Improved Modular Termination Proofs Using Dependency Pairs

René Thiemann, Jürgen Giesl, Peter Schneider-Kamp

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

Abstract. The dependency pair approach is one of the most powerful techniques for automated (innermost) termination proofs of term rewrite systems (TRSs). For any TRS, it generates inequality constraints that have to be satisfied by well-founded orders. However, proving *innermost* termination is considerably easier than termination, since the constraints for innermost termination are a subset of those for termination.
We show that surprisingly, the dependency pair approach for termination can be improved by only generating the same constraints as for innermost termination. In other words, proving full termination becomes virtually as easy as proving innermost termination. Our results are based on splitting the termination proof into several modular independent subproofs. We implemented our contributions in the automated termination prover AProVE and evaluated them on large collections of examples. These experiments show that our improvements increase the power and efficiency of automated termination proving substantially.

1 Introduction

Most traditional methods for automated termination proofs of TRSs use *simplification orders* [7, 26], where a term is greater than its proper subterms (*subterm property*). However, there are numerous important TRSs which are not *simply terminating*, i.e., termination cannot be shown by simplification orders. Therefore, the *dependency pair* approach [2, 10, 11] was developed which considerably increases the class of systems where termination is provable mechanically.

Example 1. The following variant of an example from [2] is not simply terminating, since $\mathsf{quot}(x, \mathsf{0}, \mathsf{s}(\mathsf{0}))$ reduces to $\mathsf{s}(\mathsf{quot}(x, \mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0})))$ in which it is embedded. Here, $\mathsf{div}(x, y)$ computes $\lfloor \frac{x}{y} \rfloor$ for $x, y \in \mathbb{N}$ if $y \neq 0$. The auxiliary function $\mathsf{quot}(x, y, z)$ computes $1 + \lfloor \frac{x-y}{z} \rfloor$ if $x \geq y$ and $z \neq 0$ and it computes 0 if $x < y$.

$$\mathsf{div}(\mathsf{0}, y) \to \mathsf{0} \quad (1)$$
$$\mathsf{div}(x, y) \to \mathsf{quot}(x, y, y) \quad (2)$$
$$\mathsf{quot}(\mathsf{0}, \mathsf{s}(y), z) \to \mathsf{0} \quad (3)$$
$$\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y), z) \to \mathsf{quot}(x, y, z) \quad (4)$$
$$\mathsf{quot}(x, \mathsf{0}, \mathsf{s}(z)) \to \mathsf{s}(\mathsf{div}(x, \mathsf{s}(z))) \quad (5)$$

In Sect. 2, we recapitulate dependency pairs. Sect. 3 proves that for termination, it suffices to require only the same constraints as for innermost termination.

This result is based on a refinement for termination proofs with dependency pairs by Urbain [29], but it improves upon this and related refinements [12, 24] significantly. In Sect. 4 we show that the new technique of [12] to reduce the constraints for innermost termination by integrating the concepts of "argument filtering" and "usable rules" can also be adapted for termination proofs. Finally, based on the improvements presented before, Sect. 5 introduces a new method to remove rules of the TRS which reduces the set of constraints even further.

In each section, we demonstrate the power of the respective refinement by examples where termination can now be shown, while they could not be handled before. Our results are implemented in the automated termination prover AProVE [14]. The experiments in Sect. 6 show that our contributions increase power and efficiency on large collections of examples. Thus, our results are also helpful for other tools based on dependency pairs ([1], CiME [6], TTT [19]) and we conjecture that they can also be used in other recent approaches for termination of TRSs [5, 9, 27] which have several aspects in common with dependency pairs.

2 Modular Termination Proofs Using Dependency Pairs

We briefly present the *dependency pair* approach of Arts & Giesl and refer to [2, 10–12] for refinements and motivations. We assume familiarity with term rewriting (see, e.g., [4]). For a TRS $\mathcal{R}$ over a signature $\mathcal{F}$, the *defined symbols* $\mathcal{D}$ are the roots of the left-hand sides of rules and the *constructors* are $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$. We restrict ourselves to finite signatures and TRSs. The infinite set of variables is denoted by $\mathcal{V}$ and $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is the set of all terms over $\mathcal{F}$ and $\mathcal{V}$. 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 $f^\sharp$. If $t = g(t_1, \ldots, t_m)$ with $g \in \mathcal{D}$, we write $t^\sharp$ for $g^\sharp(t_1, \ldots, t_m)$.

Definition 2 (Dependency Pair). *The set of* dependency pairs *for a TRS* $\mathcal{R}$ *is* $DP(\mathcal{R}) = \{l^\sharp \to t^\sharp \mid l \to r \in \mathcal{R},\ t \text{ is a subterm of } r \text{ with } \mathrm{root}(t) \in \mathcal{D}\}$.

So the dependency pairs of the TRS in Ex. 1 are

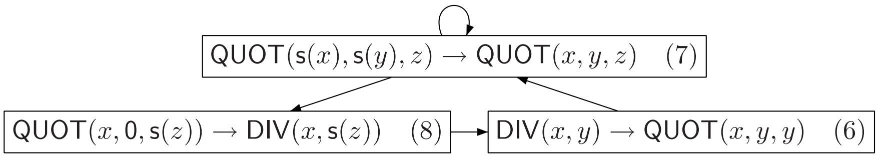
$$\mathsf{DIV}(x, y) \to \mathsf{QUOT}(x, y, y) \quad (6) \qquad \mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y), z) \to \mathsf{QUOT}(x, y, z) \quad (7)$$

$$\mathsf{QUOT}(x, \mathsf{0}, \mathsf{s}(z)) \to \mathsf{DIV}(x, \mathsf{s}(z)) \quad (8)$$

For (innermost) termination, we need the notion of (innermost) *chains*. Intuitively, a dependency pair corresponds to a (possibly recursive) function call and a chain represents possible sequences of calls that can occur during a reduction. We always assume that different occurrences of dependency pairs are variable disjoint and consider substitutions whose domains may be infinite. Here, $\xrightarrow{\text{i}}_{\mathcal{R}}$ denotes innermost reductions where one only contracts innermost redexes.

Definition 3 (Chain). *Let $\mathcal{P}$ be a set of pairs of terms. A (possibly infinite) sequence of pairs $s_1 \to t_1, s_2 \to t_2, \ldots$ from $\mathcal{P}$ is a $\mathcal{P}$-*chain over the TRS $\mathcal{R}$ *iff there is a substitution σ with $t_i\sigma \to_{\mathcal{R}}^* s_{i+1}\sigma$ for all i. The chain is an* innermost chain *iff $t_i\sigma \xrightarrow{\text{i}}{}_{\mathcal{R}}^* s_{i+1}\sigma$ and all $s_i\sigma$ are in normal form. An (innermost) chain is* minimal *iff all $s_i\sigma$ and $t_i\sigma$ are (innermost) terminating w.r.t. $\mathcal{R}$.*

To determine which pairs can follow each other in chains, one builds an *(innermost) dependency graph*. Its nodes are the dependency pairs and there is an arc from $s \to t$ to $u \to v$ iff $s \to t, u \to v$ is an (innermost) chain. Hence, every infinite chain corresponds to a cycle in the graph. In Ex. 1 we obtain the following graph with the cycles $\{(7)\}$ and $\{(6), (7), (8)\}$. Since it is undecidable whether two dependency pairs form an (innermost) chain, for automation one constructs *estimated* graphs containing the real dependency graph (see e.g., [2, 18]).[1]



Theorem 4 (Termination Criterion [2]). *A TRS $\mathcal{R}$ is (innermost) terminating iff for every cycle $\mathcal{P}$ of the (innermost) dependency graph, there is no infinite minimal (innermost) $\mathcal{P}$-chain over $\mathcal{R}$.*

To automate Thm. 4, for each cycle one generates constraints which should be satisfied by a *reduction pair* $(\succsim, \succ)$ where $\succsim$ is reflexive, transitive, monotonic and stable (closed under contexts and substitutions) and $\succ$ is a stable well-founded order compatible with $\succsim$ (i.e., $\succsim \circ \succ \subseteq \succ$ and $\succ \circ \succsim \subseteq \succ$). But $\succ$ need not be monotonic. The constraints ensure that at least one dependency pair is strictly decreasing (w.r.t. $\succ$) and all remaining pairs and all rules are weakly decreasing (w.r.t. $\succsim$). Requiring $l \succsim r$ for all $l \to r \in \mathcal{R}$ ensures that in chains $s_1 \to t_1, s_2 \to t_2, \ldots$ with $t_i\sigma \to_{\mathcal{R}}^* s_{i+1}\sigma$, we have $t_i\sigma \succsim s_{i+1}\sigma$. For innermost termination, a weak decrease is not required for all rules but only for the *usable rules*. They are a superset of those rules that can reduce right-hand sides of dependency pairs if their variables are instantiated with normal forms.

Definition 5 (Usable Rules). *For $\mathcal{F}' \subseteq \mathcal{F} \cup \mathcal{F}^\sharp$, let $Rls(\mathcal{F}') = \{l \to r \in \mathcal{R} \mid \mathrm{root}(l) \in \mathcal{F}'\}$. For any term t, the* usable rules *are the smallest set such that*

- $\mathcal{U}(x) = \varnothing$ *for* $x \in \mathcal{V}$ *and*
- $\mathcal{U}(f(t_1, \ldots, t_n)) = Rls(\{f\}) \;\cup\; \bigcup_{l \to r \in Rls(\{f\})} \mathcal{U}(r) \;\cup\; \bigcup_{j=1}^{n} \mathcal{U}(t_j)$.

