RWTH Aachen

Department of Computer Science
Technical Report

Mechanizing Dependency Pairs

Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2003-8

RWTH Aachen · Department of Computer Science · December 2003

Mechanizing Dependency Pairs

Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke

LuFG Informatik II, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany
{giesl|thiemann|psk}@informatik.rwth-aachen.de
spf@i2.informatik.rwth-aachen.de

Abstract. The dependency pair approach [2, 13, 14] is a powerful technique for automated termination and innermost termination proofs of term rewrite systems (TRSs). For any TRS, it generates inequality constraints that have to be satisfied by well-founded orders. We improve the dependency pair approach by considerably reducing the number of constraints produced for (innermost) termination proofs.
Moreover, we extend transformation techniques to manipulate dependency pairs which simplify (innermost) termination proofs significantly. In order to fully mechanize the approach, we show how transformations and the search for suitable orders can be mechanized efficiently. We implemented our results in the automated termination prover AProVE and evaluated them on large collections of examples.

1 Introduction

Termination is an essential property of term rewrite systems. Most traditional methods to prove termination of TRSs (automatically) use *simplification orders* [8, 33], where a term is greater than its proper subterms (*subterm property*). Examples for simplification orders include lexicographic or recursive path orders RPOS possibly with status [7, 23], the Knuth-Bendix order KBO [24], and (most) polynomial orders [26]. However, there are numerous important TRSs which are not *simply terminating*, i.e., their termination cannot be shown by simplification orders. Therefore, the *dependency pair* approach [2, 13, 14] was developed which allows the application of simplification orders to non-simply terminating TRSs. In this way, the class of systems where termination is provable mechanically increases significantly.

Example 1 The following TRS [2] is not simply terminating, since the left-hand side of quot's last rule is embedded in the right-hand side if y is instantiated with $\mathsf{s}(x)$. So classical approaches for termination proofs fail, while the example is easy to handle with dependency pairs.

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

In Sect. 2, we recapitulate the dependency pair approach for termination and innermost termination proofs (i.e., one tries to show that all reductions are finite, where in the innermost termination case, one only considers reductions of innermost redexes) and discuss first new refinements. Then we show that the approach can be improved significantly by reducing the constraints for termination

(Sect. 3) and innermost termination (Sect. 4). Sect. 5 introduces new conditions for transforming dependency pairs by narrowing, rewriting, and instantiation in order to simplify (innermost) termination proofs further.

The remainder of the paper is concerned with mechanizing dependency pairs. To this end, we show how to solve the indeterminisms and search problems of the dependency pair approach efficiently. One problem is the question when and how often to apply the dependency pair transformations discussed above. Therefore, in Sect. 5 we also show how to use these transformations in practice in order to guarantee the termination of their application without compromising their power.

For automated (innermost) termination proofs, one tries to solve the constraints generated by the dependency pair approach with standard simplification orders. However, if one uses orders like RPOS or KBO, then the constraints should first be pre-processed by an *argument filtering* in order to benefit from the full power of dependency pairs. Since the number of possible argument filterings is exponential, the search for a suitable filtering is one of the main problems when automating dependency pairs. We present an algorithm to generate argument filterings efficiently for our improved dependency pair approach (Sect. 6) and discuss heuristics to increase its efficiency in Sect. 7. Instead of using orders like RPOS or KBO in combination with argument filterings, one can also apply polynomial orders, which already simulate the concept of argument filtering themselves. In Sect. 8 we show how to mechanize the dependency pair approach using polynomial orders efficiently.

Our improvements and algorithms are implemented in our termination prover AProVE. In Sect. 9 we give empirical results which show that they are extremely successful in practice. Thus, the contributions of this paper are also very helpful for other tools based on dependency pairs (e.g., [1], CiME [6], TTT [21]). Moreover, we conjecture that they can also be used in other recent approaches for termination [5, 11] which have several aspects in common with dependency pairs. Finally, dependency pairs can be combined with other termination techniques (e.g., in [34] we integrated dependency pairs and the *size-change principle* from termination analysis of functional [27] and logic programs [10]). Moreover, the system TALP [31] uses dependency pairs for termination proofs of logic programs. Thus, techniques to mechanize and to improve dependency pairs are also useful for termination analysis of other kinds of programming languages.

2 Dependency Pairs

We briefly present the *dependency pair* approach of Arts and Giesl and refer to [2, 13, 14] 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 root symbols 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. 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 $g^\sharp(t_1, \ldots, t_m)$.

Definition 2 (Dependency Pair) *If $l \to r \in \mathcal{R}$ and t is a subterm of r with defined root symbol, then the rule $l^\sharp \to t^\sharp$ is a* dependency pair *of $\mathcal{R}$. The set of all dependency pairs of $\mathcal{R}$ is denoted by $DP(\mathcal{R})$.*

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

$$\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{MINUS}(x, y) \tag{1}$$

$$\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{MINUS}(x, y) \tag{2}$$

$$\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{QUOT}(\mathsf{minus}(x, y), \mathsf{s}(y)) \tag{3}$$

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

Definition 3 ($\mathcal{R}$-Chain) *A sequence of dependency pairs $s_1 \to t_1$, $s_2 \to t_2, \ldots$ is an* $\mathcal{R}$-chain *if there exists a substitution σ such that $t_j\sigma \to_{\mathcal{R}}^* s_{j+1}\sigma$ for every two consecutive pairs $s_j \to t_j$ and $s_{j+1} \to t_{j+1}$ in the sequence. Such a chain is an* innermost $\mathcal{R}$-chain *if $t_j\sigma \xrightarrow{\mathsf{i}}{}_{\mathcal{R}}^* s_{j+1}\sigma$ and if $s_j\sigma$ is a normal form for all j.*

For instance in Ex. 1, the following sequence is an (innermost) chain.

$$\mathsf{QUOT}(\mathsf{s}(x_1), \mathsf{s}(y_1)) \to \mathsf{QUOT}(\mathsf{minus}(x_1, y_1), \mathsf{s}(y_1)),$$
$$\mathsf{QUOT}(\mathsf{s}(x_2), \mathsf{s}(y_2)) \to \mathsf{QUOT}(\mathsf{minus}(x_2, y_2), \mathsf{s}(y_2))$$

since $\mathsf{QUOT}(\mathsf{minus}(x_1, y_1), \mathsf{s}(y_1))\sigma \to_{\mathcal{R}}^* \mathsf{QUOT}(\mathsf{s}(x_2), \mathsf{s}(y_2))\sigma$ holds for a suitable substitution σ. For example, σ could instantiate x_1 with $\mathsf{s}(\mathsf{0})$ and y_1, x_2, y_2 with $\mathsf{0}$. While there are chains of arbitrary finite length in Ex. 1, we have no infinite chains. We obtain the following sufficient and necessary criterion for termination and innermost termination.

Theorem 4 (Termination Criterion) *A TRS $\mathcal{R}$ is terminating iff there is no infinite chain. $\mathcal{R}$ is innermost terminating iff there is no infinite innermost chain.*

For (innermost) termination proofs by Thm. 4, one shows that there is no infinite (innermost) chain. To this end, one generates constraints which should be satisfied by some *reduction pair* $(\succsim, \succ)$ [25] consisting of a quasi-rewrite order $\succsim$ (i.e., $\succsim$ is reflexive, transitive, monotonic (closed under contexts), and stable (closed under substitutions)) and a stable well-founded order $\succ$ which is compatible with $\succsim$ (i.e., $\succsim \circ \succ \subseteq \succ$ and $\succ \circ \succsim \subseteq \succ$). However, $\succ$ need not be monotonic. The constraints ensure that all rewrite rules are weakly decreasing (w.r.t. $\succsim$) and all dependency pairs are strictly decreasing (w.r.t. $\succ$). Requiring $l \succsim r$ for all $l \to r \in \mathcal{R}$ ensures that in a chain $s_1 \to t_1, s_2 \to t_2, \ldots$ with $t_j\sigma \to_{\mathcal{R}}^* s_{j+1}\sigma$, we have $t_j\sigma \succsim s_{j+1}\sigma$ for all j. However for innermost termination, a weak decrease is not required for all rules but only for the *usable rules*. These rules are a superset of those rules that may be used to reduce right-hand sides of dependency pairs if their variables are instantiated with normal forms. In Ex. 1, the usable rules are the minus-rules. Of course, rules $l \to r$ where l contains a redex as a proper subterm should never be included in the usable rules. But since such rules cannot be used in innermost reductions anyway, they should be eliminated in a pre-processing step before attempting the innermost termination proof.

Definition 5 (Usable Rules) *For* $f \in \mathcal{F}$*, let* $Rls_{\mathcal{R}}(f) = \{l \to r \in \mathcal{R} \mid \mathrm{root}(l) = f\}$ *and let* $\mathcal{R}' = \mathcal{R} \setminus Rls_{\mathcal{R}}(f)$*. For any term, we define*

- $\mathcal{U}_{\mathcal{R}}(x) = \varnothing$ *for* $x \in \mathcal{V}$ *(where* $\mathcal{V}$ *is the set of all variables) and*
- $\mathcal{U}_{\mathcal{R}}(f(t_1, ..., t_n)) = Rls_{\mathcal{R}}(f) \ \cup \ \bigcup_{l \to r \in Rls_{\mathcal{R}}(f)} \mathcal{U}_{\mathcal{R}'}(r) \ \cup \ \bigcup_{j=1}^{n} \mathcal{U}_{\mathcal{R}'}(t_j)$.

The above definition of usable rules from [2] can be improved further by taking into account that in innermost chains $s_1 \to t_1, s_2 \to t_2, \ldots$, we only regard substitutions σ such that all subterms of $s_j\sigma$ are in normal form. Thus, subterms of t_j which also occur in the corresponding left component s_j may be ignored when computing the usable rules. This leads to the following new improved definition of usable rules.

Definition 6 (Improved Usable Rules) *Let* s *and* t *be terms with* $\mathcal{V}(t) \subseteq \mathcal{V}(s)$*. Then* $\mathcal{U}^s_{\mathcal{R}}(t) = \varnothing$ *if* t *is a subterm of* s *and otherwise*

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

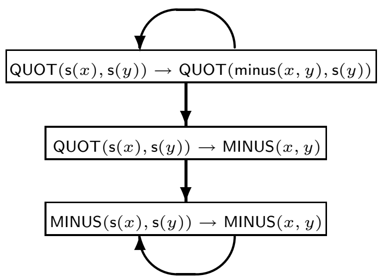
For any set $\mathcal{P}$ *of dependency pairs, we define* $\mathcal{U}_{\mathcal{R}}(\mathcal{P}) = \bigcup_{s \to t \in \mathcal{P}} \mathcal{U}^s_{\mathcal{R}}(t)$*.*

Example 7 The following TRS illustrates the improved usable rules.

$$\begin{array}{rclcrcl} \mathsf{f}(\mathsf{a}, \mathsf{h}(x)) & \to & \mathsf{f}(\mathsf{g}(x), \mathsf{h}(x)) & \qquad & \mathsf{h}(\mathsf{g}(x)) & \to & \mathsf{h}(\mathsf{a}) \\ \mathsf{g}(\mathsf{h}(x)) & \to & \mathsf{g}(x) & & \mathsf{h}(\mathsf{h}(x)) & \to & x \end{array}$$

Since the term $\mathsf{F}(\mathsf{g}(x), \mathsf{h}(x))$ from the right-hand side of the first dependency pair contains g and h, $\mathcal{U}_{\mathcal{R}}(\mathsf{F}(\mathsf{g}(x), \mathsf{h}(x)))$ consists of the g- and h-rules. On the other hand, $\mathcal{U}^{\mathsf{F}(\mathsf{a},\mathsf{h}(x))}_{\mathcal{R}}(\mathsf{F}(\mathsf{g}(x), \mathsf{h}(x)))$ only contains the g-rules, since $\mathsf{h}(x)$ is a subterm of the dependency pair's left-hand side $\mathsf{F}(\mathsf{a}, \mathsf{h}(x))$. Thus, with the improved usable rules, a weak decrease of the h-rules is no longer required. We show in Ex. 13 that this improvement is indeed necessary for the success of the innermost termination proof.

To estimate which dependency pairs may occur consecutively in (innermost) chains, we build an (innermost) *dependency graph*. Its nodes are the dependency pairs and there is an arc from $s \to t$ to $v \to w$ iff $s \to t$, $v \to w$ is an (innermost) chain. In Ex. 1, we have the following dependency graph that is identical to the innermost dependency graph.



Since it is undecidable whether two dependency pairs form an (innermost) chain, for automation we construct *estimated* graphs such that all arcs in the real graph are also arcs in the estimated graph. Let $\text{CAP}(t)$ result from replacing all subterms of t with defined root symbol by different fresh variables. Here, multiple occurrences of the same subterm with defined root are also replaced by pairwise different fresh variables. Let $\textsc{ren}(t)$ result from replacing all occurrences of variables in t by different fresh variables (i.e., $\textsc{ren}(t)$ is a linear term). Hence, $\textsc{cap}(\mathsf{QUOT}(\mathsf{minus}(x,y),\mathsf{s}(y))) = \mathsf{QUOT}(z,\mathsf{s}(y))$ and $\textsc{ren}(\mathsf{QUOT}(x,x)) = \mathsf{QUOT}(x_1,x_2)$. We define $\textsc{cap}^s$ like $\textsc{cap}$ except that subterms with defined root that already occur in s are not replaced by new variables.

Definition 8 (Estimated (innermost) dependency graph) *The* estimated dependency graph $\mathrm{EDG}(\mathcal{R})$ *of a TRS $\mathcal{R}$ is the directed graph whose nodes are the dependency pairs and there is an arc from the pair $s \to t$ to $v \to w$ iff* $\textsc{ren}(\textsc{cap}(t))$ *and v are unifiable. In the* estimated innermost dependency graph $\mathrm{EIDG}(\mathcal{R})$ *there is an arc from $s \to t$ to $v \to w$ iff* $\textsc{cap}^s(t)$ *and v are unifiable by a most general unifier (mgu) μ such that $s\mu$ and $v\mu$ are in normal form.*

Of course, to check whether there is an arc from $s \to t$ to $v \to w$ in E(I)DG, one has to rename their variables to make them variable disjoint. In Ex. 1, the estimated dependency graph and the estimated innermost dependency graph are identical to the real dependency graph.

In [19, 20, 28, 29], the above estimations of [2] were refined. While the approximations of [28] are based on computationally expensive tree automata techniques, the following refinements of EDG and EIDG from [19, 20, 29] can be computed efficiently. In order to draw an arc from $s \to t$ to $v \to w$, apart from checking the unifiability of a modification of t with v, one also requires the unifiability of a modification of v with t. To this end, one considers the reversal of $\mathcal{R}$-rules. More precisely, for any set of non-collapsing rules $\mathcal{R}' \subseteq \mathcal{R}$, let $\textsc{cap}^{-1}_{\mathcal{R}'}(v)$ result from replacing all subterms of v that have a root symbol from $\{\mathrm{root}(r) \mid l \to r \in \mathcal{R}'\}$ by different fresh variables. If $\mathcal{R}'$ is collapsing, then $\textsc{cap}^{-1}_{\mathcal{R}'}(v)$ is a fresh variable. As usual, a rule $l \to r$ is called *collapsing* if $r \in \mathcal{V}$.

