency pair approach: If $\mathcal{R}$ is size-change terminating w.r.t. the $\mathcal{C}$-restriction $(\succsim, \succ)$ of a reduction *pair as in Theorem 9, then* $(\succsim, \succ)$ *also satisfies the conditions (c′) and (d′) above for* $\mathcal{D}' = \varnothing$*. Moreover, if a reduction pair satisfies Conditions (c) and (d) of Theorem 19 for innermost termination with dependency pairs, then its* $(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp)$*-restriction* $(\succsim, \succ)$ *also satisfies the conditions (c′) and (d′) above, where* $\mathcal{D}'$ *are the defined symbols of the TRS* $\mathcal{U}(\mathcal{P})$*.*

Proof Theorem 25 can simulate the size-change principle: As in Theorem 23, size-change termination implies (c′). Moreover, if $(\succsim, \succ)$ is the $\mathcal{C}$-restriction of a reduction pair as in Theorem 9, then $\mathcal{D}' = \varnothing$ and thus, (d′) is also satisfied.

To show that Theorem 25 can simulate dependency pairs, let $(\widehat{\succsim}, \widehat{\succ})$ be a reduction pair on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ satisfying Theorem 19 (c) and (d) for some cycle $\mathcal{P}$. By choosing $\mathcal{D}'$ to consist of the defined symbols of the TRS $\mathcal{U}(\mathcal{P})$, we have $\mathcal{U}(\mathcal{D}') = \mathcal{U}(\mathcal{P})$ and all defined symbols occurring in some right-hand side t with $s \to t \in \mathcal{P} \cup \mathcal{U}(\mathcal{D}')$ are contained in $\mathcal{D}'$. Thus, in (c′) and (d′) one only compares terms s and t where $t \in \mathcal{T}(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp, \mathcal{V})$. Hence, Theorem 19 (c) and (d) do not only imply that (c′) and (d′) hold for the original reduction pair $(\widehat{\succsim}, \widehat{\succ})$, but also for its $(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp)$-restriction $(\succsim, \succ)$.

The soundness of the above criterion is shown as for Theorem 23. If $\mathcal{R}$ is not innermost terminating, then there is an infinite innermost chain $s_1 \to t_1, s_2 \to t_2, \ldots$ with $t_i\sigma \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} s_{i+1}\sigma$ and all $s_i\sigma$ are normal forms. As in Theorem 23's proof, this innermost chain corresponds to a cycle $\mathcal{P}$ of the innermost dependency graph. Moreover, in the infinite graph resulting from the corresponding size-change graphs there is an infinite path with infinitely many "$\succ$" labels. For every i, let a_i be the output node in the size-change graph corresponding to $s_i \to t_i$ which is on this infinite path. Let $(\widehat{\succsim}, \widehat{\succ})$ be the reduction pair on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ whose $(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp)$-restriction $(\succsim, \succ)$ satisfies (c′) and (d′). To conclude $t_i|_{a_{i+1}}\sigma \mathrel{\widehat{\succsim}} s_{i+1}|_{a_{i+1}}\sigma$, first note that $s_i|_{a_i} \succsim t_i|_{a_{i+1}}$ or $s_i|_{a_i} \succ t_i|_{a_{i+1}}$. Recall that $t_i|_{a_{i+1}} \in \mathcal{T}(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp, \mathcal{V})$ and that σ instantiates all variables of s_i (and hence, of t_i) to normal forms. Thus, the only rules applicable to $t_i|_{a_{i+1}}\sigma$ are from $\mathcal{U}(\mathcal{D}')$. If $\mathcal{D}' = \varnothing$, then this implies $t_i|_{a_{i+1}}\sigma = s_{i+1}|_{a_{i+1}}\sigma$. Otherwise, (d′) ensures $t_i|_{a_{i+1}}\sigma \mathrel{\widehat{\succsim}} s_{i+1}|_{a_{i+1}}\sigma$ by the stability and monotonicity of $\widehat{\succsim}$. Since we also have $s_i|_{a_i} \mathrel{\widehat{\succ}} t_i|_{a_{i+1}}$ for infinitely many i and $s_i|_{a_i} \mathrel{\widehat{\succsim}} t_i|_{a_{i+1}}$ for all remaining i, we obtain an infinite decreasing sequence w.r.t. $\widehat{\succ}$ which contradicts its well-foundedness. □

The combined technique handles TRSs where both original techniques fail, since some rules require a lexicographic or multiset comparison and others require polynomial orders. In the combined technique, a lexicographic or multiset comparison is implicit since the size-change principle is incorporated. Thus, the resulting constraints are often satisfied by simple polynomial orders. For example, we unite the plus-TRS (Example 18) with the ack-TRS (Example 15), where $\mathsf{ack}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{ack}(x, \mathsf{ack}(\mathsf{s}(x), y))$ is replaced by $\mathsf{ack}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{ack}(x, \mathsf{plus}(y, \mathsf{ack}(\mathsf{s}(x), y)))$. In the original dependency pair approach, both the ack- and plus-rules are usable for the corresponding dependency pair and thus, no standard order amenable to automation fulfills the resulting constraints. But in the combined technique, there are no usable rules and hence, the innermost termination proof works with the simple polynomial order on constructors and tuple symbols where $\mathsf{s}(x)$ is mapped to $x+1$ and $\mathsf{PLUS}(x,y)$ is mapped to $x+y$. In practice, there are many TRSs where the combined technique simplifies the termination proof significantly, cf. the following example and our experiments in Section 8.

Example 26 We consider the following TRS for sorting lists from [3, Example 3.10]. Here, nil denotes the empty list, $\mathsf{cons}(n,x)$ represents the insertion of the element n in front of the list x, $\mathsf{rm}(n,x)$ removes all occurrences of n from the list x, and $\mathsf{sort}(x,\mathsf{nil})$ returns a sorted version of the list x where duplicates are eliminated. To ease readability we use infix symbols "$=$", "$\leq$", and "$++$" for equality, comparison of natural numbers, and list concatenation, respectively.

$$
\begin{aligned}
\mathsf{0} = \mathsf{0} &\to \mathsf{true}\\
\mathsf{0} = \mathsf{s}(x) &\to \mathsf{false}\\
\mathsf{s}(x) = \mathsf{0} &\to \mathsf{false}\\
\mathsf{s}(x) = \mathsf{s}(y) &\to x = y\\
\mathsf{0} \leq y &\to \mathsf{true}\\
\mathsf{s}(x) \leq \mathsf{0} &\to \mathsf{false}\\
\mathsf{s}(x) \leq \mathsf{s}(y) &\to x \leq y\\
\mathsf{nil} ++ y &\to y\\
\mathsf{cons}(n,x) ++ y &\to \mathsf{cons}(n, x ++ y)\\
\mathsf{min}(\mathsf{cons}(n,\mathsf{nil})) &\to n\\
\mathsf{min}(\mathsf{cons}(n,\mathsf{cons}(m,x))) &\to \mathsf{if}_{\mathsf{min}}(n \leq m, \mathsf{cons}(n,\mathsf{cons}(m,x)))\\
\mathsf{if}_{\mathsf{min}}(\mathsf{true},\mathsf{cons}(n,\mathsf{cons}(m,x))) &\to \mathsf{min}(\mathsf{cons}(n,x))\\
\mathsf{if}_{\mathsf{min}}(\mathsf{false},\mathsf{cons}(n,\mathsf{cons}(m,x))) &\to \mathsf{min}(\mathsf{cons}(m,x))\\
\mathsf{rm}(n,\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{rm}(n,\mathsf{cons}(m,x)) &\to \mathsf{if}_{\mathsf{rm}}(n = m, n, \mathsf{cons}(m,x))\\
\mathsf{if}_{\mathsf{rm}}(\mathsf{true}, n, \mathsf{cons}(m,x)) &\to \mathsf{rm}(n,x)\\
\mathsf{if}_{\mathsf{rm}}(\mathsf{false}, n, \mathsf{cons}(m,x)) &\to \mathsf{cons}(m, \mathsf{rm}(n,x))\\
\mathsf{sort}(\mathsf{nil},\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{sort}(\mathsf{cons}(n,x), y) &\to \mathsf{if}_{\mathsf{sort}}(n = \mathsf{min}(\mathsf{cons}(n,x)), \mathsf{cons}(n,x), y)\\
\mathsf{if}_{\mathsf{sort}}(\mathsf{true}, \mathsf{cons}(n,x), y) &\to \mathsf{cons}(n, \mathsf{sort}(\mathsf{rm}(n,x) ++ y, \mathsf{nil}))\\
\mathsf{if}_{\mathsf{sort}}(\mathsf{false}, \mathsf{cons}(n,x), y) &\to \mathsf{sort}(x, \mathsf{cons}(n,y))
\end{aligned}
$$

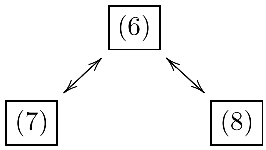
To automate the dependency pair approach, the following algorithm was suggested in [15]. One first solves the constraints of Theorem 19 for the strongly connected components (SCC) of the (estimated) dependency graph. Here, an SCC is a maximal cycle, i.e., a cycle that is not properly contained in any other cycle. Afterwards, one deletes all dependency pairs $s \to t$ from the graph where the strict constraints $s \succ t$ were satisfied. After the deletion one computes SCCs again and continues in this way.