For any set $\mathcal{P}$ of dependency pairs, we define $\mathcal{U}(\mathcal{P}) = \bigcup_{s \to t \in \mathcal{P}} \mathcal{U}(t)$.

For the automated generation of reduction pairs, one uses standard (monotonic) simplification orders. To build non-monotonic orders from simplification orders, one may drop function symbols and function arguments by an *argument filtering* [2] (we use the notation of [22]).

[1] Estimated dependency graphs may contain an additional arc from (6) to (8). However, if one uses the refinement of *instantiating* dependency pairs [10, 12], then all existing estimation techniques would detect that this arc is unnecessary.

Definition 6 (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* $[i_1, \ldots, i_k]$ *with* $1 \leq i_1 < \ldots < i_k \leq n$*. The signature* $\mathcal{F}_\pi$ *consists of all symbols* f *with* $\pi(f) = [i_1, \ldots, i_k]$*, where in* $\mathcal{F}_\pi$*,* f *has arity* k*. An argument filtering with* $\pi(f) = i$ *for some* $f \in \mathcal{F}$ *is* collapsing*. 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*:*

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

For a TRS $\mathcal{R}$*,* $\pi(\mathcal{R})$ *denotes* $\{\pi(l) \to \pi(r) \mid l \to r \in \mathcal{R}\}$*.*

For an argument filtering π and reduction pair $(\succsim, \succ)$, $(\succsim_\pi, \succ_\pi)$ is the reduction pair with $s \succsim_\pi t$ iff $\pi(s) \succsim \pi(t)$ and $s \succ_\pi t$ iff $\pi(s) \succ \pi(t)$. Let $\underset{(\sim)}{\succ} \; = \; \succsim \cup \succ$ and $\underset{(\sim)}{\succ}_\pi \; = \; \succsim_\pi \cup \succ_\pi$. In the following, we always regard filterings for $\mathcal{F} \cup \mathcal{F}^\sharp$.

Theorem 7 (Modular (Innermost) Termination Proofs [11]). *A TRS* $\mathcal{R}$ *is terminating iff for every cycle* $\mathcal{P}$ *of the dependency graph there is a reduction pair* $(\succsim, \succ)$ *and an argument filtering* π *such that both*

(a) $s \underset{(\sim)}{\succ}_\pi t$ *for all pairs* $s \to t \in \mathcal{P}$ *and* $s \succ_\pi t$ *for at least one* $s \to t \in \mathcal{P}$
(b) $l \succsim_\pi r$ *for all rules* $l \to r \in \mathcal{R}$

$\mathcal{R}$ *is innermost terminating if for every cycle* $\mathcal{P}$ *of the innermost dependency graph there is a reduction pair* $(\succsim, \succ)$ *and an argument filtering* π *satisfying both (a) and*

(c) $l \succsim_\pi r$ *for all rules* $l \to r \in \mathcal{U}(\mathcal{P})$

Thm. 7 permits modular[2] proofs, since one can use different filterings and reduction pairs for different cycles. This is inevitable to handle large programs in practice. See [12, 18] for techniques to automate Thm. 7 efficiently.

Innermost termination implies termination for locally confluent overlay systems and thus, for non-overlapping TRSs [17]. So for such TRSs one should only prove innermost termination, since the constraints for innermost termination are a subset of the constraints for termination. However, the TRS of Ex. 1 is not locally confluent: $\mathsf{div}(0, 0)$ reduces to the normal forms $\mathsf{0}$ and $\mathsf{quot}(0, 0, 0)$.

[2] In this paper, "modularity" means that one can split up the termination proof of a TRS $\mathcal{R}$ into several independent subproofs. However, "modularity" can also mean that one would like to split a TRS into subsystems and prove their termination more or less independently. For innermost termination, Thm. 7 also permits such forms of modularity. For example, if $\mathcal{R}$ is a hierarchical combination of $\mathcal{R}_1$ and $\mathcal{R}_2$, we have $\mathcal{U}(\mathcal{P}) \subseteq \mathcal{R}_1$ for every cycle $\mathcal{P}$ of $\mathcal{R}_1$-dependency pairs. Thus, one can prove innermost termination of $\mathcal{R}_1$ independently of $\mathcal{R}_2$. Thm. 11 and its improvements will show that similar modular proofs are also possible for termination instead of innermost termination. Then for hierarchical combinations, termination of $\mathcal{R}_1$ can be proved independently of $\mathcal{R}_2$, provided one uses an estimation of the dependency graph where no further cycles of $\mathcal{R}_1$-dependency pairs are introduced if $\mathcal{R}_1$ is extended by $\mathcal{R}_2$.

Example 8. An automated termination proof of Ex. 1 is virtually impossible with Thm. 7. We get the constraints $\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y), z) \succ_\pi \mathsf{QUOT}(x, y, z)$ and $l \succsim_\pi r$ for all $l \to r \in \mathcal{R}$ from the cycle $\{(7)\}$. However, they cannot be solved by a reduction pair $(\succsim, \succ)$ where $\succsim$ is a quasi-simplification order: For $t = \mathsf{quot}(x, \mathsf{0}, \mathsf{s}(\mathsf{0}))$ we have $t \succsim_\pi \mathsf{s}(\mathsf{quot}(x, \mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0})))$ by rules (5) and (2). Moreover, $\mathsf{s}(\mathsf{quot}(x, \mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0}))) \succsim_\pi \mathsf{s}(t)$ by the subterm property, since $\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y), z) \succ_\pi \mathsf{QUOT}(x, y, z)$ implies $\pi(\mathsf{s}) = [1]$. But $t \succsim_\pi \mathsf{s}(t)$ implies $\mathsf{QUOT}(\mathsf{s}(t), \mathsf{s}(t), z) \succ_\pi \mathsf{QUOT}(t, t, z) \succsim_\pi \mathsf{QUOT}(\mathsf{s}(t), \mathsf{s}(t), z)$ which contradicts the well-foundedness of $\succ_\pi$.

In contrast, innermost termination of Ex. 1 can easily be proved. There are no usable rules because the dependency pairs have no defined symbols in their right-hand sides. Hence, with a filtering $\pi(\mathsf{QUOT}) = \pi(\mathsf{DIV}) = 1$, the constraints for innermost termination are satisfied by the embedding order.

Our goal is to modify the technique for termination such that its constraints become as simple as the ones for innermost termination. As observed in [29], the following definition is useful to weaken the constraint (b) for termination.

Definition 9 ($\mathcal{C}_\varepsilon$ [16]). *The TRS $\mathcal{C}_\varepsilon$ is defined as $\{\mathsf{c}(x, y) \to x, \mathsf{c}(x, y) \to y\}$ where c is a new function symbol. A TRS $\mathcal{R}$ is $\mathcal{C}_\varepsilon$-*terminating *iff $\mathcal{R} \cup \mathcal{C}_\varepsilon$ is terminating. A relation $\succsim$ is $\mathcal{C}_\varepsilon$-*compatible[3] *iff $\mathsf{c}(x, y) \succsim x$ and $\mathsf{c}(x, y) \succsim y$. A reduction pair $(\succsim, \succ)$ is $\mathcal{C}_\varepsilon$-*compatible *iff $\succsim$ is $\mathcal{C}_\varepsilon$-compatible.*

The TRS $\mathcal{R} = \{\mathsf{f}(\mathsf{0}, \mathsf{1}, x) \to \mathsf{f}(x, x, x)\}$ of Toyama [28] is terminating, but not $\mathcal{C}_\varepsilon$-terminating, since $\mathcal{R} \cup \mathcal{C}_\varepsilon$ admits the infinite reduction $\mathsf{f}(\mathsf{0}, \mathsf{1}, \mathsf{c}(\mathsf{0}, \mathsf{1})) \to \mathsf{f}(\mathsf{c}(\mathsf{0}, \mathsf{1}), \mathsf{c}(\mathsf{0}, \mathsf{1}), \mathsf{c}(\mathsf{0}, \mathsf{1})) \to^2 \mathsf{f}(\mathsf{0}, \mathsf{1}, \mathsf{c}(\mathsf{0}, \mathsf{1})) \to \ldots$. This example shows that requiring $l \succsim_\pi r$ only for usable rules is not sufficient for termination: $\mathcal{R} \cup \mathcal{C}_\varepsilon$'s only cycle $\{\mathsf{F}(\mathsf{0}, \mathsf{1}, x) \to \mathsf{F}(x, x, x)\}$ has no usable rules and there is a reduction pair $(\succsim, \succ)$ satisfying the constraint (a).[4] So $\mathcal{R} \cup \mathcal{C}_\varepsilon$ is innermost terminating, but not terminating, since we cannot satisfy both (a) and $l \succsim r$ for the $\mathcal{C}_\varepsilon$-rules.