Definition 9 ($\mathrm{E(I)DG}^*$ **[19, 20, 29])** $\mathrm{EDG}^*(\mathcal{R})$ *is the graph whose nodes are the dependency pairs and there is an arc from $s \to t$ to $v \to w$ iff both* $\textsc{ren}(\textsc{cap}(t))$ *and v are unifiable and* $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{R}}(v))$ *and t are unifiable. In* $\mathrm{EIDG}^*(\mathcal{R})$ *there is an arc from $s \to t$ to $v \to w$ iff both* $\textsc{cap}^s(t)$ *and v are unifiable by a mgu μ_1 and* $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}_{\mathcal{R}}(t)}(v))$ *and w are unifiable by a mgu μ_2, where $s\mu_1$, $v\mu_1$, $s\mu_2$ are in normal form.*[1]

$\mathrm{E(I)DG}^*(\mathcal{R})$ contains the real (innermost) dependency graph and it is a subgraph of $\mathrm{E(I)DG}(\mathcal{R})$ [19, 20, 29]. Thus, $\mathrm{E(I)DG}^*$ is a better approximation than E(I)DG. Since EIDG^* is based on the computation of usable rules, our new improved usable rules of Def. 6 lead to the following refined approximation EIDG^{**} which results from replacing $\mathcal{U}_{\mathcal{R}}(t)$ by $\mathcal{U}^s_{\mathcal{R}}(t)$. It is straightforward to show that $\mathrm{EIDG}^{**}(\mathcal{R})$ still contains the real innermost dependency graph and it is a subgraph of $\mathrm{EIDG}^*(\mathcal{R})$. Thus, now EDG^* and EIDG^{**} are the best known approximations of (innermost) dependency graphs that can be computed efficiently. Moreover, in Sect. 5 we will show that in contrast to EIDG^*, EIDG^{**} is well suited for a combination with dependency pair transformations.

[1] Note that it is useless to require $v\mu_2$ to be a normal form, since the terms $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}_{\mathcal{R}}(t)}(v))$ and v are variable disjoint.

Definition 10 (EIDG**) *In* EIDG$^{**}(\mathcal{R})$ *there is an arc from* $s \to t$ *to* $v \to w$ *iff* $\text{CAP}^s(t)$ *and* v *unify by a mgu* μ_1 *and* $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v))$ *and* t *unify by a mgu* μ_2*, where* $s\mu_1$*,* $v\mu_1$*,* $s\mu_2$ *are in normal form.*

The advantage of EIDG* over EIDG** is illustrated by the TRS $\mathcal{R}$ from Ex. 7. The dependency pair $\mathsf{F}(\mathsf{a}, \mathsf{h}(x)) \to \mathsf{F}(\mathsf{g}(x), \mathsf{h}(x))$ forms a cycle in EIDG$^*(\mathcal{R})$, whereas this is not the case in EIDG$^{**}(\mathcal{R})$. The reason is that for $t = \mathsf{F}(\mathsf{g}(x), \mathsf{h}(x))$ and $v = \mathsf{F}(\mathsf{a}, \mathsf{h}(x'))$, $\mathcal{U}_{\mathcal{R}}(t)$ is collapsing and thus, $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}_{\mathcal{R}}(t)}(v))$ is a fresh variable, whereas $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v)) = \mathsf{F}(\mathsf{a}, \mathsf{h}(y))$ does not unify with t.

A set $\mathcal{P} \neq \varnothing$ of dependency pairs is a *cycle* if for any two 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 Ex. 1, we have the cycles $\mathcal{P}_1 = \{(1)\}$ and $\mathcal{P}_2 = \{(3)\}$ of the MINUS- and the QUOT-pair. Since we only regard finite TRSs, any infinite (innermost) chain of dependency pairs corresponds to a cycle in the (innermost) dependency graph.

To show (innermost) termination of TRSs, one can now prove absence of infinite (innermost) chains separately for every cycle of the (innermost) dependency graph. In particular, for every cycle one generates constraints separately and one can search for different reduction pairs satisfying the constraints. Moreover, it suffices if in every cycle $\mathcal{P}$ only one dependency pair is strictly decreasing, whereas the others only have to be weakly decreasing. For automation, one may use any of the estimations discussed above instead of the real graph.

We want to use standard techniques to synthesize reduction pairs satisfying the constraints of the dependency pair approach. Most existing techniques generate monotonic *orders* $\succ$ like RPOS or KBO. But for the dependency pair approach we only need a monotonic *quasi-order* $\succsim$, whereas $\succ$ does not have to be monotonic. (This is often called "*weak* monotonicity".) For that reason, before synthesizing a suitable order, some 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] (we use the notation of [25]).

Definition 11 (Argument Filtering) *An* argument filtering π *for a signature* $\mathcal{F}$ *maps every* n*-ary function symbol to an argument position* $i \in \{1, \dots, n\}$ *or to a (possibly empty) list* $[i_1, \dots, i_m]$ *of argument positions with* $1 \leq i_1 < \dots < i_m \leq n$*. The signature* $\mathcal{F}_\pi$ *consists of all function symbols* f *such that* $\pi(f) = [i_1, \dots, i_m]$*, where in* $\mathcal{F}_\pi$ *the arity of* f *is* m*. Every argument filtering* π *induces a mapping from terms over* $\mathcal{F}$ *to terms over* $\mathcal{F}_\pi$*. This mapping is also denoted by* π*:*

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

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

Now dependency pairs can be automated as follows. Here, we always use argument filterings for the signature $\mathcal{F} \cup \mathcal{F}^\sharp$.

Theorem 12 (Automating Dependency Pairs) *A TRS $\mathcal{R}$ is terminating iff for any cycle $\mathcal{P}$ of the dependency graph, there is a reduction pair $(\succsim, \succ)$ and an argument filtering π such that both*

(a) $\pi(s) \succ \pi(t)$ *for one dependency pair* $s \to t$ *from* $\mathcal{P}$ *and*
$\pi(s) \succsim \pi(t)$ *or* $\pi(s) \succ \pi(t)$ *for all other pairs* $s \to t$ *from* $\mathcal{P}$
(b) $\pi(l) \succsim \pi(r)$ *for all* $l \to r \in \mathcal{R}$

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

(c) $\pi(s) \succ \pi(t)$ *for one dependency pair* $s \to t$ *from* $\mathcal{P}$ *and*
$\pi(s) \succsim \pi(t)$ *or* $\pi(s) \succ \pi(t)$ *for all other pairs* $s \to t$ *from* $\mathcal{P}$
(d) $\pi(l) \succsim \pi(r)$ *for all* $l \to r \in \mathcal{U}_{\mathcal{R}}(\mathcal{P})$

So in Ex. 1, we obtain the following 10 constraints for termination. Here, $(\succsim_i, \succ_i)$ is the reduction pair and π_i is the argument filtering for cycle $\mathcal{P}_i$, where $i \in \{1, 2\}$.

$$\pi_1(\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y))) \succ_1 \pi_1(\mathsf{MINUS}(x, y)) \tag{4}$$
$$\pi_2(\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y))) \succ_2 \pi_2(\mathsf{QUOT}(\mathsf{minus}(x, y), \mathsf{s}(y))) \tag{5}$$
$$\pi_i(\mathsf{minus}(x, \mathsf{0})) \succsim_i \pi_i(x) \tag{6}$$
$$\pi_i(\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y))) \succsim_i \pi_i(\mathsf{minus}(x, y)) \tag{7}$$
$$\pi_i(\mathsf{quot}(\mathsf{0}, \mathsf{s}(y))) \succsim_i \pi_i(\mathsf{0}) \tag{8}$$
$$\pi_i(\mathsf{quot}(\mathsf{s}(x), \mathsf{s}(y))) \succsim_i \pi_i(\mathsf{s}(\mathsf{quot}(\mathsf{minus}(x, y), \mathsf{s}(y)))) \tag{9}$$

The argument filtering $\pi_i(\mathsf{minus}) = [1]$ replaces all terms $\mathsf{minus}(t_1, t_2)$ by $\mathsf{minus}(t_1)$. With this filtering, (4) – (9) are satisfied by the lexicographic path order (LPO) with the precedence $\mathsf{quot} > \mathsf{s} > \mathsf{minus}$. So termination of the TRS is proved. Similarly, one could also use a collapsing filtering $\pi_i(\mathsf{minus}) = \pi_i(\mathsf{quot}) = 1$ which replaces all terms $\mathsf{minus}(t_1, t_2)$ or $\mathsf{quot}(t_1, t_2)$ by t_1. Then even the embedding order orients the resulting constraints.

For innermost termination, we only obtain the constraint (4) for the cycle $\mathcal{P}_1$, since it has no usable rules. For $\mathcal{P}_2$, the constraints (8) and (9) are not necessary, since the quot-rules are not usable for any right-hand side of a dependency pair. In general, the constraints for innermost termination are always a subset of the constraints for termination. So for classes of TRSs where innermost termination already implies termination (e.g., non-overlapping TRSs) [17], one should always use the approach for innermost termination when attempting termination proofs.

Example 13 The TRS from Ex. 7 shows that the improved usable rules resp. our improved approximation EIDG** of innermost dependency graphs can be necessary for the success of the innermost termination proof. If one uses the original definition of usable rules (Def. 5) and EIDG* or EIDG, then the pair $\mathsf{F}(\mathsf{a}, \mathsf{h}(x)) \to \mathsf{F}(\mathsf{g}(x), \mathsf{h}(x))$ forms a cycle and the g- and h-rules are usable. Thus, one would have to find an argument filtering π and a reduction pair $(\succsim, \succ)$ such that

$$\pi(\mathsf{F}(\mathsf{a}, \mathsf{h}(x))) \succ \pi(\mathsf{F}(\mathsf{g}(x), \mathsf{h}(x))) \quad (10) \qquad \pi(\mathsf{h}(\mathsf{g}(x))) \succsim \pi(\mathsf{h}(\mathsf{a})) \quad (12)$$
$$\pi(\mathsf{g}(\mathsf{h}(x))) \succsim \pi(\mathsf{g}(x)) \quad (11) \qquad \pi(\mathsf{h}(\mathsf{h}(x))) \succsim x \quad (13)$$

However, there exists no argument filtering such that these constraints are satisfiable by standard reduction pairs which are based on RPOS, KBO, or polynomial orders. For RPOS, constraint (10) implies that a must be greater than g in the precedence and since the argument filtering may not eliminate the argument of h by constraint (13), this precedence contradicts constraint (12). Similarly for KBO, constraint (10) implies that the argument filtering must remove g's argument and that either a must have greater weight than g or their weight is equal and a is greater than g in the precedence. But again, this contradicts constraint (12). Finally, for polynomial orders, constraint (10) implies $\mathsf{a} \succ \mathsf{g}(\mathsf{a})$, since polynomial orders are total on ground terms. But since h may not be mapped to a constant polynomial by constraint (13), this would contradict constraint (12). Thus, with common orders that are amenable to automation, the innermost termination proof fails when using the classical usable rules of Def. 5.

In contrast, with the improved usable rules of Def. 6 one would not obtain the constraints (12) and (13). The remaining constraints are satisfied by an LPO with the precedence $\mathsf{a} > \mathsf{g}$ if one uses the argument filtering π with $\pi(\mathsf{g}) = [\,]$. Even better, when using improved usable rules already in the graph approximation EIDG**, then one can detect that the graph has no arcs and one does not obtain any constraint at all.

As shown in [20], to implement Thm. 12, one does not compute all cycles, but only maximal cycles (*strongly connected components, SCCs*) that are not contained in other cycles. When solving the constraints of Thm. 12 for an SCC, the reduction pair $(\succsim, \succ)$ and argument filtering π may satisfy the strict constraint $\pi(s) \succ \pi(t)$ for *several* dependency pairs $s \to t$ in the SCC. Thus, all subcycles of the SCC containing such a strictly decreasing dependency pair do not have to be considered anymore. So after solving the constraints for the initial SCCs, all strictly decreasing dependency pairs are removed and one now builds SCCs from the remaining dependency pairs, which were just weakly decreasing up to now, etc.

3 Improved Termination Proofs

Now we improve the technique for automated termination proofs (Thm. 12). Urbain [36] showed how to use dependency pairs for modular termination proofs of hierarchical combinations of C_ε-terminating TRSs. $\mathcal{R}$ is *C_ε-terminating* [16] iff $\mathcal{R} \cup C_\varepsilon$ terminates where $C_\varepsilon = \{c(x, y) \to x, c(x, y) \to y\}$ for a fresh symbol c not occurring in $\mathcal{R}$. However in the results of [36], Urbain did not integrate the consideration of cycles in dependency graphs and required all dependency pairs to be strictly decreasing. Thm. 14 extends his modularity results by combining them with cycles. This leads to an improvement for termination proofs with dependency pairs which can be used for TRSs in general. The advantage is that the set of constraints (b) in Thm. 12 is reduced significantly.

The crucial idea of [36] is to consider the recursion hierarchy of function symbols. A symbol f *depends on* the symbol h (denoted $f \geq_d h$) if $f = h$ or if there exists a symbol g such that g occurs in an f-rule and g depends on h. We define $>_d = \geq_d \setminus \leq_d$ and $\sim_d = \geq_d \cap \leq_d$. So $f \sim_d g$ means that f and g are mutually recursive. If $\mathcal{R} = \mathcal{R}_1 \uplus \ldots \uplus \mathcal{R}_n$ and $f \sim_d g$ iff $Rls_{\mathcal{R}}(f) \cup Rls_{\mathcal{R}}(g) \subseteq \mathcal{R}_i$, then we call $\mathcal{R}_1, \ldots, \mathcal{R}_n$ a *separation* of $\mathcal{R}$. Moreover, we extend $\geq_d$ to the sets $\mathcal{R}_i$ by defining $\mathcal{R}_i \geq_d \mathcal{R}_j$ iff $f \geq_d g$ for all f, g with $Rls_{\mathcal{R}}(f) \subseteq \mathcal{R}_i$ and $Rls_{\mathcal{R}}(g) \subseteq \mathcal{R}_j$. For any i, let $\mathcal{R}'_i$ denotes the rules that $\mathcal{R}_i$ depends on, i.e., $\mathcal{R}'_i = \bigcup_{\mathcal{R}_i \geq_d \mathcal{R}_j} \mathcal{R}_j$.