Of course, it is particularly desirable for efficiency if already the deletion of the dependency pairs $s \to t$ with $s \succ t$ from the *initial* SCCs results in graphs with no further cycles. In other words, it would be advantageous if the generated reduction pair also satisfies the following constraints instead of just Constraint (a) of Theorem 19 for every (initial) SCC $\mathcal{P}$:

(a)$_1$ $s \succsim t$ for all $s \to t \in \mathcal{P}$
(a)$_2$ $s \succ t$ for at least one $s \to t \in \mathcal{P}'$ for each cycle $\mathcal{P}' \subseteq \mathcal{P}$

In our example, the most interesting part is to show termination of sort and $\mathsf{if}_{\mathsf{sort}}$. The corresponding SCC consists of the following three dependency pairs and has the following form.

$$\mathsf{SORT}(\mathsf{cons}(n,x),y) \to \mathsf{IF}_{\mathsf{sort}}(n = \mathsf{min}(\mathsf{cons}(n,x)), \mathsf{cons}(n,x), y) \quad (6)$$
$$\mathsf{IF}_{\mathsf{sort}}(\mathsf{true}, \mathsf{cons}(n,x), y) \to \mathsf{SORT}(\mathsf{rm}(n,x) +\!\!+\, y, \mathsf{nil}) \quad (7)$$
$$\mathsf{IF}_{\mathsf{sort}}(\mathsf{false}, \mathsf{cons}(n,x), y) \to \mathsf{SORT}(x, \mathsf{cons}(n,y)) \quad (8)$$

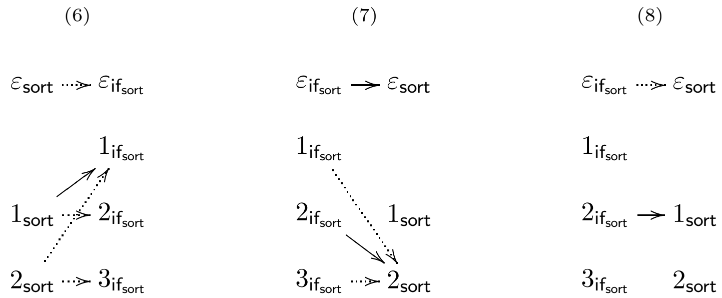


In order to prove the absence of infinite chains built from (6), (7), and (8), one can show that in each cycle either the sum of the list sizes of both SORT-arguments is reduced or the sum remains equal and the list in SORT's first argument is shortened. So one uses two different measures to compare SORT's arguments and combines these measures lexicographically. The list sizes can be expressed by simple linear polynomials, but the lexicographic combination of these measures cannot be expressed with simple polynomials. Therefore in [3], polynomials of degree 2 have been used to simulate the lexicographic comparison.

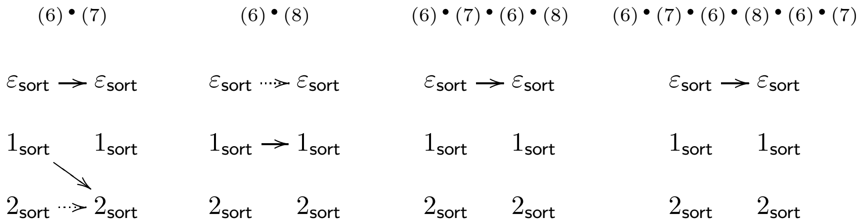
In contrast to this, with the combined approach of Corollary 24 we do not need complex polynomials, even when using the same reduction pairs for all cycles of an SCC. The reason is that the lexicographic combination can be simulated in the size-change graphs. We map $\mathsf{cons}(n,x)$ to $n+x+1$, the symbols $\mathsf{0}$, true, false, nil, $=$, $\leq$ are mapped to 0, we map $\mathsf{rm}(x,y)$ and $\mathsf{if}_{\mathsf{rm}}(b,x,y)$ to y, we map $\mathsf{min}(x)$ and $\mathsf{if}_{\mathsf{min}}(b,x)$ to x, and $x +\!\!+\, y$, $\mathsf{sort}(x,y)$, $\mathsf{SORT}(x,y)$, $\mathsf{if}_{\mathsf{sort}}(b,x,y)$, and $\mathsf{IF}_{\mathsf{sort}}(b,x,y)$ are mapped to $x+y$. Then all constraints from the rules can be oriented (i.e., we have $l \succsim r$ for all rules $l \to r$) and we obtain the following three size-change graphs.[3] To ease

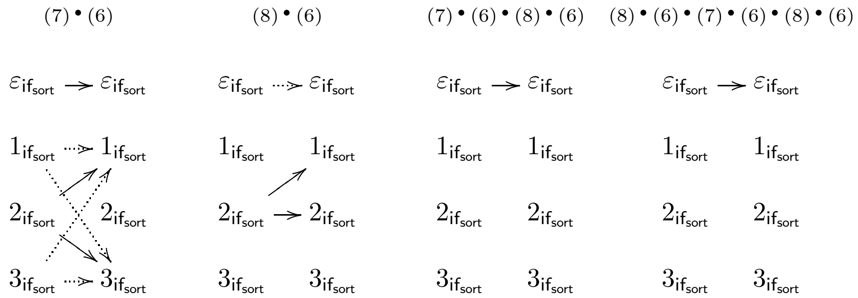
[3] In addition to the edges above, the output nodes labelled with ε have edges to all input nodes. We did not depict all of these edges to improve readability.

readability, we denoted $\xrightarrow{\succsim}$-edges by dotted arrows and $\xrightarrow{\succ}$-edges by solid arrows.



In this example, there are eight maximal multigraphs (where we again identify multigraphs with the labels $(D_1, \ldots, D_k)$ and $(D'_1, \ldots, D'_{k'})$ if $D_1 = D'_1$, $D_k = D'_{k'}$, and $\{D_1, \ldots, D_k\} = \{D'_1, \ldots, D'_{k'}\}$, provided that their nodes and edges are identical).





It is easy to see that all maximal multigraphs either contain an $\xrightarrow{\succ}$-edge between their ε-nodes or there is an $\xrightarrow{\succsim}$-edge between the ε-nodes and an $\xrightarrow{\succ}$-edge between the but-last argument nodes.

One should remark that in contrast to the approach for termination, the dependency pair approach for innermost termination in Theorem 19 only provides a sufficient, but not a necessary criterion. An example of an innermost terminating TRS where the constraints (c) and (d) of Theorem 19 are unsatisfiable is given in [2, Example 43]. Similarly, this example also shows that the combination of dependency pairs and the size-change principle in Theorem 25 does not yield a necessary criterion for innermost termination either.

To summarize, the combination of dependency pairs and the size-change principle has two main advantages: First, one can now prove (innermost) termination of TRSs automatically where up to now an automated proof was impossible. Second, for many TRSs where up to now the termination proof required complicated reduction pairs involving a large search space, one can now use much simpler orders which increases efficiency. These advantages are confirmed by our experiments in Section 8.

7 Complexity

In [24] it was shown that proving size-change termination is PSPACE-complete. In contrast, up to now there have been no results about the complexity of the dependency pair approach. To allow a comparison between dependency pairs and the size-change principle, we present new contributions analyzing the complexity of both the dependency pair approach and of the new technique from Section 6 which combines dependency pairs with the size-change principle.

Section 7.1 shows that while the size-change principle is PSPACE-hard, the dependency pair approach is only in NP if one uses standard estimation techniques for the dependency graph, argument filterings, and base orders in NP. Thus, although our experiments in Section 8 show that the dependency pair approach is more powerful for termination proofs of practical algorithms than the size-change principle, dependency pairs belong to a lower complexity class, provided that NP $\subsetneq$ PSPACE. (Nevertheless, these are only asymptotic worst-case complexities and indeed, in our experiments the dependency pair approach required significantly more runtime than the size-change principle, cf. Section 8.) Moreover, compared to a direct application of NP-complete base orders like LPO and $RPO(S)$, using them together with the dependency pair approach improves power significantly while the asymptotic worst-case complexity is not increased. To give a precise description of the complexity of the dependency pair approach, we prove that even if we are restricted to a simple reduction pair like the embedding order, the dependency pair approach is NP-hard (and thus, NP-complete).

In Section 7.2, we show that the complexity of the size-change principle is not increased when combining it with dependency pairs. In other words, with standard estimations of the dependency graph and base orders in PSPACE, the combined method is still in PSPACE although it is far more powerful than the size-change principle on its own. Moreover, we show that every method that is at least as powerful as the size-change principle is PSPACE-hard, which implies PSPACE-completeness of our combination technique. These results indicate that dependency pairs or the combination with dependency pairs are not only advantageous because of increase in power, but they are also advantageous as far as complexity is concerned.