So a reduction of the constraints in (b) is impossible in general, but it is possible if we restrict ourselves to $\mathcal{C}_\varepsilon$-compatible reduction pairs. This restriction is not severe, since virtually all reduction pairs used in practice (based on LPO [20], RPOS [7], KBO [21], or polynomial orders[5] [23]) are $\mathcal{C}_\varepsilon$-compatible.

The first step in this direction was taken by Urbain [29]. He showed that in a hierarchy of $\mathcal{C}_\varepsilon$-terminating TRSs, one can disregard all rules occurring "later" in the hierarchy when proving termination. Hence, when showing the termination of functions which call div or quot, one has to require $l \succsim_\pi r$ for the div- and quot-rules. But if one regards functions which do not depend on div or quot, then one does not have to take the div- and quot-rules into account in constraint (b).

But due to the restriction to $\mathcal{C}_\varepsilon$-termination, [29] could not use the full power of dependency graphs. For example, recent dependency graph estimations [18]

---

[3] Instead of "$\mathcal{C}_\varepsilon$-compatibility", [29] uses the corresponding notion "π extendibility".

[4] For example, it is satisfied by the reduction pair $(\to^*_{\mathcal{R} \cup DP(\mathcal{R})}, \to^+_{\mathcal{R} \cup DP(\mathcal{R})})$.

[5] Any polynomial order can be extended to the symbol c such that it is $\mathcal{C}_\varepsilon$-compatible.

detect that the dependency graph for Toyama's TRS $\mathcal{R}$ has no cycle and thus, it is terminating. But since it is not $\mathcal{C}_\varepsilon$-terminating, it cannot be handled by [29].

In [12], we integrated the approach of [29] with (arbitrary estimations of) dependency graphs, by restricting ourselves to $\mathcal{C}_\varepsilon$-compatible reduction pairs instead of $\mathcal{C}_\varepsilon$-terminating TRSs. This combines the advantages of both approaches, since now one only regards those rules in (b) that the current *cycle depends on.*

Definition 10 (Dependence). *Let $\mathcal{R}$ be a TRS. For two symbols f and g we say that f* depends on *g (denoted $f \sqsupset_0 g$) iff g occurs in an f-rule of $\mathcal{R}$ (i.e., in $Rls(\{f\})$). Moreover, every tuple symbol $f^\sharp$* depends on *f. A cycle of dependency pairs $\mathcal{P}$* depends on *all symbols occurring in its dependency pairs.*[6] *We write $\sqsupset_0^+$ for the transitive closure of $\sqsupset_0$. For every cycle $\mathcal{P}$ we define $\Delta_0(\mathcal{P},\mathcal{R}) = \{f \mid \mathcal{P} \sqsupset_0^+ f\}$. If $\mathcal{P}$ and $\mathcal{R}$ are clear from the context we just write Δ_0 or $\Delta_0(\mathcal{P})$.*

In Ex. 1, we have $\mathsf{div} \sqsupset_0 \mathsf{quot}$, $\mathsf{quot} \sqsupset_0 \mathsf{div}$, and each defined symbol depends on itself. As $\mathsf{QUOT} \sqsupset_0 \mathsf{quot} \sqsupset_0 \mathsf{div}$, Δ_0 contains quot and div for both cycles $\mathcal{P}$.

The next theorem shows that it suffices to require a weak decrease only for the rules that the cycle depends on. It improves upon Thm. 7 since the constraints of type (b) are reduced significantly. Thus, it becomes easier to find a reduction pair satisfying the resulting constraints. This increases both efficiency and power. For instance, termination of a well-known example of [25] to compute intervals of natural numbers cannot be shown with Thm. 7 and a reduction pair based on simplification orders, while a proof with Thm. 11 and LPO is easy [12].

Theorem 11 (Improved Modular Termination, Version 0 [12]). *A TRS $\mathcal{R}$ is terminating if for every cycle $\mathcal{P}$ of the dependency graph there is a $\mathcal{C}_\varepsilon$-compatible reduction pair $(\succsim, \succ)$ and an argument filtering π satisfying both constraint Thm. 7 (a) and*

(b) $l \succsim_\pi r$ *for all rules* $l \to r \in Rls(\Delta_0(\mathcal{P},\mathcal{R}))$

Proof. The proof is based on the following key observation [29, Lemma 2]:

$$\text{Every minimal } \mathcal{P}\text{-chain over } \mathcal{R} \text{ is a } \mathcal{P}\text{-chain over } Rls(\Delta_0(\mathcal{P},\mathcal{R})) \cup \mathcal{C}_\varepsilon. \qquad (9)$$

For the proof of Thm. 11, by Thm. 4 we have to show absence of minimal infinite $\mathcal{P}$-chains $s_1 \to t_1, s_2 \to t_2, \ldots$ over $\mathcal{R}$. By (9), such a chain is also a chain over $Rls(\Delta_0(\mathcal{P},\mathcal{R})) \cup \mathcal{C}_\varepsilon$. Hence, there is a substitution σ with $t_i\sigma \to^*_{Rls(\Delta_0(\mathcal{P},\mathcal{R}))\cup\mathcal{C}_\varepsilon} s_{i+1}\sigma$ for all i. We extend π to c by $\pi(\mathsf{c}) = [1,2]$. So $\mathcal{C}_\varepsilon$-compatibility of $\succsim$ implies $\mathcal{C}_\varepsilon$-compatibility of $\succsim_\pi$. By (b) we have $t_i\sigma \succsim_\pi s_{i+1}\sigma$ for all i as $\succsim_\pi$ is stable and monotonic. Using (a) and stability of $\succ_\pi$ leads to $s_i\sigma \succ_\pi t_i\sigma$ for infinitely many i and $s_i\sigma \succsim_\pi t_i\sigma$ for all remaining i contradicting $\succ_\pi$'s well-foundedness. $\square$

The proof shows that Thm. 11 only relies on observation (9). When refining the definition of Δ_0 in the next section, we only have to prove that (9) still holds.

[6] The symbol "$\sqsupset_0$" is overloaded to denote both the dependence between function symbols ($f \sqsupset_0 g$) and between cycles and function symbols ($\mathcal{P} \sqsupset_0 f$).

3 No Dependences for Tuple Symbols & Left-Hand Sides

Thm. 11 reduces the constraints for termination considerably. However for Ex. 1, the constraints according to Thm. 11 are the same as with Thm. 7. The reason is that both cycles $\mathcal{P}$ depend on quot and div and therefore, $Rls(\Delta_0(\mathcal{P})) = \mathcal{R}$. Hence, as shown in Ex. 8, an automated termination proof is virtually impossible.

To solve this problem, we improve the notion of "dependence" by dropping the condition that every tuple symbol $f^\sharp$ depends on f. Then the cycles in Ex. 1 do not depend on any defined function symbol anymore, since they contain no defined symbols. When modifying the definition of $\Delta_0(\mathcal{P})$ in this way in Thm. 11, we obtain no constraints of type (b) for Ex. 1, since $Rls(\Delta_0(\mathcal{P})) = \varnothing$. So now the constraints for termination of this example are the same as for innermost termination and the proof succeeds with the embedding order, cf. Ex. 8.[7]

Now the only difference between $\mathcal{U}(\mathcal{P})$ and $Rls(\Delta_0(\mathcal{P}))$ is that in $Rls(\Delta_0(\mathcal{P}))$, f also depends on g if g occurs in the left-hand side of an f-rule. Similarly, $\mathcal{P}$ also depends on g if g occurs in the left-hand side of a dependency pair from $\mathcal{P}$. The following example shows that disregarding dependences from left-hand sides (as in $\mathcal{U}(\mathcal{P})$) can be necessary for the success of the termination proof.

Example 12. We extend the TRS for division from Ex. 1 by the following rules.

$$\begin{array}{rlrl}
\mathsf{plus}(x,\mathsf{0}) &\to x & \mathsf{times}(\mathsf{0},y) &\to \mathsf{0}\\
\mathsf{plus}(\mathsf{0},y) &\to y & \mathsf{times}(\mathsf{s}(\mathsf{0}),y) &\to y\\
\mathsf{plus}(\mathsf{s}(x),y) &\to \mathsf{s}(\mathsf{plus}(x,y)) & \mathsf{div}(\mathsf{div}(x,y),z) &\to \mathsf{div}(x,\mathsf{times}(y,z))
\end{array}$$

Even when disregarding dependences $f^\sharp \sqsupset_0 f$, the constraints of Thm. 11 for this TRS are not satisfiable by reduction pairs based on RPOS, KBO, or polynomial orders: Any cycle containing the new dependency pair $\mathsf{DIV}(\mathsf{div}(x,y),z) \to \mathsf{DIV}(x,\mathsf{times}(y,z))$ would depend on both div and times and thus, all rules of the TRS would have to be weakly decreasing. Weak decrease of plus and times implies that one has to use an argument filtering with $\mathsf{s}(x) \succ_\pi x$. But since $t \succsim_\pi \mathsf{s}(t)$ for the term $t = \mathsf{quot}(x,\mathsf{0},\mathsf{s}(\mathsf{0}))$ as shown in Ex. 8, this gives a contradiction.

Cycles with $\mathsf{DIV}(\mathsf{div}(x,y),z) \to \mathsf{DIV}(x,\mathsf{times}(y,z))$ only depend on div because it occurs in the *left-hand side*. This motivates the following refinement of $\sqsupset_0$.