Clearly, a cycle can only consist of dependency pairs from some $\mathcal{R}_i$. So in Thm. 12 we only have to consider cycles with $\mathcal{P} \subseteq DP(\mathcal{R}_i)$. But to detect these cycles $\mathcal{P}$, we still have to regard the dependency graph of the *whole* TRS $\mathcal{R}$ and not just the dependency graph of the subsystem $\mathcal{R}'_i$. The reason is that we have to consider $\mathcal{R}$-chains instead of just $\mathcal{R}'_i$-chains. To see this, regard Toyama's TRS [35] where $\mathcal{R}_1 = \mathcal{R}'_1 = \{\mathsf{f}(0, 1, x) \to \mathsf{f}(x, x, x)\}$ and $\mathcal{R}_2 = \mathcal{R}'_2 = \{\mathsf{g}(x, y) \to x, \mathsf{g}(x, y) \to y\}$. $\mathcal{R}'_1$'s and $\mathcal{R}'_2$'s dependency graphs have no arcs, whereas the dependency graph of $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ has a cycle. Hence, if one only considers the graphs of $\mathcal{R}'_1$ and $\mathcal{R}'_2$, one could falsely prove termination.

Our improvement for termination proofs can be combined with any estimation technique for dependency graphs. If one uses the EDG-estimation of Def. 8, to detect the cycles with $\mathcal{P} \subseteq DP(\mathcal{R}_i)$, it indeed suffices to regard the *estimated* dependency graph EDG($\mathcal{R}'_i$) instead of EDG($\mathcal{R}$). The reason is that then all cycles of $\mathcal{R}$'s dependency graph with $\mathcal{P} \subseteq DP(\mathcal{R}_i)$ are also cycles of EDG($\mathcal{R}'_i$) . But for other estimations like EDG* or the technique of [28] this is not the case.

When handling a cycle with nodes from $DP(\mathcal{R}_i)$, the constraints (b) of Thm. 12 require $\pi(l) \gtrsim \pi(r)$ for all rules $l \to r \in \mathcal{R}$. The improvement of Thm. 14 states that instead, it suffices to demand $\pi(l) \gtrsim \pi(r)$ only for rules $l \to r$ that $\mathcal{R}_i$ depends on, i.e., for rules from $\mathcal{R}'_i$. So in the termination proof of Ex. 1, $\pi(l) \gtrsim \pi(r)$ does not have to be required for the quot-rules when regarding the cycle $\mathcal{P}_1 = \{\mathsf{MINUS}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{MINUS}(x, y)\}$. The reason is that for every minimal[2] chain $s_1 \to t_1, s_2 \to t_2, \ldots$ of $\mathcal{R}_i$-dependency pairs, there exists a substitution σ such that $t_j\sigma$ reduces to $s_{j+1}\sigma$ using only the rules of $\mathcal{R}'_i \cup C_\varepsilon$, cf. [36]. Thus, to ensure $t_j\sigma \gtrsim s_{j+1}\sigma$ for all j, we only have to require $l \gtrsim r$ for all rules $l \to r$ of $\mathcal{R}'_i$ provided that the quasi-order also satisfies $c(x, y) \gtrsim x$ and $c(x, y) \gtrsim y$. For automation, this is not a restriction since one usually uses *quasi-simplification orders* $\gtrsim$ (i.e., monotonic and stable quasi-orders with the *subterm property* $f(\ldots t \ldots) \gtrsim t$ for any term t and symbol f) or polynomial orders. Any quasi-simplification order orients C_ε and similarly, every polynomial order can be extended to orient C_ε.

Theorem 14 (Improved Termination Proofs with DPs) *Let* $\mathcal{R}_1, ..., \mathcal{R}_n$ *be a* separation *of* $\mathcal{R}$. *The TRS* $\mathcal{R}$ *is terminating if for all* $1 \leq i \leq n$ *and any cycle* $\mathcal{P}$ *of the dependency graph of* $\mathcal{R}$ *with* $\mathcal{P} \subseteq DP(\mathcal{R}_i)$, *there is a reduction pair* $(\gtrsim, \succ)$ *and an argument filtering* π *such that both*

(a) $\pi(s) \succ \pi(t)$ *for one dependency pair* $s \to t$ *from* $\mathcal{P}$ *and*

[2] A chain $s_1 \to t_1, s_2 \to t_2, \ldots$ is *minimal* if there is a substitution σ with $t_j\sigma \to^*_{\mathcal{R}} s_{j+1}\sigma$ and all $s_j\sigma$ are terminating. Minimal infinite chains correspond to infinite reductions of minimal non-terminating terms (i.e., their proper subterms are terminating). So absence of infinite *minimal* chains already suffices for termination.

$\pi(s) \succsim \pi(t)$ *or* $\pi(s) \succ \pi(t)$ *for all other pairs* $s \to t$ *from* $\mathcal{P}$

(b) $\pi(l) \succsim \pi(r)$ *for all* $l \to r \in \mathcal{R}'_i \cup C_\varepsilon$

Proof. If $\mathcal{R}$ is not terminating, then by Thm. 4 there exists an infinite chain $s_1 \to t_1, s_2 \to t_2, \ldots$ where $t_j\sigma \to^*_{\mathcal{R}} s_{j+1}\sigma$ for all j. As can be seen from the proof of Thm. 4 in [2, Thm. 6], the infinite chain and the substitution can be chosen such that all $s_j\sigma$ and $t_j\sigma$ are terminating, i.e., there exists a minimal infinite chain.

Without loss of generality, the dependency pairs $s_j \to t_j$ come from a cycle $\mathcal{P}$ of $\mathcal{R}$'s dependency graph where $\mathcal{P} \subseteq DP(\mathcal{R}_i)$. Let $\mathcal{R}''_i = \mathcal{R} \setminus \mathcal{R}'_i$. Then $\mathcal{R}'_i$ and $\mathcal{R}''_i$ form a hierarchical combination (thus, defined symbols of $\mathcal{R}'_i$ may occur as constructors in $\mathcal{R}''_i$, but not vice versa). By [36, Lemma 2] there is a substitution σ' such that $t_j\sigma' \to^*_{\mathcal{R}'_i \cup C_\varepsilon} s_{j+1}\sigma'$. Since $\pi(l) \succsim \pi(r)$ for all $l \to r \in \mathcal{R}'_i \cup C_\varepsilon$ by constraint (b), we obtain $\pi(t_j\sigma') = \pi(t_j)\,\sigma'_\pi \succsim \pi(s_{j+1})\,\sigma'_\pi = \pi(s_{j+1}\sigma')$ where $\sigma'_\pi(x) = \pi(\sigma'(x))$ for all $x \in \mathcal{V}$. Constraint (a) implies $\pi(s_j)\,\sigma'_\pi \succ \pi(t_j)\,\sigma'_\pi$ for infinitely many j and $\pi(s_j)\,\sigma'_\pi \succsim \pi(t_j)\,\sigma'_\pi$ for all remaining j. By compatibility of $\succsim$ and $\succ$, this contradicts the well-foundedness of $\succ$. □

Example 15 This TRS of [32] shows that Thm. 14 not only increases efficiency, but also leads to a more powerful method. Here, $\mathsf{int}(\mathsf{s}^n(\mathsf{0}), \mathsf{s}^m(\mathsf{0}))$ computes $[\mathsf{s}^n(\mathsf{0}), \mathsf{s}^{n+1}(\mathsf{0}), \ldots, \mathsf{s}^m(\mathsf{0})]$, nil is the empty list, and cons represents list insertion.

$$\mathsf{intlist}(\mathsf{nil}) \to \mathsf{nil} \tag{14}$$
$$\mathsf{intlist}(\mathsf{cons}(x, y)) \to \mathsf{cons}(\mathsf{s}(x), \mathsf{intlist}(y)) \tag{15}$$
$$\mathsf{int}(\mathsf{0}, \mathsf{0}) \to \mathsf{cons}(\mathsf{0}, \mathsf{nil}) \tag{16}$$
$$\mathsf{int}(\mathsf{0}, \mathsf{s}(y)) \to \mathsf{cons}(\mathsf{0}, \mathsf{int}(\mathsf{s}(\mathsf{0}), \mathsf{s}(y))) \tag{17}$$
$$\mathsf{int}(\mathsf{s}(x), \mathsf{0}) \to \mathsf{nil} \tag{18}$$
$$\mathsf{int}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{intlist}(\mathsf{int}(x, y)) \tag{19}$$

The TRS is separated into the $\mathsf{intlist}$-rules $\mathcal{R}_1$ and the int-rules $\mathcal{R}_2 >_d \mathcal{R}_1$. We first show that the constraints of Thm. 12 for termination of the cycle $\mathcal{P} = \{\mathsf{INTLIST}(\mathsf{cons}(x, y)) \to \mathsf{INTLIST}(y)\}$ cannot be solved with reduction pairs based on simplification orders. Thus, an automated termination proof with Thm. 12 is virtually impossible.

We must satisfy $\pi(\mathsf{INTLIST}(\mathsf{cons}(x, y))) \succ \pi(\mathsf{INTLIST}(y))$ $(*)$. We distinguish three cases depending on the filtering π. First, let $\pi(\mathsf{s}) \neq [\,]$ or $\pi(\mathsf{int}) = [\,]$. Then we have $\pi(\mathsf{int}(\mathsf{0}, \mathsf{s}(y))) \succsim \pi(\mathsf{cons}(\mathsf{0}, \mathsf{int}(\mathsf{s}(\mathsf{0}), \mathsf{s}(y)))) \succsim \pi(\mathsf{cons}(\mathsf{0}, \mathsf{int}(\mathsf{0}, \mathsf{s}(y))))$ by weak decreasingness of rule (17) and the subterm property. When substituting x by $\mathsf{0}$ and y by $\mathsf{int}(\mathsf{0}, \mathsf{s}(y))$ in $(*)$, we obtain a contradiction to well-foundedness of $\succ$.

Next, let $\pi(\mathsf{intlist}) = [\,]$. We have $\mathsf{intlist} \succsim \pi(\mathsf{cons}(\mathsf{s}(x), \mathsf{intlist}(\ldots)))$ by rule (15), which gives a similar contradiction when substituting x by $\mathsf{s}(x)$ and y by $\mathsf{intlist}(...)$ in $(*)$.

Finally, let $\pi(\mathsf{s}) = [\,]$, $\pi(\mathsf{int}) \neq [\,]$, and $\pi(\mathsf{intlist}) \neq [\,]$. Now we obtain a contradiction, since the filtered rule (19) cannot be weakly decreasing. The reason is that x or y occur on its right-hand side, but not on its left-hand side.

In contrast, when using Thm. 14, only the $\mathsf{intlist}$-rules $\mathcal{R}'_1 = \mathcal{R}_1$ must be weakly decreasing when examining $\mathcal{P}$. These constraints are satisfied by the embedding order using an argument filtering with $\pi(\mathsf{cons}) = [2]$ and $\pi(\mathsf{intlist}) = \pi(\mathsf{INTLIST}) = 1$.

The constraints from $\mathcal{R}_2$'s cycle and rules from $\mathcal{R}_2' = \mathcal{R}_1 \cup \mathcal{R}_2$ can also be oriented (by LPO and a filtering with $\pi(\mathsf{cons}) = 1$ and $\pi(\mathsf{INT}) = 2$). However, this part of the proof requires the consideration of cycles of the dependency graph. The reason is that there is no argument filtering and simplification order such that both dependency pairs of $\mathcal{R}_2$ are strictly decreasing: $\pi(\mathsf{INT}(\mathsf{s}(x), \mathsf{s}(y))) \succ \pi(\mathsf{INT}(x, y))$ implies $\pi(\mathsf{s}) = [1]$. But then $\pi(\mathsf{INT}(\mathsf{0}, \mathsf{s}(y)))$ is embedded in $\pi(\mathsf{INT}(\mathsf{s}(\mathsf{0}), \mathsf{s}(y)))$. Hence, we have $\pi(\mathsf{INT}(\mathsf{s}(\mathsf{0}), \mathsf{s}(y))) \not\succ \pi(\mathsf{INT}(\mathsf{0}, \mathsf{s}(y)))$ for any simplification order $\succ$.

In contrast, with Thm. 14, only the $\mathsf{intlist}$-rules $\mathcal{R}_1' = \mathcal{R}_1$ must be weakly decreasing for $\mathcal{P}$. These constraints are satisfied by the embedding order using an argument filtering with $\pi(\mathsf{cons}) = [2]$ and $\pi(\mathsf{intlist}) = \pi(\mathsf{INTLIST}) = 1$.

The constraints from $\mathcal{R}_2$'s cycle and rules from $\mathcal{R}_2' = \mathcal{R}_1 \cup \mathcal{R}_2$ can also be oriented (by LPO and $\pi(\mathsf{cons}) = 1$, $\pi(\mathsf{INT}) = 2$). However, for this part of the proof one has to consider cycles of the dependency graph. The reason is that there is no argument filtering and simplification order such that both dependency pairs of $\mathcal{R}_2$ are strictly decreasing.

So if one only considers cycles or if one only uses Urbain's modularity result [36], then the termination proof of Ex. 15 fails with simplification orders. Instead, both refinements should be combined as in Thm. 14. As observed in [30], if one uses the EDG-estimation from Def. 8 in combination with quasi-simplification orders, one can only prove C_ε-termination. However, since Thm. 14 permits arbitrary estimation techniques for dependency graphs such as EDG*, in contrast to the criteria in [36] it can also prove termination of TRSs like $\{\mathsf{f}(\mathsf{0}, \mathsf{1}, x) \to \mathsf{f}(x, x, x)\}$ that are not C_ε-terminating.

4 Improved Innermost Termination Proofs

Proving innermost termination with dependency pairs is easier than proving termination for two reasons: the innermost dependency graph has less arcs than the dependency graph and we only require $\pi(l) \succsim \pi(r)$ for the *usable* rules $l \to r$ instead of all rules. In Sect. 3 we showed that for termination, it suffices to require $\pi(l) \succsim \pi(r)$ only for the rules of $\mathcal{R}_i'$ if the current cycle consists of $\mathcal{R}_i$-dependency pairs. Still, $\mathcal{R}_i'$ is always a superset of the usable rules. Now we introduce an improvement of Thm. 12 for innermost termination in order to reduce the set of usable rules.

The idea is to apply the argument filtering first and to determine the usable rules afterwards. The advantage is that after the argument filtering, some symbols g may have been eliminated from the right-hand sides of dependency pairs and thus, the g-rules do not have to be included in the usable rules anymore. Moreover, if f's rules are usable and f calls a function g, then up to now g's rules are also considered usable. However, if all calls of g are only on positions that are eliminated by the argument filtering, then g's rules are not considered usable anymore. Thus, we obtain less constraints of the form $\pi(l) \succsim \pi(r)$.

However, for collapsing argument filterings this refinement is not sound. Consider the non-innermost 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 $\{\mathsf{F}(\mathsf{s}(x)) \to \mathsf{F}(\mathsf{double}(x))\}$, we can use the filtering $\pi(\mathsf{double}) = 1$ which results in $\{\mathsf{F}(\mathsf{s}(x)) \to \mathsf{F}(x)\}$. Since the filtered dependency pair contains no defined symbols, we would conclude that the cycle has no usable rules. Then we could easily orient the only resulting constraint $\mathsf{F}(\mathsf{s}(x)) \succ \mathsf{F}(x)$ for this cycle and falsely prove innermost termination. Note that the elimination of double in $\mathsf{F}(\mathsf{double}(x))$ is not due to the outer symbol F, but due to a collapsing argument filtering for double itself. For that reason a defined symbol like double may only be ignored when constructing the usable rules, if all its occurrences are in positions which are filtered away by the function symbols *above* them.