7.1 Complexity of Dependency Pairs

We first show that the dependency pair technique (both for termination and innermost termination) is in NP if one uses standard approximations of the dependency graph, argument filterings, and a class of reduction pairs $\mathcal{RP}$ such that for any set of inequality constraints, satisfiability of the constraints by some reduction pair from $\mathcal{RP}$ is in NP. Examples for such classes are reduction pairs based on *LPO* and *RPO* (here this problem is NP-complete [20]) as well as the embedding order and *KBO* (here the problem is in P [19]). For (general) polynomial orders, the problem is undecidable. For the estimation of the (innermost) dependency graph, in the following theorem we use an (innermost) *dependency graph approximation algorithm* EST which, given two dependency pairs $s \to t$ and $v \to w$, can sometimes determine that these pairs do not form an (innermost) chain. So if EST returns "*no*", then $s \to t$, $v \to w$ is indeed no (innermost) chain. But if EST returns "*yes*", then this does not guarantee that $s \to t$, $v \to w$ is really an (innermost) chain.

One can estimate (innermost) dependency graphs by such an algorithm EST by drawing an edge from $s \to t$ and $v \to w$ whenever EST returns "*yes*". In this way, one indeed obtains a graph containing the real (innermost) dependency graph. The following theorem states that if EST is in NP (i.e., if its non-deterministic runtime is polynomial in the size of the TRS), then a termination proof using this estimated graph can also be performed in NP.

Theorem 27 (Dependency Pairs are in NP) *Let $\mathcal{RP}$ be a class of reduction pairs such that satisfiability of constraints by some reduction pair from the class is in NP. Moreover, let* EST *be an (innermost) dependency graph approximation algorithm in NP. If one estimates (innermost) dependency graphs by* EST *and if one is restricted to reduction pairs resulting from arbitrary argument filterings and pairs from $\mathcal{RP}$, then proving (innermost) termination by Theorem 19 is in NP.*

Proof For every term t, let $|t|$ be the size of t (i.e., the number of symbols) and let $n = \sum_{l \to r \in \mathcal{R}} |l| + |r|$ be the size of the TRS $\mathcal{R}$. We show that the non-deterministic complexity for an (innermost) termination proof with dependency pairs is polynomial in n.

From each rule $l \to r \in \mathcal{R}$ one obtains at most $|r|$ dependency pairs. So the overall number of dependency pairs is bounded by n and for each dependency pair $s \to t$ we also have $|s| + |t| \leq n$.

To compute the estimated (innermost) dependency graph, we have to check for all dependency pairs $s \to t$ and $v \to w$ whether $s \to t$ should be connected to $v \to w$. To this end we call the algorithm EST which is in NP. The input to EST is bounded by $3n$ ($2n$ for the two dependency pairs and another n for $\mathcal{R}$) and since we perform at most n^2 of these calls, the computation of the estimated graph can be done in (non-deterministic) polynomial time.

Once we have obtained the estimated graph, we can compute all SCCs (i.e., all maximal cycles) in linear time using a standard graph algorithm. Note that we do not compute every cycle in the graph, since there may be exponentially many. Instead, we use the result of [15] that it is sufficient to inspect SCCs repeatedly. As explained in Example 26, one first solves the constraints for an SCC and afterwards, one deletes all dependency pairs $s \to t$ from the graph where the strict constraints $s \succ t$ were satisfied. After the deletion one computes SCCs again and continues in this way. As there are at most n dependency pairs, we have at most n iterations with the cost of a (linear) SCC analysis and the treatment of a single cycle. So it only remains to show that handling one cycle can be done in NP.

In the innermost case, we first compute the set $\mathcal{U}(\mathcal{P})$ which can clearly be done in polynomial time. Next, for both termination and innermost termination proofs, we non-deterministically choose one dependency pair which we require to be strictly decreasing.[4] Then we choose the argument filtering non-deterministically in linear time. Finally, for the set of filtered constraints, satisfiability by some reduction pair of $\mathcal{RP}$ is in NP. $\square$

For the estimation techniques of the (innermost) dependency graph from [2,14,26], the algorithm EST to determine that two dependency pairs do not have to be connected runs in polynomial time. Thus, with these standard estimations of dependency graphs and base orders like LPO, $RPO(S)$, KBO, or the embedding order, by Theorem 27 the dependency pair approach is in NP.[5]

Next, we show that even if we are restricted to the embedding order, the dependency pair approach (using argument filterings) is NP-hard. We prove this result for any sound estimation of the (innermost) dependency graph which can at least detect that two dependency pairs $s \to t$ and $v \to w$ cannot be connected if t and v have different tuple symbols on their root positions. Together with the previous theorem this implies NP-completeness of dependency pairs.

Theorem 28 (Dependency Pairs are NP-hard) *Both the termination and the innermost termination technique of Theorem 19 are NP-hard if one uses reduction pairs based on argument filterings and the embedding order.*

Proof We give a reduction from the NP-complete problem 3-SAT. Let $\varphi = C_0 \wedge \ldots \wedge C_n$ be a formula in 3-conjunctive normal form over the variables $\{v_0, \ldots, v_m\}$. Every clause C_i has the form $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ where each of the literals $l_{i,1}, l_{i,2}, l_{i,3}$ is either a variable v_j or a negated variable $\neg v_j$ with $0 \le j \le m$. A formula φ is satisfiable iff there exists a variable assignment

[4] Alternatively, we could loop over all dependency pairs, since there can be at most n of them.

[5] The proof of Theorem 27 also reveals that if one uses a class of reduction pairs where satisfiability of constraints is in P, if EST is in P, and if one does not use argument filterings, then the dependency pair approach is in P as well.

$\tau : \{v_0, \dots, v_m\} \to \{\mathsf{true}, \mathsf{false}\}$ such that $\tau(\varphi)$ is equivalent to true using the standard semantics of $\vee$, $\wedge$, and $\neg$. In this case, we also write "$\tau(\varphi) = \mathsf{true}$".

We will present a polynomial-time translation of formulas φ into TRSs $\mathcal{R}_\varphi$ such that φ is satisfiable iff (innermost) termination of $\mathcal{R}_\varphi$ can be proved by the dependency pair approach with argument filterings and the embedding order. The idea is to define $\mathcal{R}_\varphi$ in such a way that there is a correspondence between variable assignments τ and argument filterings π: the formula φ is satisfied under the variable assignment τ iff (innermost) termination of $\mathcal{R}_\varphi$ can be proved using the corresponding argument filtering π.

Let $\mathcal{V}$ be a set of fresh variables and let $\mathcal{F}$ be a signature with the binary function symbols $\mathsf{v}_0, \dots, \mathsf{v}_m$ and a constant $\perp$. For any $x \in \mathcal{V}$, we first define a translation T_x from literals over $\{v_0, \dots, v_m\}$ to terms from $\mathcal{T}(\mathcal{F}, \mathcal{V})$.

$$
\begin{aligned}
T_x(v_j) &= \mathsf{v}_j(x, \perp)\\
T_x(\neg v_j) &= \mathsf{v}_j(\perp, x)
\end{aligned}
$$

Now we can define our transformation from formulas φ as above to TRSs $\mathcal{R}_\varphi$. For every clause C_i, we introduce two ternary symbols g_i and h_i and moreover, our signature $\mathcal{F}$ must contain an additional unary symbol s.

$$
\begin{aligned}
\mathcal{R}_\varphi = \{\mathsf{v}_j(x, x) \to x \mid 0 \le j \le m\} \;\cup \\
\left\{ \begin{array}{c} \mathsf{g}_i(\mathsf{s}(x_1), \mathsf{s}(x_2), \mathsf{s}(x_3)) \to \mathsf{h}_i(T_{x_1}(l_{i,1}), T_{x_2}(l_{i,2}), T_{x_3}(l_{i,3})) \\ \mathsf{h}_i(x, x, x) \to \mathsf{g}_{i+1 \bmod n+1}(x, x, x) \end{array} \;\middle|\; 0 \le i \le n \right\}
\end{aligned}
$$

Clearly, the transformation from φ to $\mathcal{R}_\varphi$ can be computed in polynomial time. It remains to show that (innermost) termination of $\mathcal{R}_\varphi$ can be proved by dependency pairs with argument filterings and the embedding order iff φ is satisfiable.