Definition 13 (Refined Dependence, Version 1). *For two function symbols f and g, the refined dependence relation $\sqsupset_1$ is defined as $f \sqsupset_1 g$ iff g occurs in the right-hand side of an f-rule and a cycle $\mathcal{P}$ depends on all symbols in the right-hand sides of its dependency pairs. Again, $\Delta_1(\mathcal{P},\mathcal{R}) = \{f \mid \mathcal{P} \sqsupset_1^+ f\}$.*

With Def. 13, the constraints of Thm. 11 are the same as in the innermost case: $\mathcal{U}(\mathcal{P}) = Rls(\Delta_1(\mathcal{P}))$ and termination of Ex. 12 can be proved using LPO.

To show that one may indeed regard $\Delta_1(\mathcal{P})$ instead of $\Delta_0(\mathcal{P})$ in Thm. 11, we prove an adapted version of (9) with Δ_1 instead of Δ_0. As in the proofs for Δ_0 in

[7] If an estimated dependency graph has the additional cycle $\{(6),(8)\}$, here one may use an LPO with $\pi(\mathsf{DIV}) = \pi(\mathsf{QUOT}) = 2$, $\pi(\mathsf{s}) = [\,]$, and the precedence $\mathsf{0} > \mathsf{s}$.

[24, 29] and in the original proofs of Gramlich [16], we map any $\mathcal{R}$-reduction to a reduction w.r.t. $\mathit{Rls}(\Delta_1) \cup \mathcal{C}_\varepsilon$. However, our mapping $\mathcal{I}_1$ is a modification of these earlier mappings, since terms $g(t_1, \ldots, t_n)$ with $g \notin \Delta_1$ are treated differently. Fig. 1 illustrates that by this mapping, every minimal chain over $\mathcal{R}$ corresponds to a chain over $\mathit{Rls}(\Delta_1) \cup \mathcal{C}_\varepsilon$, but instead of the substitution σ one uses a different substitution $\mathcal{I}_1(\sigma)$. Thus, the observation (9) also holds for Δ_1 instead of Δ_0.

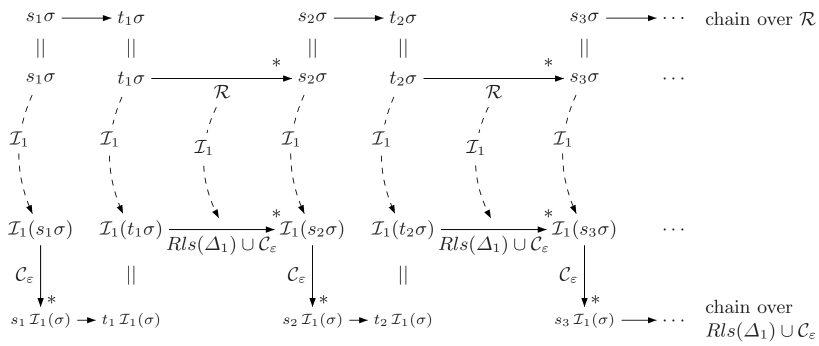


Fig. 1. Transformation of chains

Intuitively, $\mathcal{I}_1(t)$ "collects" all terms that t can be reduced to. However, we only regard reductions on or below symbols that are not from Δ_1. Normal forms whose roots are not from Δ_1 may be replaced by a fresh variable. To represent a collection $t_1, \ldots, t_n$ of terms by just one term, one uses $\mathsf{c}(t_1, \mathsf{c}(t_2, \ldots \mathsf{c}(t_n, x) \ldots))$.

Definition 14. *Let $\Delta \subseteq \mathcal{F} \cup \mathcal{F}^\sharp$ and let $t \in \mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ be a terminating term. We define $\mathcal{I}_1(t)$:*

$$\begin{aligned} \mathcal{I}_1(x) &= x && \text{for } x \in \mathcal{V} \\ \mathcal{I}_1(f(t_1, ..., t_n)) &= f(\mathcal{I}_1(t_1), ..., \mathcal{I}_1(t_n)) && \text{for } f \in \Delta \\ \mathcal{I}_1(g(t_1, ..., t_n)) &= \mathcal{C}omp(\{g(\mathcal{I}_1(t_1), ..., \mathcal{I}_1(t_n))\} \cup \mathcal{R}ed_1(g(t_1, ..., t_n))) && \text{for } g \notin \Delta \end{aligned}$$

where $\mathcal{R}ed_1(t) = \{\mathcal{I}_1(t') \mid t \to_{\mathcal{R}} t'\}$. Moreover, $\mathcal{C}omp(\{t\} \uplus M) = \mathsf{c}(t, \mathcal{C}omp(M))$ and $\mathcal{C}omp(\varnothing) = x_{new}$, where x_{new} is a fresh variable. To ensure that $\mathcal{C}omp$ is well-defined we assume that in the recursive definition of $\mathcal{C}omp(\{t\} \uplus M)$, t is smaller than all terms in M due to some total well-founded order $>_{\mathcal{T}}$ on terms.

For every terminating substitution σ (i.e., $\sigma(x)$ is terminating for all $x \in \mathcal{V}$), we define the substitution $\mathcal{I}_1(\sigma)$ as $\mathcal{I}_1(\sigma)\,(x) = \mathcal{I}_1(\sigma(x))$ for all $x \in \mathcal{V}$.

Note that Def. 14 is only possible for terminating terms t, since otherwise, $\mathcal{I}_1(t)$ could be infinite. Before we can show that Thm. 11 can be adapted to the refined definition Δ_1, we need some additional properties of $\mathcal{C}omp$ and $\mathcal{I}_1$. In contrast to the corresponding lemmas in [24, 29], they demonstrate that the rules of $\Delta_0 \setminus \Delta_1$ are not needed and we show in Lemma 16 (ii) and (iii) how to handle dependency pairs and rules where the left-hand side is not from $\mathcal{T}(\Delta_1, \mathcal{V})$.[8]

Lemma 15 (Properties of $\mathcal{C}omp$). *If $t \in M$ then $\mathcal{C}omp(M) \to_{\mathcal{C}_\varepsilon}^{+} t$.*

Proof. For $t_1 <_{\mathcal{T}} \cdots <_{\mathcal{T}} t_n$ and any $1 \le i \le n$ we have $\mathcal{C}omp(\{t_1, \ldots, t_n\}) = \mathsf{c}(t_1, \ldots \mathsf{c}(t_i, \ldots \mathsf{c}(t_n, x) \ldots) \ldots) \to_{\mathcal{C}_\varepsilon}^{*} \mathsf{c}(t_i, \ldots \mathsf{c}(t_n, x) \ldots) \to_{\mathcal{C}_\varepsilon} t_i$. $\square$

Lemma 16 (Properties of $\mathcal{I}_1$). *Let $\Delta \subseteq \mathcal{F} \cup \mathcal{F}^\sharp$ where $f \in \Delta$ and $f \sqsupset_1 g$ implies $g \in \Delta$. Let $t, s, t\sigma \in \mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ be terminating terms and let σ be a terminating substitution.*

(i) If $t \in \mathcal{T}(\Delta, \mathcal{V})$ then $\mathcal{I}_1(t\sigma) = t\, \mathcal{I}_1(\sigma)$.
(ii) $\mathcal{I}_1(t\sigma) \to_{\mathcal{C}_\varepsilon}^{*} t\, \mathcal{I}_1(\sigma)$.
(iii) If $t \to_{\{l \to r\}} s$ by a root reduction step where $l \to r \in \mathcal{R}$ and $\mathrm{root}(l) \in \Delta$, then $\mathcal{I}_1(t) \to_{\{l \to r\} \cup \mathcal{C}_\varepsilon}^{+} \mathcal{I}_1(s)$.
(iv) If $t \to_{\mathcal{R}} s$ with $\mathrm{root}(t) \notin \Delta$, then $\mathcal{I}_1(t) \to_{\mathcal{C}_\varepsilon}^{+} \mathcal{I}_1(s)$.
(v) If $t \to_{\{l \to r\}} s$ where $l \to r \in \mathcal{R}$, then $\mathcal{I}_1(t) \to_{\{l \to r\} \cup \mathcal{C}_\varepsilon}^{+} \mathcal{I}_1(s)$ if $\mathrm{root}(l) \in \Delta$ and $\mathcal{I}_1(t) \to_{\mathcal{C}_\varepsilon}^{+} \mathcal{I}_1(s)$ otherwise.

Proof.