Definition 16 (Usable Rules w.r.t. Argument Filtering) *For an argument filtering π and an n-ary symbol f, the set $rp_\pi(f)$ of* regarded positions *is $\{i\}$, if $\pi(f) = i$, and it is $\{i_1, \ldots, i_m\}$, if $\pi(f) = [i_1, \ldots, i_m]$. Again, let $\mathcal{R}' = \mathcal{R} \setminus Rls_\mathcal{R}(f)$. For any term, we define*

- $\mathcal{U}_\mathcal{R}(x, \pi) = \varnothing$ *for $x \in \mathcal{V}$ and*
- $\mathcal{U}_\mathcal{R}(f(t_1, \ldots, t_n), \pi) = Rls_\mathcal{R}(f) \,\cup\, \bigcup_{l \to r \in Rls_\mathcal{R}(f)} \mathcal{U}_{\mathcal{R}'}(r, \pi) \,\cup\, \bigcup_{j \in rp_\pi(f)} \mathcal{U}_{\mathcal{R}'}(t_j, \pi)$.

For a term s, let $\mathcal{U}^s_\mathcal{R}(t, \pi) = \varnothing$ if t is a subterm of s and otherwise

$$\mathcal{U}^s_\mathcal{R}(f(t_1, .., t_n), \pi) = Rls_\mathcal{R}(f) \cup \bigcup_{l \to r \in Rls_\mathcal{R}(f)} \mathcal{U}_{\mathcal{R}'}(r, \pi) \cup \bigcup_{j \in rp_\pi(f)} \mathcal{U}^s_{\mathcal{R}'}(t_j, \pi).$$

For any set $\mathcal{P}$ of dependency pairs, let $\mathcal{U}_\mathcal{R}(\mathcal{P}, \pi) = \bigcup_{s \to t \in \mathcal{P}} \mathcal{U}^s_\mathcal{R}(t, \pi)$.

Example 17 We illustrate the new definition of usable rules with the following TRS of [22] for list reversal.

$$\mathsf{rev}(\mathsf{nil}) \to \mathsf{nil} \tag{20}$$
$$\mathsf{rev}(\mathsf{cons}(x, l)) \to \mathsf{cons}(\mathsf{rev1}(x, l), \mathsf{rev2}(x, l)) \tag{21}$$
$$\mathsf{rev1}(x, \mathsf{nil}) \to x \tag{22}$$
$$\mathsf{rev1}(x, \mathsf{cons}(y, l)) \to \mathsf{rev1}(y, l) \tag{23}$$
$$\mathsf{rev2}(x, \mathsf{nil}) \to \mathsf{nil} \tag{24}$$
$$\mathsf{rev2}(x, \mathsf{cons}(y, l)) \to \mathsf{rev}(\mathsf{cons}(x, \mathsf{rev}(\mathsf{rev2}(y, l)))) \tag{25}$$

Note that for any cycle containing the dependency pair $\mathsf{REV2}(x, \mathsf{cons}(y, l)) \to \mathsf{REV}(\mathsf{cons}(x, \mathsf{rev}(\mathsf{rev2}(y, l))))$, up to now all rules would have been usable, since rev and $\mathsf{rev2}$ occur in the right-hand side and the function rev calls rev_1. However, if one uses an argument filtering π with $\pi(\mathsf{cons}) = [2]$, then with our new definition of usable rules from Def. 16, the rev_1-rules would no longer be usable. The reason is that while rev and $\mathsf{rev2}$ still occur in the right-hand side of the filtered dependency pair, $\mathsf{rev1}$ no longer occurs in right-hand sides of filtered rev- or $\mathsf{rev2}$-rules. We will show in Ex. 20 that this reduction of the set of usable rules is crucial for the success of the innermost termination proof.

To prove the soundness of our refinement for innermost termination proofs, we need the following lemma. For a reduction pair $(\succsim, \succ)$, the pair $(\succsim_\pi, \succ_\pi)$ results from applying an argument filtering, where $t \succsim_\pi u$ holds iff $\pi(t) \succsim \pi(u)$ and $t \succ_\pi u$ holds iff $\pi(t) \succ \pi(u)$. In [2] it was shown that $(\succsim_\pi, \succ_\pi)$ is indeed a reduction pair as well.

Lemma 18 (Properties of Usable Rules) *Let $\mathcal{R}$ be a TRS, let π be an argument filtering, let $\succsim$ be a quasi-rewrite order, let σ be a normal substitution (i.e., $\sigma(x)$ is in normal form for all $x \in \mathcal{V}$), and let s be a term such that $s\sigma$ is in normal form. For each term t, "$Or(t)$" abbreviates the property that $\pi(l) \succsim \pi(r)$ holds for all $l \to r \in \mathcal{U}^s_{\mathcal{R}}(t,\pi)$. For all terms t, v we have*

(i) $\mathcal{U}^s_{\mathcal{R}}(t,\pi) \subseteq \mathcal{U}_{\mathcal{R}}(t,\pi)$

(ii) *If $Or(t)$ and $t\sigma \xrightarrow{\text{i}}_{\mathcal{R}} v$, then $t\sigma \succsim_\pi v$ and there is a term u and a normal substitution σ' with $u\sigma' = v$, $s\sigma' = s\sigma$, $t\sigma' = t\sigma$, and $Or(u)$.*

(iii) *If $Or(t)$ and $t\sigma \xrightarrow{\text{i}}^*_{\mathcal{R}} v$ then $t\sigma \succsim_\pi v$.*

Proof. We have $\mathcal{U}^s_{\mathcal{R}}(f(t_1,..,t_n),\pi = Rls_{\mathcal{R}}(f) \cup \bigcup_{l\to r \in Rls_{\mathcal{R}}(f)} \mathcal{U}_{\mathcal{R}}(r,\pi) \cup \bigcup_{j \in rp_\pi(f)} \mathcal{U}^s_{\mathcal{R}}(t_j,\pi)$, where we replaced $\mathcal{R}'$ by $\mathcal{R}$. In the rest of the proof omit the subscript $\mathcal{R}$ to increase readability.

(i) We use induction on t. If t is a subterm of s, then we have $\mathcal{U}^s(t,\pi) = \varnothing$. Otherwise, $t = f(t_1,\ldots,t_n)$ and

$$\begin{aligned}\mathcal{U}^s(t,\pi) &= Rls(f) \cup \textstyle\bigcup_{l\to r\in Rls(f)} \mathcal{U}(r,\pi) \cup \bigcup_{j\in rp_\pi(f)} \mathcal{U}^s(t_j,\pi)\\ &\overset{(ind.)}{\subseteq} Rls(f) \cup \textstyle\bigcup_{l\to r\in Rls(f)} \mathcal{U}(r,\pi) \cup \bigcup_{j\in rp_\pi(f)} \mathcal{U}(t_j,\pi) = \mathcal{U}(t,\pi)\end{aligned}$$

(ii) Since $s\sigma$ is in normal form, t is not a subterm of s. We perform induction on the position of the reduction. This position must be in t because σ is normal. So, t has the form $f(t_1,\ldots,t_n)$.
If $t\sigma = l\tau \xrightarrow{\text{i}}_{\mathcal{R}} r\tau = v$, then τ is a normal substitution due to the innermost strategy. We may assume that the rule is variable disjoint from s and t and define $\sigma'(x) = \tau(x)$ for $x \in \mathcal{V}(r)$ and $\sigma'(x) = \sigma(x)$ for $x \in \mathcal{V}(s) \cup \mathcal{V}(t)$. Let $u = r$. We have $\mathcal{U}^s(t,\pi) \supseteq Rls(f) \cup \bigcup_{l'\to r' \in Rls(f)} \mathcal{U}(r',\pi) \supseteq \{l \to r\} \cup \mathcal{U}^s(r,\pi)$ by (i). So $Or(t)$ implies $Or(u)$ and $\pi(l) \succsim \pi(r)$ which also shows $\pi(t\sigma) \succsim \pi(v)$.
Otherwise, $t\sigma = f(t_1\sigma \ldots t_j\sigma \ldots t_n\sigma) \to_{\mathcal{R}} f(t_1\sigma \ldots v_j \ldots t_n\sigma) = v$. If $j \notin rp_\pi(f)$, then $\pi(t\sigma) = \pi(v)$. Let v'_j result from v_j by replacing its variables x by corresponding fresh variables x'. We define $\sigma'(x') = x$ for all these fresh variables and $\sigma'(x) = \sigma(x)$ for all $x \in \mathcal{V}(s) \cup \mathcal{V}(t)$. Then let $u = f(t_1 \ldots v'_j \ldots t_n)$. Now $u\sigma' = v$, and $\mathcal{U}^s(t,\pi) = \mathcal{U}^s(u,\pi)$ implies $Or(u)$. For $j \in rp_\pi(f)$ we have $\mathcal{U}^s(t,\pi) \supseteq \mathcal{U}^s(t_j,\pi)$ which implies $Or(t_j)$. By the induction hypothesis we have $\pi(t_j\sigma) \succsim \pi(v_j)$ and monotonicity of $\succsim_\pi$ implies $\pi(t\sigma) \succsim \pi(v)$. By the induction hypothesis there is also some u_j and σ' with $u_j\sigma' = v_j$ and $Or(u_j)$. Let $u = f(t_1 \ldots u_j \ldots t_n)$. So $u\sigma' = v$ and $\mathcal{U}^s(u,\pi) \subseteq \mathcal{U}^s(t,\pi) \cup \mathcal{U}^s(u_j,\pi)$. Thus, $Or(t)$ and $Or(u_j)$ imply $Or(u)$.

(iii) We use induction on the length n of the reduction. For $n = 0$ the claim is trivial. Otherwise, $t\sigma \xrightarrow{\text{i}}_{\mathcal{R}} v' \xrightarrow{\text{i}}^{n-1}_{\mathcal{R}} v$. By (ii), we have $\pi(t\sigma) \succsim \pi(v')$ and there exists a term u and a normal substitution σ' with $u\sigma' = v'$ and $s\sigma' = s\sigma$ such that $Or(u)$. Since $s\sigma'$ is a normal form, the induction hypothesis implies $\pi(u\sigma) \succsim \pi(v)$ and the claim follows from transitivity of $\succsim_\pi$. □

Now we can refine the innermost termination technique of Thm. 12 to the following one where the set of usable rules is reduced significantly.

Theorem 19 (Improved Innermost Termination with DPs) *A TRS $\mathcal{R}$ is innermost terminating if for any cycle $\mathcal{P}$ of the innermost dependency graph, there is a reduction pair $(\succsim, \succ)$ and an argument filtering π such that both*

(c) $\pi(s) \succ \pi(t)$ *for one dependency pair* $s \to t$ *from* $\mathcal{P}$ *and* $\pi(s) \succsim \pi(t)$ *or* $\pi(s) \succ \pi(t)$ *for all other pairs* $s \to t$ *from* $\mathcal{P}$

(d) $\pi(l) \succsim \pi(r)$ *for all* $l \to r \in \mathcal{U}_\mathcal{R}(\mathcal{P}, \pi)$

Proof. By Thm. 4, we have to show absence of infinite innermost chains. Let $s_1 \to t_1, s_2 \to t_2, \ldots$ be an infinite innermost chain from the cycle $\mathcal{P}$. So there is a normal substitution σ with $t_j\sigma \xrightarrow{\text{i}}{}^*_\mathcal{R} s_{j+1}\sigma$ where $s_j\sigma$ are in normal form for all j. Hence, we obtain $t_j\sigma \succsim_\pi s_{j+1}\sigma$ by (d) and Lemma 18 (iii). By (c) and closure of $\succ_\pi$ under substitutions, we have $s_1\sigma \succsim_\pi t_1\sigma \succsim_\pi s_2\sigma \succsim_\pi \ldots$ where $s_j\sigma \succ_\pi t_j\sigma$ holds for infinitely many j in contradiction to the well-foundedness of $\succ_\pi$. □

Example 20 The TRS for list reversal from Ex. 17 [22] shows the advantages of Thm. 19. When proving innermost termination with Thm. 12, for the cycle of the REV- and $\mathsf{REV2}$-dependency pairs, we obtain inequalities from the dependency pairs and $\pi(l) \succsim \pi(r)$ for all rules $l \to r$, since all rules are usable. But with standard reduction pairs based on RPOS, KBO, or polynomial orders, these constraints are not satisfiable for any argument filtering.

For RPOS and KBO, we first show that if an argument position is eliminated by an argument filtering π, then the constraints cannot be satisfied. From (22) we obtain $1 \in rp_\pi(\mathsf{rev1})$ which leads to $2 \in rp_\pi(\mathsf{rev1})$ and $1, 2 \in rp_\pi(\mathsf{cons})$ by using (23) twice, so $\pi(\mathsf{rev1}) = \pi(\mathsf{cons}) = [1, 2]$. Using (21) we obtain $1 \in rp_\pi(\mathsf{rev})$. Now we can conclude $\pi(\mathsf{rev2}) = [1, 2]$ from (25). If we have $\pi(\mathsf{rev}) = 1$, then (21) yields a contradiction to the subterm property. Hence, $\pi(\mathsf{rev}) = [1]$. Thus, if we search for a simplification order such that the rules are weakly decreasing, then we are not allowed to drop any argument or function symbol in the filtering. Hence, it is sufficient to examine whether the orders above are able to make the unfiltered rules weakly decreasing.

There is no KBO satisfying these constraints since (21) is duplicating. If we want to orient the constraints by some lexicographic or recursive path order, we need a precedence with $\mathsf{rev2} > \mathsf{rev}$ due to (25). But this precedence cannot be extended further to orient (21).

There is also no polynomial order satisfying the constraints of Thm. 12. A polynomial interpretation has the following form.

$$\begin{array}{lll}
\mathcal{P}ol(\mathsf{rev}(l)) & = p_1(l), & \text{where } \; p_1(l) = p'_1 \cdot l^{n_1} + p''_1(l) \\
\mathcal{P}ol(\mathsf{rev1}(x, l)) & = p_2(x, l), & \text{where } p_2(x, l) = p'_2(x) \cdot l^{n_2} + p''_2(x, l) \\
\mathcal{P}ol(\mathsf{rev2}(x, l)) & = p_3(x, l), & \text{where } p_3(x, l) = p'_3(x) \cdot l^{n_3} + p''_3(x, l) \\
\mathcal{P}ol(\mathsf{cons}(x, l)) & = p_4(x, l), & \text{where } p_4(x, l) = p'_4(x) \cdot l^{n_4} + p''_4(x, l) \\
\mathcal{P}ol(\mathsf{nil}) & = p_5 &
\end{array}$$

Here, n_1, n_2, n_3, n_4 denote the highest exponents used for l in the respective polynomials, where p'_i and p''_i are polynomials with coefficients from $\mathbb{N}$. So in $p''_1(l)$, $p''_2(x, l)$, $p''_3(x, l)$, and $p''_4(x, l)$, the variable l occurs only with exponents smaller than the corresponding n_i. Similar to the argumentation above where we showed that with simplification orders one may not filter away any arguments, it is easy to show that $\mathcal{P}ol(\mathsf{rev1}(x, l))$, $\mathcal{P}ol(\mathsf{rev2}(x, l))$, and $\mathcal{P}ol(\mathsf{cons}(x, l))$ must depend on x and l and $\mathcal{P}ol(\mathsf{rev}(l))$ must depend on l. Hence, all values n_i must be at least 1 and the polynomials p'_i are not the number 0.