Thus, we now analyze the structure of a possible (innermost) termination proof of $\mathcal{R}_\varphi$. Due to the rules $\mathsf{v}_j(x, x) \to x$, the dependency graph and the innermost dependency graph have edges between the dependency pairs $\mathsf{G}_i(.) \to \mathsf{H}_i(.)$ and $\mathsf{H}_i(.) \to \mathsf{G}_{i+1 \bmod n+1}(.)$. The reason is that by the substitution σ that replaces every variable by $\perp$, we obtain $\sigma(\mathsf{H}_i(T_{x_1}(l_{i,1}), \dots, T_{x_3}(l_{i,3}))) = \mathsf{H}_i(\mathsf{v}_{i_1}(\perp, \perp), \dots, \mathsf{v}_{i_3}(\perp, \perp)) \xrightarrow{\mathrm{i}}{}^3 \mathsf{H}_i(\perp, \perp, \perp) = \sigma(\mathsf{H}_i(x, x, x))$ for some i_1, i_2, i_3 from $\{0, \dots, m\}$. Moreover, there is an edge from the dependency pair $\mathsf{H}_i(.) \to \mathsf{G}_{i+1 \bmod n+1}(.)$ to the pair $\mathsf{G}_{i+1 \bmod n+1}(.) \to \mathsf{H}_{i+1 \bmod n+1}(.)$. Thus, the TRS $\mathcal{R}_\varphi$ has the cycle $\mathcal{P} = \{\mathsf{G}_0(.) \to \mathsf{H}_0(.), \mathsf{H}_0(.) \to \mathsf{G}_1(.), \dots, \mathsf{H}_n(.) \to \mathsf{G}_0(.)\}$ in its (innermost) dependency graph. It is easy to see (and can be detected by every approximation algorithm for the (innermost) dependency graph that inspects at least the tuple symbols to estimate (innermost) chains) that there are no other cycles in the graph of $\mathcal{R}_\varphi$. Hence, the (innermost) termination proof of $\mathcal{R}_\varphi$ using Theorem 19 is equivalent to solving the constraints arising from $\mathcal{P}$. In the termination case, these have the following form if one uses an argument filtering π and the embedding order $\succ_{Emb}$ (where $\succ_{Emb} = \to^{+}_{Emb_{\mathcal{F}_\pi \cup \mathcal{F}^\sharp_\pi}}$ and $\succeq_{Emb} = \to^{*}_{Emb_{\mathcal{F}_\pi \cup \mathcal{F}^\sharp_\pi}}$).

$$\pi(\mathsf{v}_j(x,x)) \succeq_{Emb} \pi(x) \tag{9}$$
$$\pi(\mathsf{g}_i(\mathsf{s}(x_1),\mathsf{s}(x_2),\mathsf{s}(x_3))) \succeq_{Emb} \pi(\mathsf{h}_i(T_{x_1}(l_{i,1}),T_{x_2}(l_{i,2}),T_{x_3}(l_{i,3}))) \tag{10}$$
$$\pi(\mathsf{h}_i(x,x,x)) \succeq_{Emb} \pi(\mathsf{g}_{i+1 \bmod n+1}(x,x,x)) \tag{11}$$
$$\pi(\mathsf{G}_i(\mathsf{s}(x_1),\mathsf{s}(x_2),\mathsf{s}(x_3))) \succeq_{Emb} \pi(\mathsf{H}_i(T_{x_1}(l_{i,1}),T_{x_2}(l_{i,2}),T_{x_3}(l_{i,3}))) \tag{12}$$
$$\pi(\mathsf{H}_i(x,x,x)) \succeq_{Emb} \pi(\mathsf{G}_{i+1 \bmod n+1}(x,x,x)) \tag{13}$$

Moreover, one of the dependency pair constraints of the form (12) or (13) has to be strictly decreasing. In the innermost termination case, the constraints (10) and (11) are missing.

As we use the embedding order as base order, in a successful proof we have to filter away all symbols on the right-hand sides of the constraints that do not occur in the corresponding left-hand side. This implies that we have to use a collapsing argument filtering for the symbols H_i and G_i, i.e., $\pi(\mathsf{H}_i), \pi(\mathsf{G}_i) \in \{1,2,3\}$. In the termination case, we also have to use a collapsing filtering for h_i and g_i due to the constraints (10) and (11). Then the constraints (11) and (13) result in $x \succeq_{Emb} x$ which is obviously satisfied. This shows that one of the constraints (12) must be strict.

Since $\perp$ and the symbols v_j do not occur in the left-hand sides of (12), $\pi(\mathsf{H}_i(T_{x_1}(l_{i,1}),T_{x_2}(l_{i,2}),T_{x_3}(l_{i,3})))$ must not contain these symbols either. Hence, $\pi(\mathsf{H}_i(T_{x_1}(l_{i,1}),T_{x_2}(l_{i,2}),T_{x_3}(l_{i,3})))$ has to be a variable from x_1, x_2, x_3. This shows that $\pi(\mathsf{G}_i)$ and $\pi(\mathsf{H}_i)$ are identical, i.e.,

$$\pi(\mathsf{G}_i) = \pi(\mathsf{H}_i) = k_i \text{ with } k_i \in \{1,2,3\}. \tag{14}$$

Moreover, we must have

$$\pi(\mathsf{s}) = [1], \tag{15}$$

since otherwise, none of the constraints (12) would be strictly decreasing. Similar to (14), in the termination case we obtain the following requirements from (10):

$$\pi(\mathsf{g}_i) = \pi(\mathsf{h}_i) = k'_i \text{ with } k'_i \in \{1,2,3\}. \tag{16}$$

Now the constraints (12) have the form $\mathsf{s}(x_{k_i}) \succ_{Emb} \pi(T_{x_{k_i}}(l_{i,k_i}))$. This is equivalent to the requirement

$$\pi(T_{x_{k_i}}(l_{i,k_i})) = x_{k_i}. \tag{17}$$

Similarly, in the termination case, the constraints (10) are equivalent to

$$\pi(T_{x_{k'_i}}(l_{i,k'_i})) = x_{k'_i}. \tag{18}$$

Finally, the constraints (9) are equivalent to

$$\pi(\mathsf{v}_j) \neq [\,]. \tag{19}$$

Thus, termination of $\mathcal{R}_\varphi$ can be shown by dependency pairs with argument filterings and the embedding order iff there exists an argument filtering π

satisfying the requirements (14) – (19). Similarly, innermost termination can be shown if π satisfies the requirements (14), (15), (17), and (19). To conclude the proof of the theorem it remains to show that the existence of such an argument filtering is equivalent to satisfiability of φ.

For the "if" direction, let τ be a variable assignment with $\tau(\varphi) = \mathsf{true}$. We define $\pi(\mathsf{v}_j) = 1$ iff $\tau(v_j) = \mathsf{true}$ and $\pi(\mathsf{v}_j) = 2$ iff $\tau(v_j) = \mathsf{false}$. Thus, (19) is fulfilled and for any literal $l_{i,k}$, we obtain $\pi(T_x(l_{i,k})) = x$ iff $\tau(l_{i,k}) = \mathsf{true}$ and $\pi(T_x(l_{i,k})) = \bot$ iff $\tau(l_{i,k}) = \mathsf{false}$. For any clause $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$, we have $\tau(C_i) = \mathsf{true}$ and thus, there exists a k_i such that $\tau(l_{i,k_i}) = \mathsf{true}$. Hence, by defining $\pi(\mathsf{g}_i) = \pi(\mathsf{G}_i) = \pi(\mathsf{h}_i) = \pi(\mathsf{H}_i) = k_i \in \{1, 2, 3\}$, the conditions (14), (16), (17), and (18) are satisfied. Finally, we define $\pi(\mathsf{s}) = [1]$ to satisfy condition (15) as well.

For the "only if" direction, let π be an argument filtering satisfying at least the condition (17). For all variables v_j in the literals l_{i,k_i}, (17) implies $\pi(\mathsf{v}_j) \in \{1, 2\}$. If one defines a variable assignment τ with $\tau(v_j) = \mathsf{true}$ if $\pi(\mathsf{v}_j) = 1$ and $\tau(v_j) = \mathsf{false}$ if $\pi(\mathsf{v}_j) = 2$, then by the condition (17) we obtain $\tau(l_{i,k_i}) = \mathsf{true}$. Thus, every clause C_i contains a literal l_{i,k_i} which is true under the variable assignment τ and hence, we have $\tau(\varphi) = \mathsf{true}$. $\square$

7.2 Complexity of Combined Dependency Pairs and Size-Change Principle

We have shown that the dependency pair approach is NP-complete while the size-change principle is PSPACE-complete [24]. In Section 6 we presented a new technique to combine these two approaches and we proved in Theorems 23 and 25 that the combined approach is more powerful than both original techniques. We now show that the combination does not increase the complexity. In the combination technique for termination of a TRS $\mathcal{R}$ (Theorem 23), one may use a non-duplicating TRS $\mathcal{S}$ over the tuple symbols and the constructors of $\mathcal{R}$ to compare terms according to its rewrite relation $\to_{\mathcal{S}}^{+}$. In order to implement Theorem 23 in PSPACE, of course both the procedure to compute an appropriate TRS $\mathcal{S}$ and the decision procedure for the relation $\to_{\mathcal{S}}^{+}$ may only require space that is polynomial in the size of the TRS $\mathcal{R}$ to be proved terminating. For example, $\mathcal{S} = \mathit{Emb}_{\mathcal{C} \cup \mathcal{F}^{\sharp}}$ obviously satisfies this requirement.