(i) The proof is a straightforward structural induction on t.
(ii) The proof is by structural induction on t. The only interesting case is $t = g(t_1, \ldots, t_n)$ where $g \notin \Delta$. Then we obtain

$$\begin{array}{rcll} \mathcal{I}_1(g(t_1, ..., t_n)\sigma) & = & \mathcal{C}omp(\{g(\mathcal{I}_1(t_1\sigma), ..., \mathcal{I}_1(t_n\sigma))\} \cup \mathcal{R}ed_1(g(t_1\sigma, \ldots, t_n\sigma))) & \\ & \to_{\mathcal{C}_\varepsilon}^{+} & g(\mathcal{I}_1(t_1\sigma), ..., \mathcal{I}_1(t_n\sigma)) & \text{by Lemma 15} \\ & \to_{\mathcal{C}_\varepsilon}^{*} & g(t_1\, \mathcal{I}_1(\sigma), \ldots, t_n\, \mathcal{I}_1(\sigma)) & \text{by induction hypothesis} \\ & = & g(t_1, \ldots, t_n)\, \mathcal{I}_1(\sigma) & \end{array}$$

(iii) We have $t = l\sigma \to_{\mathcal{R}} r\sigma = s$. By the definition of $\sqsupset_1$, r is a term of $\mathcal{T}(\Delta, \mathcal{V})$. Using (ii) and (i) we get $\mathcal{I}_1(l\sigma) \to_{\mathcal{C}_\varepsilon}^{*} l\, \mathcal{I}_1(\sigma) \to_{\{l \to r\}} r\, \mathcal{I}_1(\sigma) = \mathcal{I}_1(r\sigma)$.
(iv) follows by $\mathcal{I}_1(t) = \mathcal{C}omp(\{\ldots\} \cup \mathcal{R}ed_1(t))$, $\mathcal{I}_1(s) \in \mathcal{R}ed_1(t)$, and Lemma 15.
(v) We do induction on the position p of the redex. If $\mathrm{root}(t) \notin \Delta$, we use (iv). If $\mathrm{root}(t) \in \Delta$ and p is the root position, we apply (iii). Otherwise, p is below the root, $t = f(t_1, \ldots, t_i, \ldots, t_n)$, $s = f(t_1, \ldots, s_i, \ldots, t_n)$, $f \in \Delta$, and $t_i \to_{\{l \to r\}} s_i$. Then the claim follows from the induction hypothesis. $\square$

Now we show that in Thm. 11 one may replace Δ_0 by Δ_1.

Theorem 17 (Improved Modular Termination, Version 1). *A TRS $\mathcal{R}$ is terminating if for every cycle $\mathcal{P}$ of the dependency graph there is a $\mathcal{C}_\varepsilon$-compatible reduction pair $(\succsim, \succ)$ and an argument filtering π satisfying both constraint Thm. 7 (a) and*

(b) $l \succsim_\pi r$ for all rules $l \to r \in Rls(\Delta_1(\mathcal{P}, \mathcal{R}))$

[8] Here, equalities in the lemmas of [24, 29] are replaced by $\mathcal{C}_\varepsilon$-steps. This is possible by including the term $g(\mathcal{I}_1(t_1), \ldots, \mathcal{I}_1(t_n))$ in the definition of $\mathcal{I}_1(g(t_1, ..., t_n))$.

Proof. The proof is as for Thm. 11, but instead of (9) one uses this observation:

$$\text{Every minimal } \mathcal{P}\text{-chain over } \mathcal{R} \text{ is a } \mathcal{P}\text{-chain over } Rls(\Delta_1(\mathcal{P},\mathcal{R})) \cup \mathcal{C}_\varepsilon. \quad (10)$$

To prove (10), let $s_1 \to t_1, s_2 \to t_2, \ldots$ be a minimal $\mathcal{P}$-chain over $\mathcal{R}$. Hence, there is a substitution σ such that $t_i\sigma \to_{\mathcal{R}}^* s_{i+1}\sigma$ and all terms $s_i\sigma$ and $t_i\sigma$ are terminating. This enables us to apply $\mathcal{I}_1$ to both $t_i\sigma$ and $s_i\sigma$ (where we choose Δ to be $\Delta_1(\mathcal{P},\mathcal{R})$). Using Lemma 16 (v) we obtain $\mathcal{I}_1(t_i\sigma) \to^*_{Rls(\Delta_1)\cup\mathcal{C}_\varepsilon} \mathcal{I}_1(s_{i+1}\sigma)$.

Moreover, by the definition of $\sqsupset_1$, all t_i are terms over the signature Δ_1. Thus, by Lemma 16 (i) and (ii) we get $t_i\,\mathcal{I}_1(\sigma) = \mathcal{I}_1(t_i\sigma) \to^*_{Rls(\Delta_1)\cup\mathcal{C}_\varepsilon} \mathcal{I}_1(s_{i+1}\sigma) \to^*_{\mathcal{C}_\varepsilon} s_{i+1}\,\mathcal{I}_1(\sigma)$ stating that $s_1 \to t_1, s_2 \to t_2, \ldots$ is also a chain over $Rls(\Delta_1)\cup\mathcal{C}_\varepsilon$. $\square$

4 Dependences With Respect to Argument Filterings

For innermost termination, one may first apply the argument filtering π and determine the usable rules $\mathcal{U}(\mathcal{P},\pi)$ afterwards, cf. [12]. The advantage is that the argument filtering may eliminate some symbols f from right-hand sides of dependency pairs and rules. Then, the f-rules do not have to be weakly decreasing anymore. We also presented an algorithm to determine suitable argument filterings, which is non-trivial since the filtering determines the resulting constraints.

We now introduce a corresponding improvement for termination by defining "dependence" w.r.t. an argument filtering. Then a cycle only depends on those symbols that are not dropped by the filtering. However, this approach is only sound for non-collapsing argument filterings. Consider the non-terminating TRS

$$\mathsf{f}(\mathsf{s}(x)) \to \mathsf{f}(\mathsf{double}(x)) \qquad \mathsf{double}(\mathsf{0}) \to \mathsf{0} \qquad \mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

In the cycle $\{\mathsf{F}(\mathsf{s}(x)) \to \mathsf{F}(\mathsf{double}(x))\}$, the filtering $\pi(\mathsf{double}) = 1$ results in $\{\mathsf{F}(\mathsf{s}(x)) \to \mathsf{F}(x)\}$. Since the filtered pair has no defined symbols, we would conclude that no rule must be weakly decreasing for this cycle. But then we can solve the cycle's only constraint $\mathsf{F}(\mathsf{s}(x)) \succ \mathsf{F}(x)$ and falsely prove termination.[9]

Example 18. We extend the TRS of Ex. 12 by rules for prime numbers.

$$\begin{array}{rlrl}
\mathsf{prime}(\mathsf{s}(\mathsf{s}(x))) & \to \mathsf{pr}(\mathsf{s}(\mathsf{s}(x)), \mathsf{s}(x)) & \mathsf{pr}(x, \mathsf{s}(\mathsf{0})) & \to \mathsf{true} \\
\mathsf{eq}(\mathsf{0}, \mathsf{0}) & \to \mathsf{true} & \mathsf{pr}(x, \mathsf{s}(\mathsf{s}(y))) & \to \mathsf{if}(\mathsf{divides}(\mathsf{s}(\mathsf{s}(y)), x), x, \mathsf{s}(y)) \\
\mathsf{eq}(\mathsf{s}(x), \mathsf{0}) & \to \mathsf{false} & \mathsf{if}(\mathsf{true}, x, y) & \to \mathsf{false} \\
\mathsf{eq}(\mathsf{0}, \mathsf{s}(y)) & \to \mathsf{false} & \mathsf{if}(\mathsf{false}, x, y) & \to \mathsf{pr}(x, y) \\
\mathsf{eq}(\mathsf{s}(x), \mathsf{s}(y)) & \to \mathsf{eq}(x, y) & \mathsf{divides}(y, x) & \to \mathsf{eq}(x, \mathsf{times}(\mathsf{div}(x, y), y))
\end{array}$$

The cycle $\{\mathsf{PR}(x, \mathsf{s}(\mathsf{s}(y))) \to \mathsf{IF}(\mathsf{divides}(\mathsf{s}(\mathsf{s}(y)), x), x, \mathsf{s}(y)),\ \mathsf{IF}(\mathsf{false}, x, y) \to \mathsf{PR}(x, y)\}$ depends on divides and hence, on div and times. So for this cycle, Thm. 17

[9] Essentially, we prove absence of infinite $\pi(\mathcal{P})$-chains over $\pi(\mathcal{R})$. But if π is collapsing, then the rules of $\pi(\mathcal{R})$ may have left-hand sides l with $\mathrm{root}(l) \in \mathcal{C}$ or $l \in \mathcal{V}$. Thus, inspecting the defined symbols in a term $\pi(t)$ is not sufficient to estimate which rules may be used for the $\pi(\mathcal{R})$-reduction of $\pi(t)$.

requires the div- and times-rules to be weakly decreasing. This is impossible with reduction pairs based on RPOS, KBO, or polynomial orders, cf. Ex. 12.

But if we first use the filtering $\pi(\mathsf{IF}) = [2,3]$ and compute dependences afterwards, then the cycle no longer depends on divides, div, or times. If one modifies "dependence" in this way, then the constraints can again be solved by LPO.

Definition 19 (Refined Dependence, Version 2). *Let π be a non-collapsing argument filtering. For two function symbols f and g we define $f \sqsupset_2 g$ iff there is a rule $l \to r \in Rls(\{f\})$ where g occurs in $\pi(r)$. For a cycle of dependency pairs $\mathcal{P}$, we define $\mathcal{P} \sqsupset_2 g$ iff there is a pair $s \to t \in \mathcal{P}$ where g occurs in $\pi(t)$. We define $\Delta_2(\mathcal{P}, \mathcal{R}, \pi) = \{f \mid \mathcal{P} \sqsupset_2^+ f\}$ and omit $\mathcal{P}$, $\mathcal{R}$, π if they are clear from the context.*

To show that Δ_1 may be replaced by Δ_2 in Thm.17, we define a new mapping $\mathcal{I}_2$.