From the constraints of (21) and (25) we obtain

$$\mathcal{P}ol(\mathsf{rev}(\mathsf{cons}(x, l))) \geq \mathcal{P}ol(\mathsf{cons}(\mathsf{rev1}(x, l), \mathsf{rev2}(x, l)))$$

$$\mathcal{P}ol(\mathsf{rev2}(x, \mathsf{cons}(x, l))) \geq \mathcal{P}ol(\mathsf{rev}(\mathsf{cons}(x, \mathsf{rev}(\mathsf{rev2}(x, l))))).$$

We now examine those parts of the polynomials which have the largest exponent for l. So for large enough instantiations of l (and instantiations of x where the p'_i are non-zero) we must have

$$p'_1 \cdot p'_4(x)^{n_1} \cdot l^{n_1 \cdot n_4} \geq p'_4(p'_2(x) \cdot l^{n_2}) \cdot p'_3(x)^{n_4} \cdot l^{n_3 \cdot n_4} \tag{26}$$

$$p'_3(x) \cdot p'_4(x)^{n_3} \cdot l^{n_3 \cdot n_4} \geq p'_4(x)^{n_1} \cdot {p'_1}^{n_1 \cdot n_4 + 1} \cdot p'_3(x)^{n_1^2 \cdot n_4} \cdot l^{n_1^2 \cdot n_3 \cdot n_4} \tag{27}$$

Comparison of the highest exponents of l yields $n_1 \cdot n_4 \geq n_3 \cdot n_4 \geq n_1^2 \cdot n_3 \cdot n_4$ and thus, $n_1 = n_3 = 1$. Moreover, $p'_4(x)$ may not depend on x, since otherwise (26) would imply $n_1 \cdot n_4 \geq n_3 \cdot n_4 + n_2$. Now (26) and (27) simplify to

$$p'_1 \geq p'_3(x)^{n_4} \tag{28}$$

$$p'_3(x) \geq {p'_1}^{n_4+1} \cdot p'_3(x)^{n_4} \tag{29}$$

From (28) and (29) we can conclude that $p'_3(x)$ does not depend on x and $p'_3 = p'_1 = 1$. Hence, our polynomial interpretation is as follows:

$$\begin{array}{ll}
\mathcal{P}ol(\mathsf{rev}(l)) & = l + p''_1 \\
\mathcal{P}ol(\mathsf{rev1}(x, l)) & = p'_2(x) \cdot l^{n_2} + p''_2(x, l) \\
\mathcal{P}ol(\mathsf{rev2}(x, l)) & = l + p''_3(x) \\
\mathcal{P}ol(\mathsf{cons}(x, l)) & = p'_4 \cdot l^{n_4} + p''_4(x, l) \\
\mathcal{P}ol(\mathsf{nil}) & = p_5
\end{array}$$

Now we obtain

$$p'_4 \cdot p_5^{n_4} + p''_4(x, p_5) + p''_1 = \tag{30}$$

$$\mathcal{P}ol(\mathsf{rev}(\mathsf{cons}(x, \mathsf{nil}))) \geq \tag{31}$$

$$\mathcal{P}ol(\mathsf{cons}(\mathsf{rev1}(x, \mathsf{nil}), \mathsf{rev2}(x, \mathsf{nil}))) \geq \tag{32}$$

$$\mathcal{P}ol(\mathsf{cons}(x, \mathsf{rev2}(x, \mathsf{nil}))) = \tag{33}$$

$$p'_4 \cdot (p_5 + p''_3(x))^{n_4} + p''_4(x, p_5 + p''_3(x)) \geq \tag{34}$$

$$p'_4 \cdot p_5^{n_4} + p''_4(x, p_5) + p'_4 \cdot p''_3(x)^{n_4} \tag{35}$$

The step from (31) to (32) is due to the weak decreasingness of rule (21) and the step from (32) to (33) follows from monotonicity and rule (22). Note that these inequalities give a contradiction if one instantiates x with a large enough value like $p''_1 + 1$, since $\mathcal{P}ol(\mathsf{rev2}(x, l))$ and hence $p''_3(x)$ must depend on x.

So the most common orders that are amenable to automation fail when trying to prove innermost termination according to Thm. 12. In contrast, with Thm. 19 and $\pi(\mathsf{cons}) = [2]$, $\pi(\mathsf{REV}) = \pi(\mathsf{rev}) = 1$, and $\pi(\mathsf{REV1}) = \pi(\mathsf{REV2}) = \pi(\mathsf{rev2}) = 2$, we obtain no constraints from the rev1-rules, cf. Ex. 17. Then all filtered constraints can be oriented by the embedding order.

Our experiments in Sect. 9 show that Thm. 14 and 19 indeed improve upon Thm. 12 by increasing power (in particular if reduction pairs are based on simple fast orders like the embedding order) and by reducing runtimes (in particular if one uses more complex orders).

5 Transforming Dependency Pairs

To increase the power of the dependency pair technique, a dependency pair may be transformed into new pairs by *narrowing*, *rewriting*, and *instantiation* [2, 13]. A term t' is an *$\mathcal{R}$-narrowing of t with mgu μ* if a subterm $t|_p \notin \mathcal{V}$ of t unifies with the left-hand side of a (variable-renamed) rule $l \to r \in \mathcal{R}$ with mgu μ, and $t' = t[r]_p\,\mu$. To distinguish the variants of narrowing and instantiation for ***t***ermination and ***i***nnermost termination, we speak of *t-* and *i-narrowing* resp. *-instantiation.*

Definition 21 (Transformations) *For a set $\mathcal{P}$ of pairs of terms*

- $\mathcal{P} \uplus \{s \to t\}$ t-narrows *to* $\mathcal{P} \uplus \{s\mu_1 \to t_1, \ldots, s\mu_n \to t_n\}$ *iff* $t_1, \ldots, t_n$ *are all $\mathcal{R}$-narrowings of t with the mgu's $\mu_1, \ldots, \mu_n$ and t does not unify with (variable-renamed) left-hand sides of pairs in $\mathcal{P}$. Moreover, t must be linear.*
- $\mathcal{P} \uplus \{s \to t\}$ i-narrows *to* $\mathcal{P} \uplus \{s\mu_1 \to t_1, \ldots, s\mu_n \to t_n\}$ *iff* $t_1, \ldots, t_n$ *are all $\mathcal{R}$-narrowings of t with the mgu's $\mu_1, \ldots, \mu_n$ such that $s\mu_i$ is in normal form. Moreover, for all $v \to w \in \mathcal{P}$ where t unifies with the (variable-renamed) left-hand side v by a mgu μ, one of the terms $s\mu$ or $v\mu$ must not be in normal form.*
- $\mathcal{P} \uplus \{s \to t\}$ rewrites *to* $\mathcal{P} \uplus \{s \to t'\}$ *iff* $\mathcal{U}^s_{\mathcal{R}}(t|_p)$ *is non-overlapping and* $t \to_{\mathcal{R}} t'$*, where p is the position of the redex.*
- $\mathcal{P} \uplus \{s \to t\}$ *is* t-instantiated *to*
  $\mathcal{P} \uplus \{s\mu \to t\mu \mid \mu = mgu(\textsc{ren}(\textsc{cap}(w)), s),\ v \to w \in \mathcal{P}\}$ *or to*
  $\mathcal{P} \uplus \{s\mu \to t\mu \mid \mu = mgu(\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{R}}(v)), t),\ v \to w \in \mathcal{P}\}$
- $\mathcal{P} \uplus \{s \to t\}$ *is* i-instantiated *to*
  $\mathcal{P} \uplus \{s\mu \to t\mu \mid \mu = mgu(\textsc{cap}^v(w), s), v \to w \in \mathcal{P}, s\mu, v\mu \text{ normal}\}$ *or to*
  $\mathcal{P} \uplus \{s\mu \to t\mu \mid \mu = mgu(\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v)), t), v \to w \in \mathcal{P}, s\mu \text{ normal}\}$

In termination proofs, one may modify dependency pairs by t-narrowing and t-instantiation and for innermost termination, one may apply i-narrowing, i-instantiation, and rewriting of dependency pairs.

In contrast to the instantiation technique in [13], Def. 21 also builds instantiations by regarding pairs that may *follow* $s \to t$ in a chain instead of only regarding pairs that may *precede* $s \to t$. To determine instantiations from pairs that precede $s \to t$, one uses estimations as in E(I)DG. To obtain instantiations from pairs that follow $s \to t$, we use concepts as in EDG* and EIDG**.

Example 22 The instantiation technique of [13] cannot transform the dependency pairs of the TRS $\mathsf{f}(x, y, z) \to \mathsf{g}(x, y, z)$, $\mathsf{g}(\mathsf{0}, \mathsf{1}, x) \to \mathsf{f}(x, x, x)$, since in chains $v \to w$, $s \to t$, the mgu of $\textsc{ren}(\textsc{cap}(w))$ and s does not modify s. But in the chain $\mathsf{F}(x, y, z) \to \mathsf{G}(x, y, z)$, $\mathsf{G}(\mathsf{0}, \mathsf{1}, x') \to \mathsf{F}(x', x', x')$, the mgu of $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{R}}(\mathsf{G}(\mathsf{0}, \mathsf{1}, x')))$ and $\mathsf{G}(x, y, z)$ modifies the first pair. So with the improved technique of Def. 21, $\mathsf{F}(x, y, z) \to \mathsf{G}(x, y, z)$ is instantiated to $\mathsf{F}(\mathsf{0}, \mathsf{1}, z) \to \mathsf{G}(\mathsf{0}, \mathsf{1}, z)$. Now the termination proof succeeds since EDG* has no cycle, while it would fail without instantiations for any reduction pair based on simplification orders.

For innermost termination, Def. 21 extends the transformations of [2, 13] by permitting their application for a larger set of TRSs. In [13], narrowing a pair $s \to t$ was not permitted if t unifies with the left-hand side of some dependency pair, whereas now this is possible under certain conditions. Rewriting dependency pairs was only allowed if all usable rules for the current cycle were non-overlapping, whereas now this is only required for the usable rules of the redex to be rewritten. Finally, when instantiating dependency pairs, in contrast to [13] one can now use $\textsc{cap}^v$. Moreover, for both instantiation and narrowing of dependency pairs, now one only has to consider instantiations which turn left-hand sides of dependency pairs into normal forms.

Before proving the soundness of these transformations, we first illustrate their application with an example which also shows that these transformations are often crucial for the success of the proof.

Example 23 The following alternative TRS for division is from [3].

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

Without transformations, no simplification order satisfies Thm. 19's constraints for innermost termination of the following cycle.

$$\mathsf{QUOT}(x, \mathsf{s}(y)) \to \mathsf{IF}(\mathsf{le}(\mathsf{s}(y), x), x, \mathsf{s}(y)) \tag{36}$$

$$\mathsf{IF}(\mathsf{true}, x, y) \to \mathsf{QUOT}(\mathsf{minus}(x, y), y) \tag{37}$$

The reason is that the dependency pair constraints of this cycle imply

$$\begin{aligned} \pi(\mathsf{IF}(\mathsf{true}, x, \mathsf{s}(y))) &\succsim \\ \pi(\mathsf{QUOT}(\mathsf{minus}(x, \mathsf{s}(y)), \mathsf{s}(y))) &\succsim \\ \pi(\mathsf{IF}(\mathsf{le}(\mathsf{s}(y), \mathsf{minus}(x, \mathsf{s}(y))), \mathsf{minus}(x, \mathsf{s}(y)), \mathsf{s}(y))) & \end{aligned}$$

where one of the constraints has to be strict. Hence, we have

$$\pi(\mathsf{IF}(\mathsf{true}, x, \mathsf{s}(y))) \succ \pi(\mathsf{IF}(\mathsf{le}(\mathsf{s}(y), \mathsf{minus}(x, \mathsf{s}(y))), \mathsf{minus}(x, \mathsf{s}(y)), \mathsf{s}(y)))$$

From the minus-rules we see that an argument filtering π must not drop the first argument of minus. Hence, by the subterm property we get $\pi(\mathsf{minus}(x, y)) \succsim \pi(x)$. This leads to

$$\pi(\mathsf{IF}(\mathsf{true}, x, \mathsf{s}(y))) \succ \pi(\mathsf{IF}(\mathsf{le}(\mathsf{s}(y), x), x, \mathsf{s}(y))). \tag{38}$$

In order to obtain a contradiction we first show the following property.

$$\pi(\mathsf{le}(\mathsf{s}(\mathsf{true}), \mathsf{s}(\mathsf{true}))) \succsim \pi(\mathsf{true}) \tag{39}$$

If $\pi(\mathsf{le}) = [\,]$, then using the first le-rule we can directly conclude that (39) holds. Otherwise, by the last le-rule we get $\pi(\mathsf{le}(\mathsf{s}(\mathsf{true}), \mathsf{s}(\mathsf{true}))) \succsim \pi(\mathsf{le}(\mathsf{true}, \mathsf{true}))$ and $\pi(\mathsf{le}(\mathsf{true}, \mathsf{true})) \succsim \pi(\mathsf{true})$ by the subterm property.

Now, using (38), (39), and the substitution $\{x/\mathsf{s}(\mathsf{true}), y/\mathsf{true}\}$ we obtain the desired contradiction.

$$\begin{array}{r} \pi(\mathsf{IF}(\mathsf{true}, \mathsf{s}(\mathsf{true}), \mathsf{s}(\mathsf{true}))) \succ \\ \pi(\mathsf{IF}(\mathsf{le}(\mathsf{s}(\mathsf{true}), \mathsf{s}(\mathsf{true})), \mathsf{s}(\mathsf{true}), \mathsf{s}(\mathsf{true}))) \succsim \\ \pi(\mathsf{IF}(\mathsf{true}, \mathsf{s}(\mathsf{true}), \mathsf{s}(\mathsf{true}))). \end{array}$$

On the other hand, when transforming the dependency pairs, the resulting constraints can easily be satisfied. Intuitively, $x \succ \mathsf{minus}(x, y)$ only has to be required if $\mathsf{le}(\mathsf{s}(y), x)$ reduces to true. This argumentation can be simulated using the transformations of Def. 21. By i-narrowing, we perform a case analysis on how the le-term in (36) can be evaluated. In the first narrowing, x is instantiated by $\mathsf{0}$. This results in a pair $\mathsf{QUOT}(\mathsf{0}, \mathsf{s}(y)) \to \mathsf{IF}(\mathsf{false}, \mathsf{0}, \mathsf{s}(y))$ which is not in a cycle. The other narrowing is

$$\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{IF}(\mathsf{le}(y, x), \mathsf{s}(x), \mathsf{s}(y)) \tag{40}$$

which forms a cycle with (37). Now we perform i-instantiation of (37) and see that x and y must be of the form $\mathsf{s}(\ldots)$. So (37) is replaced by

$$\mathsf{IF}(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{QUOT}(\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{s}(y)) \tag{41}$$

that forms a cycle with (40). Finally, by rewriting (41) we obtain

$$\mathsf{IF}(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{QUOT}(\mathsf{minus}(x, y), \mathsf{s}(y)) \tag{42}$$

The constraints of the resulting cycle $\{(40), (42)\}$ (and all other cycles) are solved by $\pi(\mathsf{minus}) = \pi(\mathsf{QUOT}) = 1$, $\pi(\mathsf{IF}) = 2$, and the embedding order.