Theorem 29 (Dep. Pairs & Size-Change Principle is in PSPACE) *Let $\mathcal{RP}$ be a class of reduction pairs such that satisfiability of constraints by some reduction pair from the class is in PSPACE. Moreover, let* EST *be an (innermost) dependency graph approximation algorithm in PSPACE. Finally, in the termination case, let* CON *be a PSPACE-algorithm that computes for a given TRS $\mathcal{R}$ another TRS $\mathcal{S}$ such that the signature of $\mathcal{S}$ are the tuple symbols and the constructors of $\mathcal{R}$, $\mathcal{S}$ is non-duplicating and terminating, and $\to_{\mathcal{S}}^{+}$ is decidable in PSPACE. If one estimates (innermost) dependency graphs by* EST*, if one is restricted to reduction pairs resulting from arbitrary argument filterings and pairs from $\mathcal{RP}$, and if one chooses* *the TRS $\mathcal{S} = \textsc{con}(\mathcal{R})$ in the termination case, then Theorems 23 and 25 can be implemented by PSPACE-algorithms.*

Proof We first regard the termination case (Theorem 23). To prove the termination of a TRS $\mathcal{R}$, we first compute $\mathcal{S} = \textsc{con}(\mathcal{R})$ in PSPACE. Thus, the size of the TRS $\mathcal{R} \cup \mathcal{S}$ is polynomial in the size of $\mathcal{R}$ (where we again define the *size* of $\mathcal{R}$ as $n = \sum_{l \to r \in \mathcal{R}} |l| + |r|$). As in the proof of Theorem 27, we can compute the estimated dependency graph of $\mathcal{R} \cup \mathcal{S}$ in PSPACE. In contrast to the proof of Theorem 27 where we regarded SCCs instead of cycles, now we can just iterate over all cycles of the estimated graph, since this iteration only requires polynomial space.

So it remains to show that for a single cycle $\mathcal{P}$, the conditions (a$'$) and (b$'$) of Theorem 23 can be checked in PSPACE. Because of the restrictions on the reduction pairs in $\mathcal{RP}$ and on the size of $\mathcal{S}$, checking $l \succsim r$ for all rules $l \to r \in \mathcal{R} \cup \mathcal{S}$ can be done in polynomial space. Thus, Condition (b$'$) can indeed be computed in PSPACE.

For Condition (a$'$), we first show that building the extended size-change graphs can be done in PSPACE. From each rule $l \to r \in \mathcal{R}$ one obtains at most $|r|$ extended size-change graphs. So the overall number of extended size-change graphs is bounded by n. In each extended size-change graph resulting from the rule $l \to r$, one has at most $|l| \times |r|$ arrows and a label of size $|l| + |r|$. Hence, each extended size-change graph has polynomial size. Finally, computing the arrows in the extended size-change graphs can be done in PSPACE due to the requirements on the reduction pairs in $\mathcal{RP}$ and on the relation $\to_{\mathcal{S}}^{+}$.

Finally, we have to show that one can decide in PSPACE whether every maximal multigraph labelled with the cycle $\mathcal{P}$ contains an edge $i \xrightarrow{\succ} i$. The naive approach of first computing all possible multigraphs (by building the transitive closure under concatenation "$\bullet$") and then inspecting all maximal multigraphs labelled with $\mathcal{P}$ cannot be done in PSPACE since there may be exponentially many multigraphs. More precisely, the number of possible multigraphs is $e := |\mathcal{P}|^2 \times 2^{|\mathcal{P}|} \times 3^{(ar+1)^2}$. Here, $|\mathcal{P}|$ is the number of dependency pairs in the cycle $\mathcal{P}$ and ar is the maximal arity of all function symbols in $\mathcal{R}$. The first part $|\mathcal{P}|^2 \times 2^{|\mathcal{P}|}$ describes the number of different labels in multigraphs. As in Section 6, we identify multigraphs with the labels $(D_1, \ldots, D_k)$ and $(D'_1, \ldots, D'_{k'})$ if $D_1 = D'_1$, $D_k = D'_{k'}$, and $\{D_1, \ldots, D_k\} = \{D'_1, \ldots, D'_{k'}\}$. Then the quadratic term $|\mathcal{P}|^2$ describes all possible choices for the leftmost and the rightmost dependency pair in the label. The exponential term $2^{|\mathcal{P}|}$ arises from all possible subsets of dependency pairs from $\mathcal{P}$ which may occur in the label. Finally, a multigraph can have up to $ar + 1$ nodes on each side and so there are $(ar + 1)^2$ possible combinations of output- and input-nodes. For each of these combinations we have the possibility to connect them by $\xrightarrow{\succ}$, by $\xrightarrow{\succsim}$, or by no edge. This leads to the base 3 in the factor $3^{(ar+1)^2}$.

To avoid the exponential space complexity of the above naive algorithm, we now give a non-deterministic PSPACE-algorithm that decides if there is a maximal multigraph labelled with $\mathcal{P}$ which does not contain an edge of the form $i \xrightarrow{\succsim} i$.

Input: A cycle $\mathcal{P}$ and a set Γ of extended size-change graphs labelled with dependency pairs from $\mathcal{P}$
Output: "false", if there is a maximal multigraph labelled with $\mathcal{P}$ in the transitive closure of Γ that has no edge $i \xrightarrow{\succsim} i$
"true", otherwise

1. $d := 0$
2. Choose a size-change graph $G \in \Gamma$
3. $e := |\mathcal{P}|^2 \times 2^{|\mathcal{P}|} \times 3^{(ar+1)^2}$
4. While $d < e$ do
   (a) $d := d + 1$
   (b) If G is maximal, labelled with $\mathcal{P}$, and contains no edge $i \xrightarrow{\succsim} i$, then stop and return "false"
   (c) Choose a size-change graph $G' \in \Gamma$
   (d) $G := G \bullet G'$
5. Return "true"

The algorithm can construct and inspect every multigraph that can be obtained by concatenating at most e extended size-change graphs, where these size-change graphs do not have to be distinct. We show by contradiction that any of the e possible different multigraphs can be constructed in this way. Assume that there is a multigraph G that can only be constructed by concatenating at least $e + k$ size-change graphs where $k > 0$. So $G = G_1 \bullet \ldots \bullet G_{e+k}$ with all $G_i \in \Gamma$. We define the graphs $G'_n = G_1 \bullet \ldots \bullet G_n$ for all $1 \leq n \leq e + k$. As there can only be at most e different multigraphs, there must be two graphs $G'_i, G'_j \in \{G'_n \mid 1 \leq n \leq e + k\}$ with $G'_i = G'_j$ and $i < j$. Hence, $G = G'_j \bullet G_{j+1} \bullet \ldots \bullet G_{e+k} = G'_i \bullet G_{j+1} \bullet \ldots \bullet G_{e+k}$. This is a contradiction to the assumption, since now G can be built by concatenating only $e + k - (j - i)$ size-change graphs. Hence, the above algorithm can indeed construct every possible multigraph, which proves its correctness.

For the space complexity of the algorithm, we look at the data that has to be stored during its execution. It mainly consists of the two numbers d and e, the input values $\mathcal{P}$ and Γ, and the multigraph G. Since e requires $O(\log e)$ space and since $\log e$ is polynomial in $|\mathcal{P}|$ and ar, it is easy to see that the algorithm only uses polynomial space.

Thus, we have a non-deterministic PSPACE-algorithm that decides if every maximal multigraph labelled with $\mathcal{P}$ contains an edge $i \xrightarrow{\succsim} i$. As NPSPACE = PSPACE [27], there also exists such a decision procedure in PSPACE.

The proof for the innermost termination case (Theorem 25) is analogous. Here, one needs the observations that usable rules can be computed in polynomial time (and thus, in polynomial space) and that the iteration over all subsets $\mathcal{D}'$ of $\mathcal{D}$ can be done in polynomial space. ☐

The size-change principle for functional programs is PSPACE-complete [24], even if one uses a simple underlying well-founded order like the embedding order on constructors. This result directly carries over to size-change termination for TRSs. So the methods of Theorem 9 for innermost termination proofs with the $\mathcal{C}$-restriction of the reduction pair $(\rightarrow^*_{Emb_\mathcal{C}}, \rightarrow^+_{Emb_\mathcal{C}})$ as well as the method of Theorem 12 for termination proofs with $\mathcal{S} = Emb_\mathcal{C}$ are PSPACE-complete as well. Furthermore, we now show that any sound technique that is at least as powerful as the size-change principle with the embedding order is PSPACE-hard. Here, we call a method "more powerful" if it can at least verify innermost termination for those TRSs where termination or innermost termination can be concluded from Theorem 12 using $\mathcal{S} = Emb_\mathcal{C}$ or from Theorem 9 using the $\mathcal{C}$-restriction of $(\rightarrow^*_{Emb_\mathcal{C}}, \rightarrow^+_{Emb_\mathcal{C}})$. Since the combination with dependency pairs yields a more powerful technique than the size-change principle by Theorems 23 and 25, this implies that this combination technique is also PSPACE-hard and thus, PSPACE-complete by Theorem 29.