Definition 20. *Let π be a non-collapsing argument filtering, $\Delta \subseteq \mathcal{F} \cup \mathcal{F}^\sharp$, $t \in \mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ be terminating. We define $\mathcal{I}_2(t)$. Here, $\mathcal{R}ed_2(t) = \{\mathcal{I}_2(t') \mid t \to_\mathcal{R} t'\}$.*

$$\begin{aligned}
\mathcal{I}_2(x) &= x && \text{for } x \in \mathcal{V}\\
\mathcal{I}_2(f(t_1,\ldots,t_n)) &= f(\mathcal{I}_2(t_{i_1}),\ldots,\mathcal{I}_2(t_{i_k})) && \text{for } f \in \Delta,\ \pi(f) = [i_1,...,i_k]\\
\mathcal{I}_2(g(t_1,\ldots,t_n)) &= \mathcal{C}omp(\ \{g(\mathcal{I}_2(t_{i_1}),\ldots,\mathcal{I}_2(t_{i_k}))\} \\
&\qquad \cup \mathcal{R}ed_2(g(t_1,\ldots,t_n))\ ) && \text{for } g \notin \Delta,\ \pi(g) = [i_1,...,i_k]
\end{aligned}$$

Lemma 21 differs from the earlier Lemma 16, since $\mathcal{I}_2$ already applies the argument filtering π and in (v), we have "$*$" instead of "$+$", as a reduction on a position that is filtered away leads to the same transformed terms w.r.t. $\mathcal{I}_2$.

Lemma 21 (Properties of $\mathcal{I}_2$). *Let π be a non-collapsing argument filtering and let $\Delta \subseteq \mathcal{F} \cup \mathcal{F}^\sharp$ such that $f \in \Delta$ and $f \sqsupset_2 g$ implies $g \in \Delta$. Let $t, s, t\sigma \in \mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ be terminating and let σ be a terminating substitution.*

(i) *If $\pi(t) \in \mathcal{T}(\Delta_\pi, \mathcal{V})$ then $\mathcal{I}_2(t\sigma) = \pi(t)\,\mathcal{I}_2(\sigma)$.*
(ii) *$\mathcal{I}_2(t\sigma) \to^*_{\mathcal{C}_\varepsilon} \pi(t)\,\mathcal{I}_2(\sigma)$.*
(iii) *If $t \to_{\{l \to r\}} s$ by a root reduction step where $l \to r \in \mathcal{R}$ and $\mathrm{root}(l) \in \Delta$, then $\mathcal{I}_2(t) \to^+_{\{\pi(l)\to\pi(r)\}\cup\mathcal{C}_\varepsilon} \mathcal{I}_2(s)$.*
(iv) *If $t \to_\mathcal{R} s$ with $\mathrm{root}(t) \notin \Delta$, then $\mathcal{I}_2(t) \to^+_{\mathcal{C}_\varepsilon} \mathcal{I}_2(s)$.*
(v) *If $t \to_{\{l\to r\}} s$ where $l \to r \in \mathcal{R}$, then $\mathcal{I}_2(t) \to^*_{\{\pi(l)\to\pi(r)\}\cup\mathcal{C}_\varepsilon} \mathcal{I}_2(s)$ if $\mathrm{root}(l) \in \Delta$ and $\mathcal{I}_2(t) \to^*_{\mathcal{C}_\varepsilon} \mathcal{I}_2(s)$ otherwise.*

Proof. The proof is analogous to the proof of Lemma 16. □

We are restricted to non-collapsing filterings when determining the rules that have to be weakly decreasing. But one can still use arbitrary (possibly collapsing) filterings in the dependency pair approach. For every filtering π we define its *non-collapsing variant* π' as $\pi'(f) = \pi(f)$ if $\pi(f) = [i_1,\ldots,i_k]$ and $\pi'(f) = [i]$ if $\pi(f) = i$. Now we show that in Thm. 17 one may replace Δ_1 by Δ_2.

Theorem 22 (Improved Modular Termination, Version 2). *A TRS $\mathcal{R}$ is terminating if for every cycle $\mathcal{P}$ of the dependency graph there is a $\mathcal{C}_\varepsilon$-compatible reduction pair $(\succsim, \succ)$ and an argument filtering π satisfying both constraint Thm. 7 (a) and* *(b) $l \succsim_\pi r$ for $l \to r \in Rls(\Delta_2(\mathcal{P},\mathcal{R},\pi'))$, where π' is π's non-collapsing variant*

Proof. Instead of (10), now we need the following main observation for the proof.

$$\text{If } s_1 \to t_1, s_2 \to t_2, \ldots \text{ is a minimal } \mathcal{P}\text{-chain over } \mathcal{R}, \text{ then } \pi'(s_1) \to \pi'(t_1), \pi'(s_2) \to \pi'(t_2), \ldots \text{ is a } \pi'(\mathcal{P})\text{-chain over } \pi'(Rls(\Delta_2(\mathcal{P},\mathcal{R},\pi'))) \cup \mathcal{C}_\varepsilon. \tag{11}$$

Similar to the proof of (10), $t_i\sigma \to_{\mathcal{R}}^* s_{i+1}\sigma$ implies that $\pi'(t_i)\mathcal{I}_2(\sigma) = \mathcal{I}_2(t_i\sigma) \to_{\pi'(Rls(\Delta_2))\cup\mathcal{C}_\varepsilon}^* \mathcal{I}_2(s_{i+1}\sigma) \to_{\mathcal{C}_\varepsilon}^* \pi'(s_{i+1})\,\mathcal{I}_2(\sigma)$ by Lemma 21 (i), (v), and (ii), which proves (11).

To show that (11) implies Thm. 22, assume that $s_1 \to t_1, s_2 \to t_2, \ldots$ is a minimal infinite $\mathcal{P}$-chain over $\mathcal{R}$. Then by (11) there is a substitution δ ($\mathcal{I}_2(\sigma)$ from above) with $\pi'(t_i)\,\delta \to_{\pi'(Rls(\Delta_2))\cup\mathcal{C}_\varepsilon}^* \pi'(s_{i+1})\,\delta$ for all i. Let π'' be the argument filtering for the signature $\mathcal{F}_{\pi'} \cup \mathcal{F}_{\pi'}^\sharp$ which only performs the collapsing steps of π (i.e., if $\pi(f) = i$ and thus $\pi'(f) = [i]$, we have $\pi''(f) = 1$). All other symbols of $\mathcal{F}_{\pi'} \cup \mathcal{F}_{\pi'}^\sharp$ are not filtered by π''. Hence, $\pi = \pi'' \circ \pi'$. We extend π'' to the new symbol c by defining $\pi''(\mathsf{c}) = [1,2]$. Hence, $\mathcal{C}_\varepsilon$-compatibility of $\succsim$ implies $\mathcal{C}_\varepsilon$-compatibility of $\succsim_{\pi''}$. Constraint (b) requires $\pi(l) \succsim \pi(r)$ for all rules of $Rls(\Delta_2)$. Therefore, we have $\pi'(l) \succsim_{\pi''} \pi'(r)$, and thus, all rules of $\pi'(Rls(\Delta_2)) \cup \mathcal{C}_\varepsilon$ are decreasing w.r.t. $\succsim_{\pi''}$. This implies $\pi'(t_i)\,\delta \succsim_{\pi''} \pi'(s_{i+1})\,\delta$ for all i. Moreover, (a) implies $\pi'(s_i)\,\delta \succ_{\pi''} \pi'(t_i)\,\delta$ for infinitely many i and $\pi'(s_i)\,\delta \succsim_{\pi''} \pi'(t_i)\,\delta$ for all remaining i. This contradicts the well-foundedness of $\succ_{\pi''}$. □

Now we are nearly as powerful as for innermost termination. The only difference between $\Delta_2(\mathcal{P},\mathcal{R},\pi)$ and $\mathcal{U}(\mathcal{P},\pi)$ is that $\mathcal{U}(\mathcal{P},\pi)$ may disregard subterms of right-hand sides of dependency pairs if they also occur on the left-hand side [12], since they are instantiated to normal forms in innermost chains. But for the special case of constructor systems, the left-hand sides of dependency pairs are constructor terms and thus $\Delta_2(\mathcal{P},\mathcal{R},\pi) = \mathcal{U}(\mathcal{P},\pi)$. The other differences between termination and innermost termination are that the innermost dependency graph is a subgraph of the dependency graph and may have fewer cycles. Moreover, the conditions for applying dependency pair transformations by narrowing, rewriting, or instantiation [2, 10, 12] are less restrictive for innermost termination. Finally for termination, we use $\mathcal{C}_\varepsilon$-compatible reduction pairs, which is not necessary for innermost termination. However, virtually all reduction pairs used in practice are $\mathcal{C}_\varepsilon$-compatible. So in general, innermost termination is still easier to prove than termination, but the difference has become much smaller.