Lemma 24 is needed to prove that if an innermost termination proof is possible without transformations, then it still works after transformations. In part (iv), we show that if a term t can be reduced to u, then the usable rules of t are a superset of u's usable rules. Thus, any quasi-order which orients the usable rules of t also orients the usable rules of u. However, this only holds if after the argument filtering, all variables on right-hand sides of usable rules still occur on the corresponding left-hand side. Otherwise, the TRS $\mathcal{R}$ with the rule $\mathsf{f}(x) \to x$ and additional rules for a symbol a would be a counterexample if one uses an argument filtering with $\pi(\mathsf{f}) = [\,]$. Now $\mathsf{f}(\mathsf{a})$ would reduce to a, but we would have $\mathcal{U}_\mathcal{R}(\mathsf{f}(\mathsf{a}), \pi) \not\supseteq \mathcal{U}_\mathcal{R}(\mathsf{a}, \pi)$, since $\mathcal{U}_\mathcal{R}(\mathsf{f}(\mathsf{a}), \pi)$ does not contain the a-rules. So after rewriting or narrowing, one could obtain additional constraints which require that the a-rules are weakly decreasing. This might destroy the success of the innermost termination proof.

Lemma 24 (More Properties of Usable Rules) *Let π be an argument filtering, let $\succsim$ be a quasi-rewrite order, let σ be a substitution, and let s and t be terms, where s is in normal form. Then we have*

(i) *If $\mathcal{V}(t) \subseteq \mathcal{V}(s)$, then $\mathcal{U}^{s\sigma}_\mathcal{R}(t\sigma, \pi) \subseteq \mathcal{U}^{s}_\mathcal{R}(t, \pi)$.*

(ii) *A position p of a term t is called* regarded *iff $p = \varepsilon$ or $p = jp'$, $t = f(t_1, \ldots, t_n)$, $j \in rp_\pi(f)$, and p' is regarded in t_j.*
If u is a subterm of t at a regarded position,

then we have $\mathcal{U}^s_{\mathcal{R}}(u,\pi) \subseteq \mathcal{U}^s_{\mathcal{R}}(t,\pi)$.

(iii) *If* $\mathcal{V}(\pi(u)) \subseteq \mathcal{V}(\pi(t))$, *then* $\mathcal{U}^s_{\mathcal{R}}(u\sigma,\pi) \subseteq \mathcal{U}^s_{\mathcal{R}}(t\sigma,\pi) \cup \mathcal{U}_{\mathcal{R}}(u,\pi)$.

(iv) *If* $\mathcal{V}(\pi(r)) \subseteq \mathcal{V}(\pi(l))$ *for all* $l \to r \in \mathcal{U}^s_{\mathcal{R}}(t,\pi)$,
then $t \to_{\mathcal{R}} u$ *implies* $\mathcal{U}^s_{\mathcal{R}}(t,\pi) \supseteq \mathcal{U}^s_{\mathcal{R}}(u,\pi)$.

(v) *If* $l \succsim_\pi r$ *for all* $l \to r \in \mathcal{U}^s_{\mathcal{R}}(t,\pi)$ *and* $t \to_{\mathcal{R}} u$, *then* $t \succsim_\pi u$.

Proof. Again we omit the subscript $\mathcal{R}$ and use $\mathcal{U}^s_{\mathcal{R}}(f(t_1,..,t_n),\pi) = Rls_{\mathcal{R}}(f) \cup \bigcup_{l\to r\in Rls_{\mathcal{R}}(f)} \mathcal{U}_{\mathcal{R}}(r,\pi) \cup \bigcup_{j\in rp_\pi(f)} \mathcal{U}^s_{\mathcal{R}}(t_j,\pi)$.

(i) As in Lemma 18 (i), we use induction on t. If $t \in \mathcal{V}$, then t is a subterm of s and thus, $\mathcal{U}^{s\sigma}(t\sigma,\pi) = \mathcal{U}^s(t,\pi) = \varnothing$. Otherwise, $t = f(t_1,\ldots,t_n)$. Then

$$\begin{aligned}\mathcal{U}^{s\sigma}(t\sigma,\pi) &= Rls(f) \cup \textstyle\bigcup_{l\to r\in Rls(f)} \mathcal{U}(r,\pi) \cup \bigcup_{j\in rp_\pi(f)} \mathcal{U}^{s\sigma}(t_j\sigma,\pi)\\ &\overset{(ind.)}{\subseteq} Rls(f) \cup \textstyle\bigcup_{l\to r\in Rls(f)} \mathcal{U}(r,\pi) \cup \bigcup_{j\in rp_\pi(f)} \mathcal{U}^{s}(t_j,\pi) = \mathcal{U}^s(t,\pi)\end{aligned}$$

(ii) We use induction on t. If $t = u$ then the claim is trivial. If t is a subterm of s then so is u and $\mathcal{U}^s(t,\pi) = \mathcal{U}^s(u,\pi) = \varnothing$. Otherwise, t has the form $f(t_1,\ldots,t_n)$ and u is a subterm of some t_j for $j \in rp_\pi(f)$. So the claim follows from the induction hypothesis.

(iii) We perform induction on u. If u is a variable, then $u\sigma$ is a subterm of $t\sigma$ at a regarded position and thus, the claim follows from (ii). Otherwise, we have $u = f(u_1,\ldots,u_n)$. The interesting case is if $u\sigma$ is not a subterm of s. Then $\mathcal{U}^s(u\sigma,\pi) = Rls(f) \cup \bigcup_{l\to r\in Rls(f)} \mathcal{U}(r,\pi) \cup \bigcup_{j\in rp_\pi(f)} \mathcal{U}^s(u_j\sigma,\pi)$. We have $Rls(f) \cup \bigcup_{l\to r\in Rls(f)} \mathcal{U}(r,\pi) \subseteq \mathcal{U}(u,\pi)$ and for all $j \in rp_\pi(f)$, the induction hypothesis implies $\mathcal{U}^s(u_j\sigma,\pi) \subseteq \mathcal{U}^s(t\sigma,\pi) \cup \mathcal{U}(u_j,\pi) \subseteq \mathcal{U}^s(t\sigma,\pi) \cup \mathcal{U}(u,\pi)$.

(iv) Let $t \to_{\mathcal{R}} u$ using the rule $l \to r \in \mathcal{R}$. As t can be reduced, it is no normal form and thus, no subterm of s. The only interesting case is if u is no subterm of s either. We perform structural induction on the position p of the redex. If $p = \varepsilon$ then $t = l\sigma \to_{\mathcal{R}} r\sigma = u$ for a substitution σ. As $l \to r \in \mathcal{U}^s(t,\pi)$, we have $\mathcal{V}(\pi(r)) \subseteq \mathcal{V}(\pi(l))$. Since $\mathcal{U}(r,\pi) \subseteq \mathcal{U}^s(l\sigma,\pi)$, now the claim follows from (iii).
Otherwise $p = jp'$, $t = f(t_1 \ldots t_j \ldots t_n)$, $u = f(t_1 \ldots u_j \ldots t_n)$, and $t_j \to_{\mathcal{R}} u_j$. If $j \notin rp_\pi(f)$, then $\mathcal{U}^s(t,\pi) = \mathcal{U}^s(u,\pi)$. If $j \in rp_\pi(f)$, then

$$\begin{aligned}\mathcal{U}^s(t,\pi) &= Rls(f) \cup \textstyle\bigcup_{l\to r\in Rls(f)} \mathcal{U}(r,\pi) \cup \ldots \cup \mathcal{U}^s(t_j,\pi) \cup \ldots\\ &\overset{(ind.)}{\supseteq} Rls(f) \cup \textstyle\bigcup_{l\to r\in Rls(f)} \mathcal{U}(r,\pi) \cup \ldots \cup \mathcal{U}^s(u_j,\pi) \cup \ldots\\ &= \mathcal{U}^s(u,\pi).\end{aligned}$$

(v) The proof is an easy induction on the position of the redex. □

The following theorem states the soundness of the dependency pair transformations in part (a). So in (innermost) termination proofs with Thm. 14 and 19, instead of the original dependency pairs one may regard pairs that are transformed according to Def. 21. Of course, then Thm. 14 and 19 have to be updated accordingly (e.g., in Thm. 14, instead of $\mathcal{P} \subseteq DP(\mathcal{R}_i)$ we now permit that $\mathcal{P}$ results from pairs of $DP(\mathcal{R}_i)$ by transformations). While soundness of the earlier versions of these transformations was already shown in [2, 13], part (b) of the following theorem is a new completeness result that is important in practice: if (innermost) termination can be proved without transformations, then the same proof still works after performing transformations. In other words, applying transformations can never harm.

Theorem 25 (Sound- and Completeness) *Let $DP(\mathcal{R})'$ result from $DP(\mathcal{R})$ by t-narrowing and t-instantiation (for termination) resp. by i-narrowing, rewriting, and i-instantiation (for innermost termination).*

(a) If the dependency pair constraints for (innermost) termination are satisfiable using $DP(\mathcal{R})'$, then $\mathcal{R}$ is (innermost) terminating.

(b) If certain reduction pairs and argument filterings satisfy the constraints for $DP(\mathcal{R})$, then the same *reduction pairs and filterings satisfy the constraints for $DP(\mathcal{R})'$. Here, one has to use the estimation* EDG *or* EDG* *for dependency graphs. For innermost termination, one must use* EIDG *or* EIDG***, and reduction pairs $(\succsim, \succ)$ and filterings π where $t \succsim_\pi u$ implies $\mathcal{V}(\pi(u)) \subseteq \mathcal{V}(\pi(t))$.*

Proof. For *soundness* (part (a)), one proves that if there is an infinite (innermost) chain of pairs from $DP(\mathcal{R})$, then there is also an infinite (innermost) chain of pairs from $DP(\mathcal{R})'$.[3] Hence, if the constraints using $DP(\mathcal{R})'$ are satisfiable, then the correctness of the dependency pair approach implies (innermost) termination. Soundness is proved in [2, Thm. 27] for t-narrowing, in [13, Thm. 12] for i-narrowing, and in [13, Thm. 20] for instantiation. These proofs can easily be adapted to the new refined versions of i-narrowing and instantiation in Def. 21, due to the soundness of EDG* and EIDG** and since in innermost chains one only regards substitutions which instantiate all left-hand sides of dependency pairs to normal forms.

For soundness of the refined version of rewriting, we adapt the proof of [13, Thm. 18]. We assume that $DP(\mathcal{R})'$ is the result of rewriting a dependency pair $s \to t$ from $DP(\mathcal{R})$ to $s \to t'$ (i.e., t rewrites to t' at some position p). Let $\ldots, s \to t, v \to w, \ldots$ be an innermost chain of pairs from $DP(\mathcal{R})$. Hence, there exists a substitution σ with $t\sigma \xrightarrow{\text{i}}{}^*_{\mathcal{R}} v\sigma$ and $s\sigma, v\sigma$ are normal forms. Thus, $t\sigma$ is weakly innermost terminating. Due to the innermost reduction strategy, we can split up the reduction of $t\sigma$ into two parts. First, we reduce only on positions on or below p until $t|_p\sigma$ is a normal form u. Afterwards we perform the remaining reduction steps from $t\sigma[u]_p$ to $v\sigma$. The only rules applicable to $t|_p\sigma$ are $\mathcal{U}^{s\sigma}_{\mathcal{R}}(t|_p\sigma)$. We have $\mathcal{U}^{s\sigma}_{\mathcal{R}}(t|_p\sigma) \subseteq \mathcal{U}^{s}_{\mathcal{R}}(t|_p)$ by Lemma 24 (i) and thus, $\mathcal{U}^{s\sigma}_{\mathcal{R}}(t|_p\sigma)$ is non-overlapping. Hence, by [18, Thm. 3.2.11 (1a) and (4a)], $t|_p\sigma$ is confluent and terminating. With $t|_p \to_{\mathcal{R}} t'|_p$ we obtain $t|_p\sigma \to_{\mathcal{R}} t'|_p\sigma$. Hence, $t'|_p\sigma$ is terminating as well and thus, it also reduces innermost to the same normal form u using the confluence of $t|_p\sigma$. So we have $t'\sigma = t\sigma[t'|_p\sigma]_p \xrightarrow{\text{i}}{}^*_{\mathcal{R}} t\sigma[u]_p$. Afterwards, we can apply the same remaining steps as above that lead from $t\sigma[u]_p$ to $v\sigma$. Therefore $\ldots, s \to t', v \to w, \ldots$ is an innermost chain as well.

Next we show *completeness* (part (b)) for t- and i-narrowing. The completeness proofs for rewriting and instantiation are analogous. We assume that $DP(\mathcal{R})'$ is the result of narrowing a dependency pair $s \to t$ from $DP(\mathcal{R})$. In this transformation, $s \to t$ was replaced by narrowings of the form $s\mu \to t'$ where $t\mu \to_{\mathcal{R}} t'$. Let p be the position of the redex, i.e., $t\mu|_p = l\sigma$ and $t' = t\mu[r\sigma]_p$ for some rule $l \to r \in \mathcal{R}$.

We first show that if $\mathcal{Q}$ is a cycle of the estimated dependency graph of $DP(\mathcal{R})'$ containing $s\mu \to t'$, then there is a corresponding cycle $\mathcal{P}$ in $DP(\mathcal{R})$'s

[3] The converse direction (i.e., if there is an infinite (innermost) chain of pairs from $DP(\mathcal{R})'$ then there is also an infinite (innermost) chain of pairs from $DP(\mathcal{R})$) holds as well for rewriting, instantiation, and t-narrowing. For i-narrowing, this direction only holds if the usable rules are non-overlapping (cf. [2, Ex. 43] and [13, Thm. 17]).

estimated dependency graph which results from $\mathcal{Q}$ by adding the pair $s \to t$ and deleting its narrowings. Assume there is an arc from $v \to w$ to $s\mu \to t'$ in the estimated dependency graph of $DP(\mathcal{R})'$. Thus, $\textsc{ren}(\textsc{cap}(w))$ and $s\mu$ are unifiable and if one uses EDG* instead of EDG, then $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(s\mu))$ and w are also unifiable. Obviously, $s\mu$ is an instance of s and moreover, $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(s\mu))$ is an instance of $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(s))$ for any term s. The latter is shown by structural induction on s: if $s \in \mathcal{V}$, if $\mathcal{R}$ is collapsing, or if $\mathrm{root}(s)$ is the root of some rule's right-hand side, then $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(s)) \in \mathcal{V}$. Otherwise, the claim follows from the induction hypothesis, since $s = f(s_1, \ldots, s_n)$, $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(s)) = f(\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(s_1)), \ldots, \textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(s_n)))$, and all $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(s_1))$ are variable disjoint. Hence, if $\textsc{ren}(\textsc{cap}(w))$ is unifiable with $s\mu$, then it is also unifiable with s and if $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(s\mu))$ unifies with w, then so does $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(s))$. So there is also an arc from $v \to w$ to $s \to t$ in the estimated dependency graph of $DP(\mathcal{R})$.