Theorem 30 (Improving Size-Change Principle is PSPACE-hard) *Any sound technique for proving (innermost) termination of term rewriting is PSPACE-hard if it is at least as powerful as the size-change principle with the embedding order on constructors (i.e., if it can at least prove innermost termination of all TRSs which are size-change terminating w.r.t. the $\mathcal{C}$-restriction of $(\rightarrow^*_{Emb_\mathcal{C}}, \rightarrow^+_{Emb_\mathcal{C}})$).*

Proof We can use the same proof idea as in [24, Theorem 5] by reducing the PSPACE-complete problem of termination of boolean programs. In [24], a transformation is given which translates every boolean program B into a functional program (or TRS) P_B such that

- if B terminates, then P_B is not (innermost) terminating (20)
- if B does not terminate, then P_B is size-change terminating (21)

Here, a reduction pair is used which compares data objects by their size (i.e., this reduction pair corresponds to the $\mathcal{C}$-restriction of $(\rightarrow^*_{Emb_\mathcal{C}}, \rightarrow^+_{Emb_\mathcal{C}})$).

If the size-change principle is replaced by a stronger method, a statement analogous to (21) would obviously still hold. More precisely, if B does not terminate, then the stronger method can prove (innermost) termination of P_B. On the other hand, if the stronger method is sound then termination of B must lead to a TRS P_B that cannot be proved (innermost) terminating. Hence, B does not terminate iff the stronger method can prove (innermost) termination of P_B. □

In [23], two polynomial-time algorithms are presented that approximate size-change termination. So instead of the full size-change principle, these algorithms could be used in combination with dependency pairs to decrease the complexity. But of course, this would also decrease the power of the combined method.

The main idea of [23] is the identification of *anchors*: an anchor is a size-change graph which yields an infinite descent if it occurs infinitely often in a sequence $G_1 \circ G_2 \circ \ldots$ of size-change graphs. In other words, G is an anchor iff for any graph $G_1 \circ G_2 \circ \ldots$ with $G_i = G$ for infinitely many i, there is a path where infinitely many edges are labelled with "$\succ$".

Since size-change termination is not affected when deleting anchors, the approach of [23] repeatedly tries to identify and to remove anchors. If finally all size-change graphs have been deleted, then one can conclude size-change termination. However, since [23] only uses sufficient criteria to find anchors, not all size-change terminating systems can be detected by this approach.

The two algorithms in [23] differ in their criterion to identify anchors. The first algorithm has a runtime of $O(n^2)$ (where n is the size of the program or TRS). However, this approach is not useful in combination with dependency pairs: it can be shown that if this algorithm succeeds then dependency pairs on their own can prove termination using a reduction pair based on linear polynomial interpretations with coefficients from $\{0, 1\}$.

This is not the case for the second algorithm which has a runtime of $O(n^3)$. However, our experiments in Section 8.2 show that in practice, the size-change principle is not the bottleneck of the combined method. Therefore, in the combination with dependency pairs, we use the full size-change principle instead of this (weaker) approximation.

8 Implementation and Experiments

We developed a system AProVE (Automated Program Verification Environment) for mechanized verification of functional programs and TRSs which is available from `http://www-i2.informatik.rwth-aachen.de/AProVE`. To perform automated (innermost) termination proofs, the system offers *LPO*, *RPO*(S), *KBO*, *polynomial orders*, and *dependency pairs*. The tool is written in Java and termination proofs can be performed via a graphical user interface. For a description of the system, the reader is referred to [13].

To evaluate the results developed in the paper, we also integrated the size-change principle and our technique to combine dependency pairs with the size-change principle into the system. For the (pure) size-change principle, we implemented the criterion of Theorem 12 using a reduction pair based on the embedding order. The implementation of our combination technique for both termination (Theorem 23 and Corollary 24) and innermost termination (Theorem 25) is described in Section 8.1. Subsequently, in Section 8.2, we give an empirical evaluation in order to compare the size-change principle, dependency pairs, and the combination of both techniques.

8.1 Implementing the Size-Change- and Dependency Pair-Combination

We first present our algorithm to verify innermost termination of a TRS $\mathcal{R}$ with defined symbols $\mathcal{D}$ according to Theorem 25 and give a detailed explanation afterwards:

1. Compute the estimated innermost dependency graph of $\mathcal{R}$.
2. For each SCC $\mathcal{P}$ in the graph:
   2.1. Let $\mathcal{C}_\mathcal{P}$ be the set of the constructors occurring in $\mathcal{P}$,
   let $\mathcal{D}_\mathcal{P}$ be a subset of the defined symbols
   occurring in right-hand sides of $\mathcal{P}$,
   let π be an argument filtering
   which only filters symbols from $\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}$.
   If all such $\mathcal{D}_\mathcal{P}$ and argument filterings π have already
   been examined without success,
   then abort with "`No Success`".
   2.2. Let $s \succ t$ iff $t \in \mathcal{T}(\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}, \mathcal{V})$ and $\pi(s) \to^+_{Emb_{\mathcal{F}_\pi}} \pi(t)$.
   Let $s \succsim t$ iff $t \in \mathcal{T}(\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}, \mathcal{V})$ and $\pi(s) \to^*_{Emb_{\mathcal{F}_\pi}} \pi(t)$.
   2.3. Try to show that all maximal multigraphs w.r.t. $(\succsim, \succ)$
   contain an edge $i \xrightarrow{\succ} i$.
   2.4. If Step 2.3 fails, then go to Step 2.1 and examine
   the next argument filtering π resp. the next subset $\mathcal{D}_\mathcal{P}$.
   2.5. Otherwise, let $\mathcal{D}'$ consist of the defined symbols of $\mathcal{U}(\mathcal{D}_\mathcal{P})$.
   Try to extend π to an argument filtering on $\mathcal{F}$
   and try to find a reduction pair $(\succsim', \succ')$ with (quasi-)
   simplification orders $\succsim', \succ' \subseteq \mathcal{T}(\mathcal{F}_\pi, \mathcal{V}) \times \mathcal{T}(\mathcal{F}_\pi, \mathcal{V})$
   such that $\pi(l) \succsim' \pi(r)$ for all $l \to r \in \mathcal{U}(\mathcal{D}')$.
   2.6. If Step 2.5 fails, then go to Step 2.1 and examine
   the next argument filtering π resp. the next subset $\mathcal{D}_\mathcal{P}$.
   Otherwise, continue with the next SCC $\mathcal{P}$ in Step 2.
3. Finish with "`Innermost Termination Proved`".

For reasons of efficiency, in our implementation we did not extend size-change graphs by nodes labelled with ε, cf. Definition 21. These nodes would be necessary to simulate dependency pairs with the combined technique. Thus, if our implementation of the combined technique fails, then it can still be useful to try an innermost termination proof with dependency pairs.

Moreover, instead of labelling multigraphs by sequences $(D_1, \ldots, D_k)$ of dependency pairs, in our implementation we only label them by the first and the last pair in the sequence. In this way, many former multigraphs are identified, i.e., we obtain significantly less multigraphs which increases efficiency.

With this representation of the labels, it suffices only to regard the initial SCCs instead of cycles of the estimated innermost dependency graph. This improves efficiency even further, since there are typically far less initial SCCs than cycles.[6] Note that when examining only the initial SCCs $\mathcal{P}$, it is no longer sufficient just to regard maximal multigraphs labelled with $\mathcal{P}$ in

[6] When implementing the pure dependency pair approach, instead of inspecting all cycles, it is preferable to use the technique of [15] to compute new SCCs repeatedly from weakly decreasing dependency pairs (see the proof of Theorem 27). However, this technique of [15] cannot be adapted to the combination of dependency pairs with the size-change principle.

Theorem 25 (c′). Instead, one has to investigate all maximal multigraphs. In other words, one also has to regard multigraphs whose label only consists of a subset of $\mathcal{P}$ (i.e., of the dependency pairs from an arbitrary subcycle). For all these maximal multigraphs one has to check whether they contain an edge of the form $i \xrightarrow{\succsim} i$. However, this is already taken into account in our implementation, since we represent labels $(D_1, \ldots, D_k)$ by only storing their first and their last pair. Thus, we do not check anymore whether $\{D_1, \ldots, D_k\}$ contains all pairs from the SCC $\mathcal{P}$. Instead, now the labels are only used to determine which multigraphs may be concatenated and for this purpose, one only has to know D_1 and D_k.