5 Removing Rules

To reduce the constraints for termination proofs even further, in this section we present a technique to remove rules of the TRS that are not relevant for termination. To this end, the constraints for a cycle $\mathcal{P}$ may be pre-processed with a reduction pair $(\succsim, \succ)$. If all dependency pairs of $\mathcal{P}$ and all rules that $\mathcal{P}$ depends on are at least weakly decreasing (w.r.t. $\succsim$), then one may remove all those rules $\mathcal{R}_{\succ}$ that are strictly decreasing (w.r.t. $\succ$). So instead of proving absence of infinite $\mathcal{P}$-chains over $\mathcal{R}$ one only has to regard $\mathcal{P}$-chains over $\mathcal{R} \setminus \mathcal{R}_{\succ}$.

In contrast to related approaches to remove rules [15, 23, 30], we permit arbitrary reduction pairs and remove rules in the modular framework of dependency pairs instead of pre-processing a full TRS. So when removing rules for a cycle $\mathcal{P}$, we only have to regard the rules $\mathcal{P}$ depends on. Moreover, removing rules can be done repeatedly with different reduction pairs $(\succsim, \succ)$. Thm. 23 can also be adapted for innermost termination proofs with similar advantages as for termination.

Theorem 23 (Modular Removal of Rules). *Let $\mathcal{P}$ be a set of pairs, $\mathcal{R}$ be a TRS, and $(\succsim, \succ)$ be a reduction pair where $\succ$ is monotonic and $\mathcal{C}_\varepsilon$-compatible. If $l \mathrel{{}_{(}{\succsim}_{)}} r$ for all $l \to r \in Rls(\Delta_1(\mathcal{P}, \mathcal{R}))$ and $s \mathrel{{}_{(}{\succsim}_{)}} t$ for all $s \to t \in \mathcal{P}$ then the absence of minimal infinite $\mathcal{P}$-chains over $\mathcal{R} \setminus \mathcal{R}_{\succ}$ implies the absence of minimal infinite $\mathcal{P}$-chains over $\mathcal{R}$ where $\mathcal{R}_{\succ} = \{l \to r \in Rls(\Delta_1(\mathcal{P}, \mathcal{R})) \mid l \succ r\}$.*[10]

Proof. Let $s_1 \to t_1, s_2 \to t_2, \ldots$ be an infinite minimal $\mathcal{P}$-chain over $\mathcal{R}$. Hence, $t_i\sigma \to_{\mathcal{R}}^* s_{i+1}\sigma$. We show that in these reductions, $\mathcal{R}_{\succ}$-rules are only applied for finitely many i. So $t_i\sigma \to_{\mathcal{R}\setminus\mathcal{R}_{\succ}}^* s_{i+1}\sigma$ for all $i \geq n$ for some $n \in \mathbb{N}$. Thus, $s_n \to t_n, s_{n+1} \to t_{n+1}, \ldots$ is a minimal infinite $\mathcal{P}$-chain over $\mathcal{R} \setminus \mathcal{R}_{\succ}$ which proves Thm. 23.

Assume that $\mathcal{R}_{\succ}$-rules are applied for infinitely many i. By Lemma 16 (v) we get $\mathcal{I}_1(t_i\sigma) \to_{Rls(\Delta_1)\cup\mathcal{C}_\varepsilon}^* \mathcal{I}_1(s_{i+1}\sigma)$. As $\succ$ is $\mathcal{C}_\varepsilon$-compatible and $\to_{Rls(\Delta_1)} \subseteq {}_{(}{\succsim}_{)}$, we have $\mathcal{I}_1(t_i\sigma) \mathrel{{}_{(}{\succsim}_{)}} \mathcal{I}_1(s_{i+1}\sigma)$. Moreover, whenever an $\mathcal{R}_{\succ}$-rule is used in $t_i\sigma \to_{\mathcal{R}}^* s_{i+1}\sigma$, then by Lemma 16 (v), the same rule or at least one $\mathcal{C}_\varepsilon$-rule is used in the reduction from $\mathcal{I}_1(t_i\sigma)$ to $\mathcal{I}_1(s_{i+1}\sigma)$. (This would not hold for $\mathcal{I}_2$, cf. Lemma 21 (v).) Thus, then we have $\mathcal{I}_1(t_i\sigma) \succ \mathcal{I}_1(s_{i+1}\sigma)$ since $\succ$ is monotonic. As $\mathcal{R}_{\succ}$-reductions are used for infinitely many i, we have $\mathcal{I}_1(t_i\sigma) \succ \mathcal{I}_1(s_{i+1}\sigma)$ for infinitely many i. Using Lemma 16 (ii), (i), and $s \mathrel{{}_{(}{\succsim}_{)}} t$ for all pairs in $\mathcal{P}$, we obtain $\mathcal{I}_1(s_i\sigma) \to_{\mathcal{C}_\varepsilon}^* s_i\mathcal{I}_1(\sigma) \mathrel{{}_{(}{\succsim}_{)}} t_i\mathcal{I}_1(\sigma) = \mathcal{I}_1(t_i\sigma)$. By $\mathcal{C}_\varepsilon$-compatibility of $\succ$, we get $\mathcal{I}_1(s_i\sigma) \mathrel{{}_{(}{\succsim}_{)}} \mathcal{I}_1(t_i\sigma)$ for all i. This contradicts the well-foundedness of $\succ$. $\square$

Rule removal has three benefits. First, the rules $\mathcal{R}_{\succ}$ do not have to be weakly decreasing anymore after the removal. Second, the rules that $\mathcal{R}_{\succ}$ depends on do not necessarily have to be weakly decreasing anymore either. More precisely, since we only regard chains over $\mathcal{R} \setminus \mathcal{R}_{\succ}$, only the rules in $\Delta_1(\mathcal{P}, \mathcal{R} \setminus \mathcal{R}_{\succ})$ or $\Delta_2(\mathcal{P}, \mathcal{R} \setminus \mathcal{R}_{\succ}, \ldots)$ must be weakly decreasing. And third, it can happen that $\mathcal{P}$ is not a cycle anymore. Then no constraints at all have to be built for $\mathcal{P}$. More precisely, we can delete all edges in the dependency graph between pairs $s \to t$ and $u \to v$ of $\mathcal{P}$ where $s \to t, u \to v$ is an $\mathcal{R}$-chain, but not an $\mathcal{R} \setminus \mathcal{R}_{\succ}$-chain.

Example 24. We extend the TRS of Ex. 18 by the following rules.

$$\mathsf{p}(\mathsf{s}(x)) \to x \qquad \mathsf{plus}(\mathsf{s}(x), y) \to \mathsf{s}(\mathsf{plus}(\mathsf{p}(\mathsf{s}(x)), y)) \qquad \mathsf{plus}(x, \mathsf{s}(y)) \to \mathsf{s}(\mathsf{plus}(x, \mathsf{p}(\mathsf{s}(y))))$$

[10] Using Δ_2 instead of Δ_1 makes Thm. 23 unsound. Consider $\{\mathsf{f}(\mathsf{a}, \mathsf{b}) \to \mathsf{f}(\mathsf{a}, \mathsf{a}), \mathsf{a} \to \mathsf{b}\}$. With $\pi(\mathsf{F}) = [1]$, an LPO-reduction pair makes the filtered dependency pair weakly decreasing and the rule strictly decreasing ($\mathsf{F}(\mathsf{a}) \succsim \mathsf{F}(\mathsf{a})$ and $\mathsf{a} \succ \mathsf{b}$). But then Thm. 23 would state that we can remove the rule and only prove absence of infinite chains of $\mathsf{F}(\mathsf{a}, \mathsf{b}) \to \mathsf{F}(\mathsf{a}, \mathsf{a})$ over the empty TRS. Then we could falsely prove termination.

For the cycle $\{\mathsf{PLUS}(\mathsf{s}(x), y) \to \mathsf{PLUS}(\mathsf{p}(\mathsf{s}(x)), y), \mathsf{PLUS}(x, \mathsf{s}(y)) \to \mathsf{PLUS}(x, \mathsf{p}(\mathsf{s}(y)))\}$ there is no argument filtering and reduction pair $(\succsim, \succ)$ with a quasi-simplification order $\succsim$ satisfying the constraints of Thm. 22. The reason is that due to p's rule, the filtering cannot drop the argument of p. So $\pi(\mathsf{PLUS}(\mathsf{p}(\mathsf{s}(x)), y)) \succsim \pi(\mathsf{PLUS}(\mathsf{s}(x), y))$ and $\pi(\mathsf{PLUS}(x, \mathsf{p}(\mathsf{s}(y)))) \succsim \pi(\mathsf{PLUS}(x, \mathsf{s}(y)))$ hold for any quasi-simplification order $\succsim$. Furthermore, the transformation technique of "narrowing dependency pairs" [2, 10, 12] is not applicable, since the right-hand side of each dependency pair above unifies with the left-hand side of the other dependency pair. Therefore, automated tools based on dependency pairs fail.

In contrast, by Thm. 23 and a reduction pair with the polynomial interpretation $\mathcal{P}ol(\mathsf{PLUS}(x, y)) = x+y$, $\mathcal{P}ol(\mathsf{s}(x)) = x+1$, $\mathcal{P}ol(\mathsf{p}(x)) = x$, p's rule is strictly decreasing and can be removed. Then, p is a constructor. If one uses the technique of "instantiating dependency pairs" [10, 12], for this cycle the second dependency pair can be replaced by $\mathsf{PLUS}(\mathsf{p}(\mathsf{s}(x)), \mathsf{s}(y)) \to \mathsf{PLUS}(\mathsf{p}(\mathsf{s}(x)), \mathsf{p}(\mathsf{s}(y)))$. Now the two pairs form no cycle anymore and thus, no constraints at all are generated.