Similarly, if there is an arc from $s\mu \to t'$ to $v \to w$, then there is also an arc from $s \to t$ to $v \to w$ in $DP(\mathcal{R})$'s estimated dependency graph. To see this, recall that $\textsc{ren}(\textsc{cap}(t'))$ and v must be unifiable and if one uses EDG*, then $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v))$ and t' are also unifiable. Unifiability of $\textsc{ren}(\textsc{cap}(t'))$ and v implies unifiability of $\textsc{ren}(\textsc{cap}(t))$ and v, since $\textsc{ren}(\textsc{cap}(t'))$ is an instance of $\textsc{ren}(\textsc{cap}(t\mu))$ which in turn is an instance of $\textsc{ren}(\textsc{cap}(t))$. It remains to show that if $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v))$ and t' unify by a substitution τ, then $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v))$ and t are unifiable by a substitution τ' such that $t\mu\tau = t\tau'$ for any terms t and t' with $t\mu \to_{\mathcal{R}} t'$. We perform induction on the position p of the redex in the reduction from $t\mu$ to t'. In the induction base, we have $p = \varepsilon$ and $t' = r\sigma$. Unifiability of $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v))$ and t' implies that $v \in \mathcal{V}$, $\mathcal{R}$ is collapsing, or $\mathrm{root}(v)$ is the root of some rule's right-hand side. In all these cases, $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v))$ is a variable and thus, the claim holds. Otherwise, $p = jp'$, $t = f(t_1, \ldots, t_n)$, and $t' = f(t_1\mu, \ldots, t_j\mu[r\sigma]_{p'}, \ldots, t_n\mu)$. If $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v)) \notin \mathcal{V}$, then f is not the root of any rule's right-hand side and $v = f(v_1, \ldots, v_n)$. Hence, $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v)) = f(\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v_1)), \ldots, \textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v_n)))$ and the claim follows from the induction hypothesis and the fact that all $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v_i))$ are variable disjoint. Here we also need that the unifier of $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v_j))$ and $t_j\mu[r\sigma]_{p'}$ instantiates $t\mu$'s variables in the same way as all unifiers of $\textsc{ren}(\textsc{cap}_{\mathcal{R}}^{-1}(v_i))$ and $t_i\mu$ for $i \neq j$.

Now we show that if a reduction pair $(\succsim, \succ)$ and an argument filtering π satisfy all constraints for the cycle $\mathcal{P}$ then they also satisfy the constraints for $\mathcal{Q}$. These constraints only differ in that $s \succsim_\pi t$ resp. $s \succ_\pi t$ is replaced by $s\mu \succsim_\pi t'$ resp. $s\mu \succ_\pi t'$. The constraints of Type (b) are the same. Note that if $\mathcal{P}$ and $\mathcal{Q}$ contain a pair $F(\ldots) \to \ldots$, then for every function symbol $g \in \mathcal{F}$ occurring in pairs of $\mathcal{P}$ or $\mathcal{Q}$, we have $g \leq_d f$. Then $s \succsim_\pi t$ implies $s\mu \succsim_\pi t\mu \succsim_\pi t'$ by stability of $\succsim_\pi$ and by the fact that $t\mu$ rewrites to t' using a g-rule for a function symbol $g \leq_d f$. Hence, the constraints of Type (b) imply that all g-rules are weakly decreasing. Similarly, $s \succ_\pi t$ implies $s\mu \succ_\pi t\mu \succsim_\pi t'$.

Finally we prove completeness of i-narrowing. As for t-narrowing, we first show that if $\mathcal{Q}$ is a cycle of the estimated innermost dependency graph of $DP(\mathcal{R})'$ containing $s\mu \to t'$, then the set $\mathcal{P}$ resulting from adding $s \to t$ and removing its narrowings is a cycle in the estimated innermost dependency graph of $DP(\mathcal{R})$. As in the termination case, one can show that arcs from $v \to w$ to $s\mu \to t'$ correspond to arcs from $v \to w$ to $s \to t$ in the estimated innermost dependency graph of $DP(\mathcal{R})$ if one uses the estimations EIDG or EIDG**: Assume there is an arc from $v \to w$ to $s\mu \to t'$ in the estimated innermost dependency graph of $DP(\mathcal{R})'$. Thus, $\textsc{cap}^v(w)$ and $s\mu$ are unifiable by some mgu τ_1 and if one uses EIDG** instead of EIDG, then $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^v_\mathcal{R}(w)}(s\mu))$ and w are also unifiable by a mgu τ_2. In addition, $v\tau_i$ and $s\mu\tau_1$ are normal forms. Again, $s\mu$ is an instance of s and thus, the substitution τ_1' which is like τ_1 on the variables of v and $\textsc{cap}^v(w)$ and like $\mu\tau_1$ on the variables of s unifies $\textsc{cap}^v(w)$ and s where $v\tau_1'$ and $s\tau_1'$ are normal forms. Since τ_1' is an instance of the mgu of $\textsc{cap}^v(w)$ and s, this mgu instantiates v and s to normal forms, too.

Moreover, by structural induction on s one can show that $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^v_\mathcal{R}(w)}(s\mu))$ is an instance of $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^v_\mathcal{R}(w)}(s))$ for any term s. If $s \in \mathcal{V}$, if $\mathcal{U}^v_\mathcal{R}(w)$ is collapsing, or if $\mathrm{root}(s)$ is the root of some right-hand side of a rule from $\mathcal{U}^v_\mathcal{R}(w)$, then $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^v_\mathcal{R}(w)}(s)) \in \mathcal{V}$. Otherwise, the claim follows from the induction hypothesis, since we have $s = f(s_1, \ldots, s_n)$, $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^v_\mathcal{R}(w)}(s)) = f(\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^v_\mathcal{R}(w)}(s_1)), \ldots, \textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^v_\mathcal{R}(w)}(s_n)))$, and all $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^v_\mathcal{R}(w)}(s_i))$ are variable disjoint. Hence, unifiability of $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^v_\mathcal{R}(w)}(s\mu))$ and w implies unifiability $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^v_\mathcal{R}(w)}(s))$ and w and both mgu's are the same on the variables of v (thus, v is still instantiated to a normal form). Therefore, there is also an arc from $v \to w$ to $s \to t$ in the estimated innermost dependency graph of $DP(\mathcal{R})$.

Similarly, if there is an arc from $s\mu \to t'$ to $v \to w$, then there is also an arc from $s \to t$ to $v \to w$ in the estimated innermost dependency graph of $DP(\mathcal{R})$. To see this, recall that $\textsc{cap}^{s\mu}(t')$ and v must be unifiable by some mgu τ_1 and if one uses EIDG**, then $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^{s\mu}_\mathcal{R}(t')}(v))$ and t' are unifiable by a mgu τ_2. Moreover, $s\mu\tau_1$, $v\tau_1$, and $s\mu\tau_2$ are normal forms. Unifiability of $\textsc{cap}^{s\mu}(t')$ and v implies unifiability of $\textsc{cap}^s(t)$ and v, since $\textsc{cap}^{s\mu}(t')$ is an instance of $\textsc{cap}^{s\mu}(t\mu)$ which in turn is an instance of $\textsc{cap}^s(t)$. To see this, recall that $t\mu$ rewrites to t'. Thus, the subterm of $t\mu$ that is the redex in this reduction cannot occur in $s\mu$, since $s\mu$ must be a normal form. Hence, in $\textsc{cap}^{s\mu}(t\mu)$, this subterm (or a subterm containing this redex) is replaced by a fresh variable and thus, $\textsc{cap}^{s\mu}(t')$ is an instance of $\textsc{cap}^{s\mu}(t\mu)$. If a subterm of t occurs also in s, then the corresponding subterm of $t\mu$ also occurs in $s\mu$. In contrast, there may be subterms of $t\mu$ that occur in $s\mu$, whereas no corresponding subterm of t occurs in s. This indicates that $\textsc{cap}^{s\mu}(t\mu)$ is an instance of $\textsc{cap}^s(t)$. So $\textsc{cap}^s(t)$ and v are unifiable by some mgu τ_1', where $v\tau_1' = v\tau_1$ and $s\mu\tau_1$ is an instance of $s\tau_1'$. Thus, $s\tau_1'$ and $v\tau_1'$ are also in normal form.

Now we show that $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^s_\mathcal{R}(t)}(v))$ and t are unifiable by a substitution τ_2' such that $t\mu\tau_2 = t\tau_2'$. We first regard the case where $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^{s\mu}_\mathcal{R}(t')}(v)) \in \mathcal{V}$. Then we have $v \in \mathcal{V}$, $\mathcal{U}^{s\mu}_\mathcal{R}(t')$ is collapsing, or $\mathrm{root}(v)$ is the root of some right-hand side of a rule from $\mathcal{U}^{s\mu}_\mathcal{R}(t')$. Since $\mathcal{U}^{s\mu}_\mathcal{R}(t') \subseteq \mathcal{U}^{s\mu}_\mathcal{R}(t\mu) \subseteq \mathcal{U}^s_\mathcal{R}(t)$ by Lemma 24 (iv) and (i), in all these cases we also have $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^s_\mathcal{R}(t)}(v)) \in \mathcal{V}$. Here, the prerequisite $\mathcal{V}(r) \subseteq \mathcal{V}(l)$ of Lemma 24 (iv) is fulfilled, since we do not use any argument filtering. To handle the other cases, we again perform induction on the position p of the redex in the reduction from $t\mu$ to t'. In the induction base, we have $p = \varepsilon$ and $t' = r\sigma$ for a rule $l \to r \in \mathcal{U}^s_\mathcal{R}(t)$. If $\textsc{ren}(\textsc{cap}^{-1}_{\mathcal{U}^{s\mu}_\mathcal{R}(t')}(v)) \notin \mathcal{V}$ and if this term is unifiable with t', then we must have $r \in \mathcal{V}$ or $\mathrm{root}(v) = \mathrm{root}(r)$. So in both cases, we obtain $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v)) \in \mathcal{V}$. Otherwise, $p = jp'$, $t = f(t_1, \ldots, t_n)$, and $t' = f(t_1\mu, \ldots, t_j\mu[r\sigma]_{p'}, \ldots, t_n\mu)$. If $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v)) \notin \mathcal{V}$, then f is not the root of any right-hand side of a rule from $\mathcal{U}^s_{\mathcal{R}}(t)$ and $v = f(v_1, \ldots, v_n)$. Hence, $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v)) = f(\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v_1)), \ldots, \text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v_n)))$ and the claim follows from the induction hypothesis and from the fact that $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^{s\mu}_{\mathcal{R}}(t')}(v_i))$ is an instance of $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v_i))$ and all $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v_i))$ are variable disjoint. As in the termination case, we need that the unifier of $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v_j))$ and $t_j\mu[r\sigma]_{p'}$ instantiates $t\mu$'s variables in the same way as all unifiers of $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v_i))$ and $t_i\mu$ for $i \neq j$. Since τ_2' is an instance of the mgu of $\text{REN}(\text{CAP}^{-1}_{\mathcal{U}^s_{\mathcal{R}}(t)}(v))$ and t, their mgu instantiates s to a normal form, since $s\tau_2' = s\mu\tau_2$ is already a normal form.

Next we show that if a reduction pair $(\succsim, \succ)$ and an argument filtering π satisfy all constraints for the cycle $\mathcal{P}$, then they also satisfy the constraints for $\mathcal{Q}$. One difference between these constraints is that $s \succsim_\pi t$ resp. $s \succ_\pi t$ is replaced by $s\mu \succsim_\pi t'$ resp. $s\mu \succ_\pi t'$. Note that $s \succsim_\pi t$ again implies $s\mu \succsim_\pi t\mu \succsim_\pi t'$ by stability of $\succsim_\pi$ and by the fact that $t\mu$ rewrites to t'. Here, $t\mu \succsim t'$ follows by Lemma 24 (v), since all rules in $\mathcal{U}^{s\mu}_{\mathcal{R}}(t\mu, \pi) \subseteq \mathcal{U}^s_{\mathcal{R}}(t, \pi)$ are weakly decreasing w.r.t. $\succsim_\pi$ (cf. Lemma 24 (i)). Similarly, $s \succ_\pi t$ implies $s\mu \succ_\pi t'$.

The other difference is in the set of usable rules. But we have $\mathcal{U}^{s\mu}_{\mathcal{R}}(t', \pi) \subseteq \mathcal{U}^{s\mu}_{\mathcal{R}}(t\mu, \pi) \subseteq \mathcal{U}^s_{\mathcal{R}}(t, \pi)$ by Lemma 24 (iv) and (i). The application of Lemma 24 (iv) is possible here since we have $l \succsim_\pi r$ for all $l \to r \in \mathcal{U}^s_{\mathcal{R}}(t, \pi)$ and thus, we may conclude the needed requirement $\mathcal{V}(\pi(r)) \subseteq \mathcal{V}(\pi(l))$ for Lemma 24 (iv) from the restrictions on the quasi-order $\succsim$ and π. Therefore, we obtain $\mathcal{U}_{\mathcal{R}}(\mathcal{Q}, \pi) \subseteq \mathcal{U}_{\mathcal{R}}(\mathcal{P}, \pi)$. $\square$

The following example shows that the completeness of the transformations for innermost termination in part (b) does not hold anymore if one uses the non-improved version of usable rules from Def. 5 or the estimation EIDG* that is based on these non-improved usable rules.

Example 26 We regard a modification of the TRS from Ex. 7.

$$
\begin{array}{rclrcl}
\mathsf{f}(\mathsf{a}, x) & \to & \mathsf{f}(\mathsf{g}(x), x) \qquad & \mathsf{h}(\mathsf{g}(x)) & \to & \mathsf{h}(\mathsf{a}) \\
\mathsf{g}(\mathsf{h}(x)) & \to & \mathsf{g}(x) \qquad & \mathsf{h}(\mathsf{h}(x)) & \to & x
\end{array}
$$

The EIDG has one cycle $\{\mathsf{F}(\mathsf{a}, x) \to \mathsf{F}(\mathsf{g}(x), x)\}$, the g-rule is usable, and the resulting constraints are satisfied by LPO using the argument filtering $\pi(\mathsf{g}) = [\,]$. However, if one performs i-narrowing, then the dependency pair is replaced by $\mathsf{F}(\mathsf{a}, \mathsf{h}(x)) \to \mathsf{F}(\mathsf{g}(x), \mathsf{h}(x))$. If one uses the non-improved usable rules of Def. 5, one obtains the constraints of Ex. 13 which are not satisfiable by RPOS, KBO, or polynomial orders. Thus, completeness of i-narrowing does not hold anymore for EIDG.