As in Theorem 25, we only regard a reduction pair on a subset $\mathcal{D}'$ of the defined symbols. To this end, we choose a subset $\mathcal{D}_\mathcal{P}$ of the defined symbols in right-hand sides of $\mathcal{P}$'s dependency pairs and define $\mathcal{D}'$ to consist of all defined symbols of the TRS $\mathcal{U}(\mathcal{D}_\mathcal{P})$. The motivation for this is as follows: if two terms s and t have to be compared when computing the edges of size-change graphs, then t can only contain defined symbols which occur in right-hand sides of $\mathcal{P}$'s dependency pairs.[7] Moreover, the set $\mathcal{D}'$ contains all defined symbols which occur in the right-hand sides of the usable rules, since they have to be oriented according to Theorem 25 (d′).

Now we have to generate a suitable monotonic reduction pair. As in the dependency pair approach, we use argument filterings π in combination with simplification orders on $\mathcal{T}(\mathcal{F}_\pi, \mathcal{V}) \times \mathcal{T}(\mathcal{F}_\pi, \mathcal{V})$ for this purpose. (Different from Theorem 25, we do not compare terms with tuple symbols from $\mathcal{F}^\sharp$, since we do not regard nodes labelled with ε.) When computing size-change graphs, we already fix a part of the argument filtering, viz., we determine how π operates on function symbols from $\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}^{\text{left}}$. Here, $\mathcal{D}_\mathcal{P}^{\text{left}}$ are the defined symbols occurring on left-hand sides in $\mathcal{P}$. For $\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}$, the argument filtering is chosen in Step 2.1 and the symbols in $\mathcal{D}_\mathcal{P}^{\text{left}} \setminus \mathcal{D}_\mathcal{P}$ are not filtered. But we do not yet fix π on $\mathcal{F} \setminus (\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}^{\text{left}})$, since these symbols are not compared when computing edges of size-change graphs. Moreover, at this point, we still leave the simplification order open. Thus, for the size-change graphs we use a reduction pair $(\succsim, \succ)$ where $s \succ t$ holds iff $t \in \mathcal{T}(\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}, \mathcal{V})$ and $\pi(s) \to^+_{Emb_{\mathcal{F}_\pi}} \pi(t)$. Moreover, $s \succsim t$ iff $t \in \mathcal{T}(\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}, \mathcal{V})$ and $\pi(s) \to^*_{Emb_{\mathcal{F}_\pi}} \pi(t)$. The reason for only using the embedding order when comparing the arguments in the size-change graphs is efficiency. More sophisticated orders have several parameters (e.g., status and precedence in $RPOS$). When using such orders for ordinary termination proofs (possibly with dependency pairs), these parameters are determined incrementally. However, it is not clear how to transfer such an incremental approach to the size-change principle, since one would have to draw conclu-

[7] Defined symbols that only occur on left-hand sides of dependency pairs or usable rules do not have to be included in $\mathcal{D}'$, since the $(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp)$-restriction is a relation from $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V}) \times \mathcal{T}(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp, \mathcal{V})$, i.e., the "greater" term may be from the full signature $\mathcal{F} \cup \mathcal{F}^\sharp$.

sions from an unsuccessful size-change analysis to modify the parameters of the order (e.g., by extending the precedence).

After computing the size-change graphs we have to calculate the maximal multigraphs and check whether all of them have an edge of the form $i \xrightarrow{\succsim} i$. In case of success, the current reduction pair is refined. To this end, π is also determined on the remaining symbols from $\mathcal{F} \setminus (\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}^{\mathrm{left}})$ and the reduction pair $(\succsim, \succ)$ is refined such that $s \succsim t$ iff $\pi(s) \succsim' \pi(t)$ and $s \succ t$ iff $\pi(s) \succ' \pi(t)$ for some quasi-simplification order $\succsim'$ and simplification order $\succ'$ on $\mathcal{T}(\mathcal{F}_\pi, \mathcal{V}) \times \mathcal{T}(\mathcal{F}_\pi, \mathcal{V})$.[8] Note that the relations used for computing the size-change graphs are indeed contained in these refined relations since $\pi(s) \rightarrow_{Emb_{\mathcal{F}_\pi}} \pi(t)$ implies $\pi(s) \succsim' \pi(t)$ and $\pi(s) \succ' \pi(t)$. The reason is that any (quasi-)simplification order contains the embedding order. As $(\succsim', \succ')$ is a reduction pair and as $\succsim'$ is monotonic, $(\succsim, \succ)$ is a monotonic reduction pair, too. Since the size-change graphs were computed with a subset of the final refined reduction pair $(\succsim, \succ)$, some edges in the graphs may be missing, but this only affects the power, not the soundness of the approach. Hence, building the size-change graphs with the embedding order instead of other efficient orders has the advantage that the size-change graphs are correct w.r.t. any simplification order. So one may indeed use any (quasi-) simplification order when orienting the usable rules. For example, in our experiments in Section 8.2 we used *LPO*.

We also implemented a `hybrid` variant of the above combination algorithm. Here, if Step 2.1 returns "`No Success`", then we try to solve the constraints resulting from the original dependency pair approach. If this succeeds, then we continue with the next SCC in the hybrid algorithm. Otherwise we return a final "`No Success`".

The combination algorithm for termination is like the innermost termination algorithm except for two differences: In Step 2.1 we always let $\mathcal{D}_\mathcal{P}$ consist of all defined symbols occurring in $\mathcal{P}$ and in Step 2.5 we have to analyze all rules $l \to r \in \mathcal{R}$ instead of just those of $\mathcal{U}(\mathcal{D}')$. This implementation of Theorem 23 always chooses $\mathcal{S} = \varnothing$, i.e., we use Corollary 24 for the automation.

8.2 Empirical Evaluation

Now we describe our experiments to evaluate the performance of the three approaches discussed in the paper (size-change principle, dependency pairs, and the combination of the two techniques). To this end, we tested our

[8] The relation $\succ'$ itself is not needed in the implementation. Instead, one only has to determine a quasi-simplification order $\succsim'$ which forms a reduction pair with some simplification order. The reason is that the reduction pair $(\succsim', \succ')$ is only needed to ensure that the usable rules are weakly decreasing w.r.t. the $(\mathcal{C} \cup \mathcal{D}')$-restriction of $\succsim$ (where $\succsim = \succsim'_\pi$). To this end, it suffices to require $\pi(l) \succsim' \pi(r)$ for all $l \to r \in \mathcal{U}(\mathcal{D}')$ in Step 2.5 since r only contains defined symbols from $\mathcal{D}'$ by the construction of $\mathcal{D}'$.

implementation on the large collection of examples from [3, 7, 28] (108 TRSs for termination, 151 TRSs for innermost termination). More precisely, we used the following **algorithms** for (innermost) termination proofs:

– ***SCP*** is the size-change principle for TRSs according to Theorem 12.
– ***DP*** is the original dependency pair approach.
– ***DP_SCP*** is the combination of dependency pairs and the size-change principle as described in Section 8.1. To increase efficiency, we only tried sets $\mathcal{D}_\mathcal{P}$ with $|\mathcal{D}_\mathcal{P}| \leq 2$ in the algorithm for innermost termination and we only allowed an argument filtering of at most two function symbols in Step 2.1 (i.e., when building size-change graphs). Later, when orienting the rules in Step 2.5, we permitted arbitrary filterings on $\mathcal{F} \setminus (\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}^{\text{left}})$.
– ***H_DP_SCP*** is the hybrid version of *DP_SCP*.

In the experiments, we used the following base **orders** (or reduction pairs).

– ***EMB*** is the embedding order.
– ***LPO*** is the lexicographic path order where arguments are compared lexicographically from left to right. The required precedence is determined automatically and different symbols may be equal in precedence.

Algorithm	Order	Power		Time	Avg. Time
Termination (108 examples)					
SCP	*EMB*	22	[20.4 %]	0.1s	[0.0 s, 0.0 s, 0.0 s]
DP	*EMB*	40	[37.0 %]	19.1 s	[0.2 s, 0.0 s, 0.3 s]
DP_SCP	*EMB*	46	[42.6 %]	19.3 s	[0.2 s, 0.1 s, 0.2 s]
DP	*LPO*	68	[63.0 %]	144.9 s	[1.3 s, 0.3 s, 3.1 s]
DP_SCP	*LPO*	69	[63.9 %]	21.4 s	[0.2 s, 0.1 s, 0.3 s]
H_DP_SCP	*LPO*	73	[67.6 %]	93.1 s	[0.9 s, 0.2 s, 2.3 s]
Innermost Termination (151 examples)					
SCP	*EMB*	22	[14.6 %]	0.0 s	[0.0 s, 0.0 s, 0.0 s]
DP	*EMB*	77	[51.0 %]	28.3 s	[0.2 s, 0.0 s, 0.3 s]
DP_SCP	*EMB*	87	[57.6 %]	25.6 s	[0.2 s, 0.1 s, 0.3 s]
DP	*LPO*	98	[64.9 %]	204.0 s	[1.4 s, 0.2 s, 3.5 s]
DP_SCP	*LPO*	101	[66.9 %]	35.3 s	[0.2 s, 0.2 s, 0.2 s]
H_DP_SCP	*LPO*	106	[70.2 %]	113.5 s	[0.8 s, 0.3 s, 1.9 s]

Table 1 Performance of the Different Techniques on the Examples of [3, 7, 28].