If we also add the rule $\mathsf{p}(\mathsf{0}) \to \mathsf{0}$, then again $\mathsf{p}(\mathsf{s}(x)) \to x$ can be removed by Thm. 23 but p does not become a constructor and we cannot delete the whole cycle. Still, the resulting constraints are satisfied by an argument filtering with $\pi(\mathsf{PLUS}) = [1, 2]$, $\pi(\mathsf{s}) = \pi(\mathsf{p}) = [\,]$ and an LPO with the precedence $\mathsf{s} > \mathsf{p} > \mathsf{0}$.

Note that here, it is essential that Thm. 23 only requires $l \succsim r$ for rules $l \to r$ that $\mathcal{P}$ depends on. In contrast, previous techniques [15, 23, 30] would demand that all rules including the ones for div and times would have to be at least weakly decreasing. As shown in Ex. 12, this is impossible with standard orders.

To automate Thm. 23, we use reduction pairs $(\succsim, \succ)$ based on linear polynomial interpretations with coefficients from $\{0, 1\}$. Since $\succ$ must be monotonic, n-ary function symbols can only be mapped to $\sum_{i=1}^{n} x_i$ or to $1 + \sum_{i=1}^{n} x_i$. Thus, there are only two possible interpretations resulting in a small search space. Moreover, polynomial orders can solve constraints where one inequality must be strictly decreasing and all others must be weakly decreasing in just one search attempt without backtracking [13]. In this way, Thm. 23 can be applied very efficiently. Since removing rules never complicates termination proofs, Thm. 23 should be applied repeatedly as long as some rule is deleted in each application.

Note that whenever a dependency pair (instead of a rule) is strictly decreasing, one has solved the constraints of Thm. 17 and can delete the cycle. Thus, one should not distinguish between rule- and dependency pair-constraints when applying Thm. 23 and just search for a strict decrease in any of the constraints.

6 Conclusion and Empirical Results

We presented new results to reduce the constraints for termination proofs with dependency pairs substantially. By Sect. 3 and 4, it suffices to require weak decrease of the dependent rules, which correspond to the usable rules regarded for innermost termination. So surprisingly, the constraints for termination and innermost termination are (almost) the same. Moreover, we showed in Sect. 5 that one may pre-process the constraints for each cycle and eliminate rules that are strictly decreasing. All our results can also be used together with dependency pair transformations [2, 10, 12] which often simplify (innermost) termination proofs.

We implemented our results in the system AProVE[11] [14] and tested it on the 130 terminating TRSs from [3, 8, 25]. The following table gives the percentage of the examples where termination could be proved within a timeout of 30 s and the time for running the system on all examples (including the ones where the proof failed). Our experiments were performed on a Pentium IV with 2.4 GHz and 1 GB memory. We used reduction pairs based on the embedding order, LPO, and linear polynomial interpretations with coefficients from $\{0, 1\}$ ("Polo"). The table shows that with every refinement from Thm. 7 to Thm. 22, termination proving becomes more powerful and for more complex orders than embedding, efficiency also increases considerably. Moreover, a pre-processing with Thm. 23 using "Polo" makes the approach even more powerful. Finally, if one also uses dependency pair transformations ("tr"), one can increase power further. To measure the effect of our contributions, in the first 3 rows we did not use the technique for innermost termination proofs, even if the TRS is non-overlapping. (If one applies the innermost termination technique in these examples, we can prove termination of 95 % of the examples in 23 s with "Polo".) Finally, in the last row ("Inn") we verified *innermost* termination with "Polo" and usable rules $\mathcal{U}(\mathcal{P})$ as in Thm. 17, with usable rules $\mathcal{U}(\mathcal{P}, \pi)$ as in Thm. 22, with a pre-processing as in Thm. 23, and with dependency pair transformations. This row demonstrates that termination is now almost as easy to prove as innermost termination. To summarize, our experiments show that the contributions of this paper are indeed relevant and successful in practice, since the reduction of constraints makes automated termination proving significantly more powerful and faster.

	Thm. 7	Thm. 11	Thm. 17	Thm. 22	Thm. 22, 23	Thm. 22, 23, tr
Emb	39 s, 28 %	7 s, 30 %	42 s, 38 %	50 s, 52 %	51 s, 65 %	82 s, 78 %
LPO	606 s, 51 %	569 s, 54 %	261 s, 59 %	229 s, 61 %	234 s, 75 %	256 s, 84 %
Polo	9 s, 61 %	8 s, 66 %	5 s, 73 %	5 s, 78 %	6 s, 85 %	9 s, 91 %
Inn			8 s, 78 %	8 s, 82 %	10 s, 88 %	31 s, 97 %

References

1. T. Arts. System description: The dependency pair method. In L. Bachmair, editor, *Proc. 11th RTA*, LNCS 1833, pages 261–264, Norwich, UK, 2000.
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[12], RWTH Aachen, 2001.
4. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge, 1998.
5. C. Borralleras, M. Ferreira, and A. Rubio. Complete monotonic semantic path orderings. In D. McAllester, editor, *Proc. 17th CADE*, LNAI 1831, pages 346–364, Pittsburgh, PA, USA, 2000.

[11] `http://www-i2.informatik.rwth-aachen.de/AProVE`. Our contributions are integrated in AProVE 1.1-beta, which does not yet contain all options of AProVE 1.0.

6. E. Contejean, C. Marché, B. Monate, and X. Urbain. CiME. http://cime.lri.fr.
7. N. Dershowitz. Termination of rewriting. *J. Symb. Comp.*, 3:69–116, 1987.
8. N. Dershowitz. 33 examples of termination. In *Proc. French Spring School of Theoretical Computer Science*, LNCS 909, pages 16–26, Font Romeux, 1995.
9. O. Fissore, I. Gnaedig, and H. Kirchner. Cariboo: An induction based proof tool for termination with strategies. In C. Kirchner, editor, *Proc. 4th PPDP*, pages 62–73, Pittsburgh, PA, USA, 2002. ACM Press.
10. J. Giesl and T. Arts. Verification of Erlang processes by dependency pairs. *Appl. 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 Vardi and Voronkov, editors, *Proc 10th LPAR*, LNAI 2850, 165–179, 2003.
13. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing dependency pairs. Technical Report AIB-2003-08[12], RWTH Aachen, Germany, 2003.
14. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated termination proofs with AProVE. In v. Oostrom, editor, *Proc. 15th RTA*, LNCS, Aachen, 2004.
15. J. Giesl and H. Zantema. Liveness in rewriting. In R. Nieuwenhuis, editor, *Proc. 14th RTA*, LNCS 2706, pages 321–336, Valencia, Spain, 2003.
16. B. Gramlich. Generalized sufficient conditions for modular termination of rewriting. *Appl. Algebra in Engineering, Communication & Computing*, 5:131–158, 1994.
17. B. Gramlich. Abstract relations between restricted termination and confluence properties of rewrite systems. *Fundamenta Informaticae*, 24:3–23, 1995.
18. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. In F. Baader, editor, *Proc. 19th CADE*, LNAI 2741, Miami Beach, FL, USA, 2003.
19. N. Hirokawa and A. Middeldorp. Tsukuba termination tool. In R. Nieuwenhuis, editor, *Proc. 14th RTA*, LNCS 2706, pages 311–320, Valencia, Spain, 2003.
20. S. Kamin and J. J. Lévy. Two generalizations of the recursive path ordering. Unpublished Manuscript, University of Illinois, IL, USA, 1980.
21. D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. 1970.
22. K. Kusakari, M. Nakamura, and Y. Toyama. Argument filtering transformation. In G. Nadathur, editor, *Proc. 1st PPDP*, LNCS 1702, pages 48–62, Paris, 1999.
23. D. Lankford. On proving term rewriting systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.
24. E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, 2002.
25. J. Steinbach. Automatic termination proofs with transformation orderings. In J. Hsiang, editor, *Proc. 6th RTA*, LNCS 914, pages 11–25, Kaiserslautern, Germany, 1995. Full version in Technical Report SR-92-23, Universität Kaiserslautern.
26. J. Steinbach. Simplification orderings: History of results. *Fund. I.*, 24:47–87, 1995.
27. R. Thiemann and J. Giesl. Size-change termination for term rewriting. In R. Nieuwenhuis, editor, *Proc. 14th RTA*, LNCS 2706, pages 264–278, Valencia, Spain, 2003.
28. Y. Toyama. Counterexamples to the termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, 1987.
29. X. Urbain. Automated incremental termination proofs for hierarchically defined term rewriting systems. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proc. IJCAR 2001*, LNAI 2083, pages 485–498, Siena, Italy, 2001.
30. H. Zantema. TORPA: Termination of rewriting proved automatically. In *Proc. 15th RTA*, LNCS, Aachen, 2004. Full version in TU/e CS-Report 03-14, TU Eindhoven.

[12] Available from http://aib.informatik.rwth-aachen.de