In the "power" column, Table 1 shows the number and percentage of examples where the respective approach was successful. Here, proofs were interrupted after 30 seconds.[9] In the "time" column, it shows the time required for the proof attempts. Moreover, in square brackets we give the

[9] There are five examples where some of the algorithms and orders in Table 1 resulted in timeouts.

average time needed per example, the average time for examples where the proof succeeds, and the average time for examples where the proof fails. For further details (e.g., individual runtimes and results for each example) the reader is referred to [29]. Our experiments were performed on a Pentium IV with 2.4 GHz and 1 GB memory.

The first rows for the *SCP* technique indicate clearly that the size-change principle on its own has only very limited power. Comparing the dependency pair approach with the combined technique leads to two observations. If one uses weak but very efficient orders like *EMB*, then the main benefit is power. The combined technique can show (innermost) termination for at least 13 % more examples in about the same time. For more powerful orders like *LPO* the main advantage of the combined technique is efficiency, since here, the combination is approximately 6 times faster while power is increased slightly. Note that this increase in efficiency is indeed due to our combination technique (and not just due to the heuristic restrictions on $\mathcal{D}_\mathcal{P}$ and on the argument filterings in the *DP_SCP*-algorithm): even without these heuristic restrictions (i.e., even if one regards all possible subsets $\mathcal{D}_\mathcal{P}$ and all possible argument filterings when building size-change graphs), the combination of dependency pairs and the size-change principle is still at least three times faster than the pure dependency pair approach. Finally, with the hybrid algorithm and *LPO*, runtimes are decreased by 35 – 45 % compared to the dependency pair technique and we can show (innermost) termination of at least 7 % more examples.

The dependency pair technique as described in Section 5 can be improved in several ways (e.g., by transforming dependency pairs by narrowing, rewriting, and instantiation [2,10,12], by reducing the set of constraints in Theorem 19 (b) and (d) [12,16,31], by removing rules that do not influence the termination behavior [31], etc.). All these refinements also carry over to the new technique which combines the size-change principle with dependency pairs. However, in order to measure the effects of the combination, in our experiments we used the "pure" dependency pair method and disabled all these improvements. As shown in [29], the combination with the size-change principle also yields similar advantages if improvements like dependency pair transformations are enabled.

We also analyzed the examples where the hybrid algorithm still fails if one uses dependency pair transformations. It turns out that then the failure is mainly due to the underlying reduction pairs. If one uses other reduction pairs based on *KBO*, polynomial interpretations, and *RPOS*, we can apply our technique successfully on almost all examples. These results show that the techniques presented in the paper are indeed very powerful for mechanized (innermost) termination proofs.

9 Conclusion

In this paper, we adapted the size-change principle from functional programs to arbitrary TRSs and developed a technique to use it for possibly automated (innermost) termination proofs of TRSs. Then we compared this principle with classical simplification orders from term rewriting. We showed that it is also restricted to proving simple termination, it incorporates lexicographic and multiset comparison for root symbols (although not below the root), but it cannot handle defined symbols or term measures and weights. Nevertheless, there are even examples where the size-change principle is advantageous to dependency pairs, since it can simulate argument filtering for root symbols and it can investigate how the size of arguments changes in sequences of subsequent function calls. On the other hand, the size-change principle is not modular and it lacks a concept like the dependency graph to analyze which function calls can follow each other. For that reason, we developed a new approach which combines the size-change principle with dependency pairs. This combined approach is more powerful than both previous techniques and it has the advantage that it often succeeds with much simpler base orders than the dependency pair approach. We analyzed the complexity of the dependency pair approach and of the new technique combining dependency pairs with the size-change principle. While the size-change principle is PSPACE-complete, the dependency pair approach is only NP-complete. The combination with dependency pairs does not increase the complexity of the size-change principle, i.e., it is still PSPACE-complete.

We have implemented both the original dependency pair approach and the combined approach in the system AProVE and found that this combination often increases efficiency dramatically. With this combination (using the refinement of dependency pair transformations) and with an underlying reduction pair based on the lexicographic path order, 137 of the 151 examples in the collections of [3, 7, 28] could be proved innermost terminating fully automatically. Most of these proofs took less than a second and the longest proof took about 11 seconds. When regarding termination, the proof succeeded for 87 of the 108 examples and the longest proof took 4.7 seconds.

Acknowledgements. We are grateful to the referees for many helpful suggestions and remarks.

References

1. H. Anderson and S. C. Khoo. Affine-based size-change termination. In *Proc. 1st APLAS*, LNCS 2895, pages 122–140, 2003.
2. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
3. T. Arts and J. Giesl. A collection of examples for termination of term rewriting using dependency pairs. Technical Report AIB-2001-09, RWTH Aachen, Germany, 2001. Available from `http://aib.informatik.rwth-aachen.de`.
4. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
5. C. Borralleras, M. Ferreira, and A. Rubio. Complete monotonic semantic path orderings. In *Proc. 17th CADE*, LNAI 1831, pages 346–364, 2000.

6. N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.
7. N. Dershowitz. 33 examples of termination. In *Proc. French Spring School of Theoretical Computer Science*, LNCS 909, pages 16–26, 1995.
8. N. Dershowitz, N. Lindenstrauss, Y. Sagiv, and A. Serebrenik. A general framework for automatic termination analysis of logic programs. *Applicable Algebra in Engineering, Communication and Computing*, 12(1,2):117–156, 2001.
9. O. Fissore, I. Gnaedig, and H. Kirchner. Cariboo: An induction based proof tool for termination with strategies. In *Proc. 4th PPDP*, pages 62–73. ACM Press, 2002.
10. J. Giesl and T. Arts. Verification of Erlang processes by dependency pairs. *Applicable Algebra in Engineering, Communication and Computing*, 12(1,2):39–72, 2001.
11. J. Giesl, T. Arts, and E. Ohlebusch. Modular termination proofs for rewriting using dependency pairs. *Journal of Symbolic Computation*, 34(1):21–58, 2002.
12. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Improving dependency pairs. In *Proc. 10th LPAR*, LNAI 2850, pages 165–179, 2003.
13. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated termination proofs with AProVE. In *Proc. 15th RTA*, LNCS 3091, pages 210–220, 2004.
14. N. Hirokawa and A. Middeldorp. Approximating dependency graphs without using tree automata techniques. In *Proc. 6th WST*, 2003.
15. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. In *Proc. 19th CADE*, LNAI 2741, pages 32–46, 2003. Full version to appear in *Information and Computation*.
16. N. Hirokawa and A. Middeldorp. Dependency pairs revisited. In *Proc. 15th RTA*, LNCS 3091, pages 249–268, 2004.
17. S. Kamin and J. J. Lévy. Two generalizations of the recursive path ordering. Unpublished Manuscript, University of Illinois, IL, USA, 1980.
18. D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.
19. K. Korovin and A. Voronkov. Orienting rewrite rules with the Knuth-Bendix order. *Information and Computation*, 183:165–186, 2003.
20. M. S. Krishnamoorthy and P. Narendran. On recursive path ordering. *Theoretical Computer Science*, 40:323–328, 1985.
21. K. Kusakari, M. Nakamura, and Y. Toyama. Argument filtering transformation. In *Proc. 1st PPDP*, LNCS 1702, pages 48–62, 1999.
22. D. Lankford. On proving term rewriting systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.
23. C. S. Lee. Program termination analysis in polynomial time. In *Proc. 1st GPCE*, LNCS 2487, pages 218–235, 2002.
24. C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In *Proc. 28th POPL*, pages 81–92, 2001.
25. Z. Manna and S. Ness. On the termination of Markov algorithms. In *Proc. 3rd Hawaii International Conference on System Science*, pages 789–792, 1970.
26. A. Middeldorp. Approximations for strategies and termination. In *Proc. WRS '02*, ENTCS 70(6), 2002.
27. J. W. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

28. J. Steinbach. Automatic termination proofs with transformation orderings. In *Proc. 6th RTA*, LNCS 914, pages 11–25, 1995. Full version appeared as Technical Report SR-92-23, Universität Kaiserslautern, Germany.
29. R. Thiemann and J. Giesl. Empirical evaluation of the size-change principle for term rewriting. Available from `http://www-i2.informatik.rwth-aachen.de/AProVE/empiricalSCP.ps`.
30. R. Thiemann and J. Giesl. Size-change termination for term rewriting. In *Proc. 14th RTA*, LNCS 2706, pages 264–278, 2003.
31. R. Thiemann, J. Giesl, and P. Schneider-Kamp. Improved modular termination proofs using dependency pairs. In *Proc. 2nd IJCAR*, LNAI 3097, pages 75–90, 2004.
32. Y. Toyama. Counterexamples to the termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, 1987.