Completeness is also destroyed for EIDG*, even if one uses the improved usable rules of Def. 6 or 16 in the constraints (d) of Thm. 19. Before narrowing, the EIDG* has no cycle. So the empty set of constraints is satisfied by any argument filtering and reduction pair. After narrowing, we have the cycle $\{\mathsf{F}(\mathsf{a}, \mathsf{h}(x)) \to \mathsf{F}(\mathsf{g}(x), \mathsf{h}(x))\}$. Now we need a filtering and reduction pair as above to satisfy the constraints.

By Thm. 25, transforming dependency pairs never complicates (innermost) termination proofs, if one uses improved usable rules. The restriction[4] to quasi-orders $\succsim$ where $t \succsim_\pi u$ implies $\mathcal{V}(\pi(u)) \subseteq \mathcal{V}(\pi(t))$ is not severe. The reason is that this requirement is always satisfied if one uses quasi-orders like RPOS, KBO, or polynomial orders that are amenable to automation. In fact, it holds for any reduction pair $(\succsim, \succ)$ where $\succsim$ is a quasi-simplification order and where there exist terms s, t with $s \succ t$ and t is no ground term [14, Lemma 2.4].

However, transformations may increase the number of constraints by producing similar constraints that can be solved by the same argument filterings and reduction pairs). So sometimes the runtime is increased by these transformations. On the other hand, transformations are often necessary for the (innermost) termination proofs, as shown by Ex. 23.

In practice, the main problem is that these transformations may be applied infinitely many times. Therefore, we have developed restricted *safe* transformations which are guaranteed to terminate. Our experiments on the collections of examples from [3, 9, 32] show that whenever the proof succeeds using narrowing, rewriting, and instantiation, then applying these safe transformations is sufficient.

A narrowing or instantiation step is *safe* if it reduces the number of pairs in cycles of the estimated (innermost) dependency graph. For a set of pairs $\mathcal{P}$, let $\mathrm{SCC}(\mathcal{P})$ be the set of maximal cycles built from pairs of $\mathcal{P}$. Then the transformation is safe if $\Sigma_{\mathcal{S} \in \mathrm{SCC}(\mathcal{P})} |\mathcal{S}|$ decreases. So the instantiation in Ex. 22 was safe, since the EDG* had a cycle before instantiation, but no cycle afterwards. Moreover, a transformation step is also considered safe if by this step, all descendants of some original dependency pair disappear from cycles. For every pair $s \to t$, $o(s \to t)$ denotes the original dependency pair whose repeated transformation led to $s \to t$. Now a transformation is also safe if $\{o(s \to t) \mid s \to t \in \bigcup_{\mathcal{S} \in \mathrm{SCC}(\mathcal{P})} \mathcal{S}\}$ decreases. Finally, for each pair that was not narrowed or instantiated yet, one single narrowing and instantiation step which does not satisfy the above requirements is also considered safe. Hence, the narrowing and instantiation steps in Ex. 23 were safe as well.

As for termination, innermost termination proofs also benefit from considering the recursion hierarchy. So if $\mathcal{R}_1, \ldots, \mathcal{R}_n$ is a separation of the TRS $\mathcal{R}$ and $\mathcal{R}_i >_d \mathcal{R}_j$, then we show absence of innermost $\mathcal{R}$-chains built from $DP(\mathcal{R}_j)$ before dealing with $DP(\mathcal{R}_i)$. Now innermost rewriting a dependency pair $F(\ldots) \to \ldots$ is *safe* if it is performed with rules that do not depend on f (i.e., with g-rules where $g <_d f$). The reason is that innermost termination of g is already verified when proving innermost termination of f. So in Ex. 23, when proving innermost termination of the QUOT-cycle, we may assume innermost termination of minus. Thus, the rewrite step from (41) to (42) was safe.

Definition 27 (Safe Transformations) *$\mathcal{Q}$ results from a set of pairs $\mathcal{P}$ by transforming $s \to t \in \mathcal{P}$ as in Def. 21. The transformation is* safe *if*

(1) $s \to t$ was transformed by narrowing or instantiation and

- $\Sigma_{\mathcal{S} \in \mathrm{SCC}(\mathcal{P})} |\mathcal{S}| > \Sigma_{\mathcal{S} \in \mathrm{SCC}(\mathcal{Q})} |\mathcal{S}|$, *or*
- $\{o(s \to t) \mid s \to t \in \bigcup_{\mathcal{S} \in \mathrm{SCC}(\mathcal{P})} \mathcal{S}\} \supsetneq \{o(s \to t) \mid s \to t \in \bigcup_{\mathcal{S} \in \mathrm{SCC}(\mathcal{Q})} \mathcal{S}\}$

[4] The restriction on the reduction pair is unnecessary if one computes usable rules without taking the filtering into account (i.e., if one uses Def. 6 instead of Def. 16).

(2) $s \to t$ *was transformed by innermost rewriting with the rule* $l \to r$ *and* $\mathrm{root}(l) <_d f$ *where* $f^\sharp = \mathrm{root}(s)$
(3) $s \to t$ *was transformed by narrowing and all previous steps which transformed* $o(s \to t)$ *to* $s \to t$ *were not narrowing steps*
4 $s \to t$ *was transformed by instantiation and all previous steps which transformed* $o(s \to t)$ *to* $s \to t$ *were not instantiation steps*

The following theorem proves that the repeated application of safe transformations is indeed terminating.

Theorem 28 (Termination) *Let $\mathcal{R}$ have the separation $\mathcal{R}_1, \ldots, \mathcal{R}_n$ and $\mathcal{P} \subseteq DP(\mathcal{R}_i)$. If there are no infinite innermost $\mathcal{R}$-chains from $DP(\mathcal{R}_j)$ for all $\mathcal{R}_j <_d \mathcal{R}_i$, then any repeated application of safe transformations on $\mathcal{P}$ terminates.*

Proof. We define a measure on sets $\mathcal{P}$ consisting of four components:

(a) $|\{o(s \to t) \mid s \to t \in \bigcup_{\mathcal{S} \in \mathrm{SCC}(\mathcal{P})} \mathcal{S}\}|$
(b) $\Sigma_{\mathcal{S} \in \mathrm{SCC}(\mathcal{P})} |\mathcal{S}|$
(c) $|\mathcal{P}|$
(d) $\{t \mid s \to t \in \mathcal{P}\}$

These 4-tuples are compared lexicographically by the usual order on naturals for components (a) – (c). For (d), we use the multiset extension of the innermost rewrite relation of $\bigcup_{\mathcal{R}_j <_d \mathcal{R}_i} \mathcal{R}_j$. Thus, we obtain a well-founded relation $\succ$ where $\mathcal{P}_1 \succ \mathcal{P}_2$ iff $\mathcal{P}_1$'s measure is greater than the measure of $\mathcal{P}_2$. Due to (a), (b), and (d), any safe transformation of $\mathcal{P}$ with steps (1) or (2) decreases the measure of $\mathcal{P}$.

For a set of pairs $\mathcal{P}$, let $w(\mathcal{P}) = \langle \mathcal{P}_{\neg n, \neg i}, \mathcal{P}_{n, \neg i}, \mathcal{P}_{\neg n, i}, \mathcal{P}_{n,i} \rangle$. $\mathcal{P}_{\neg n, \neg i}$ consists of those $s \to t \in \mathcal{P}$ where no (n)arrowing or (i)nstantiation was used to transform $o(s \to t)$ to $s \to t$. $\mathcal{P}_{n, \neg i}$ are the pairs where narrowing, but no instantiation was used, etc. Every safe transformation step decreases $w(\mathcal{P})$ lexicographically w.r.t. $\succ$: The leftmost component of $w(\mathcal{P})$ that is changed decreases w.r.t. $\succ$, whereas components on its right-hand side may be increasing. In particular, transformations with (3) or (4) decrease one component of $w(\mathcal{P})$ w.r.t. $\succ$ according to (c). □

After each transformation, the current cycle or SCC of the estimated (innermost) dependency graph is re-computed. For the re-computation, one only has to regard the former neighbors of the transformed pair in the old graph. The reason is that only former neighbors may have arcs to or from the new pairs resulting from the transformation. Regarding neighbors in the graphs also suffices when performing the unifications required for narrowing and instantiation. In this way, the transformations can be performed efficiently. Recall that one should always regard SCCs first and afterwards, one builds new SCCs from the remaining pairs which were not strictly decreasing (Sect. 2) [20]. Of course, these pairs may already have been transformed during the (innermost) termination proof of the SCC. So this approach has the advantage that one never repeats transformations for the same dependency pairs.

6 Computing Argument Filterings

In the dependency pair approach we may apply an argument filtering π to a set of constraints before starting an orientation attempt with a reduction pair. When using reduction pairs based on monotonic orders $\succ$ like RPOS or KBO, this is necessary to benefit from the fact that the dependency pair approach permits reduction pairs $(\succsim, \succ)$ where $\succ$ is not monotonic. However, the number of possible argument filterings is exponential in the number and the arities of the function symbols. We now show how to search for suitable argument filterings efficiently. More precisely, for every cycle $\mathcal{P}$, we show how to compute small subsets $\Pi^t(\mathcal{P})$ and $\Pi^i(\mathcal{P})$ of argument filterings which contain all filterings which could possibly satisfy the constraints for ***t***ermination or ***i***nnermost termination, respectively. A corresponding algorithm was presented in [20] for termination proofs w.r.t. Thm. 12. However, we now develop such an algorithm for the improved versions of the dependency pair approach from Thm. 14 and Thm. 19. In particular for innermost termination (Thm. 19), the algorithm is considerably more involved since the set of constraints depends on the argument filtering used. Moreover, instead of treating constraints separately as in [20], we process them according to an efficient depth-first strategy.

Let $\mathcal{RP}$ be a class of reduction pairs describing the particular base order used (e.g., $\mathcal{RP}$ may contain all LPOs with arbitrary precedences or all RPOSs, etc.). In the termination case, we restrict ourselves to classes of pairs where the quasi-order orients the C_ε-rules. For any set of dependency pairs $\mathcal{P}$, let $\Pi(\mathcal{P})$ denote the set of all argument filterings where at least one dependency pair in $\mathcal{P}$ is strictly decreasing and the remaining ones are weakly decreasing w.r.t. some reduction pair in $\mathcal{RP}$. When referring to "dependency pairs", we also permit pairs resulting from dependency pairs by narrowing, rewriting, or instantiation.

We use the approach of [20] to consider partial argument filterings, which are only defined on a subset of the signature. For example, in a term $f(g(x), y)$, if $\pi(f) = [2]$, then we do not have to determine $\pi(g)$, since all occurrences of g are filtered away. Thus, we leave argument filterings as undefined as possible and permit the application of π to a term t whenever π is *sufficiently defined* for t. More precisely, any partial argument filtering π is sufficiently defined for a variable x. So the domain of π may even be empty, i.e., $DOM(\pi) = \varnothing$. An argument filtering π is sufficiently defined for $f(t_1, \ldots, t_n)$ iff $f \in DOM(\pi)$ and π is sufficiently defined for all t_i with $i \in rp_\pi(f)$. An argument filtering is sufficiently defined for a set of terms T iff it is sufficiently defined for all terms in T. To compare argument filterings which only differ in their domain DOM, we introduce a relation "$\sqsubseteq$". Then $\Pi(\mathcal{P})$ should only contain $\sqsubseteq$-minimal elements, i.e., if $\pi' \in \Pi(\mathcal{P})$, then $\Pi(\mathcal{P})$ does not contain any $\pi \sqsubset \pi'$. Of course, all argument filterings in $\Pi(\mathcal{P})$ must be sufficiently defined for the terms in the dependency pairs of $\mathcal{P}$.

Definition 29 ($\sqsubseteq$ and $\Pi(\mathcal{P})$) *For two (partial) argument filterings, we define $\pi \sqsubseteq \pi'$ iff $DOM(\pi) \subseteq DOM(\pi')$ and $\pi(f) = \pi'(f)$ for all $f \in DOM(\pi)$. For a set $\mathcal{P}$ of dependency pairs, let $\Pi(\mathcal{P})$ consist of all $\sqsubseteq$-minimal elements of $\{\pi \mid$ there is a $(\succsim, \succ) \in \mathcal{RP}$ such that $\pi(s) \succ \pi(t)$ for at least one $s \to t \in \mathcal{P}$ and $\pi(s) \succsim \pi(t)$ for all other $s \to t \in \mathcal{P}\}$.*

In Ex. 1, if $\mathcal{P} = \{\mathsf{QUOT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{QUOT}(\mathsf{minus}(x, y), \mathsf{s}(y))\}$ and $\mathcal{RP}$ are all LPOs, then $\Pi(\mathcal{P})$ consists of the 12 filterings π where $DOM(\pi) = \{\mathsf{QUOT}, \mathsf{s}, \mathsf{minus}\}$, $\pi(\mathsf{QUOT}) \in \{1, [1], [1, 2]\}$, and either $\pi(\mathsf{minus}) \in \{[\,], 1, [1]\}$ and $\pi(\mathsf{s}) = [1]$ or both $\pi(\mathsf{minus}) = \pi(\mathsf{s}) = [\,]$.

We now define a superset $\Pi^t(\mathcal{P})$ of all argument filterings where the constraints (a) and (b) for termination of the cycle $\mathcal{P}$ are satisfied by some reduction pair of $\mathcal{RP}$. So only these argument filterings have to be regarded when automating Thm. 14. To this end, we have to *extend* partial argument filterings.

Definition 30 (Ex_f, $\Pi^t(\mathcal{P})$) *For a partial argument filtering π and $f \in \mathcal{D}$, $Ex_f(\pi)$ consists of all $\sqsubseteq$-minimal filterings π' with $\pi \sqsubseteq \pi'$ such that there is a $(\succsim, \succ) \in \mathcal{RP}$ with $\pi'(l) \succsim \pi'(r)$ for all $l \to r \in Rls_{\mathcal{R}}(f)$. For a set Π of filterings, let $Ex_f(\Pi) = \bigcup_{\pi \in \Pi} Ex_f(\pi)$. If $\mathcal{P}$ originates from $DP(\mathcal{R}_i)$ by t-narrowing and t-instantiation and $\{f_1, \ldots, f_k\}$ are $\mathcal{R}'_i$'s defined symbols, then $\Pi^t(\mathcal{P}) = Ex_{f_k}(\ldots Ex_{f_1}(\Pi(\mathcal{P}))\ldots)$.*

We compute $\Pi^t(\mathcal{P})$ by depth-first search. We start with a $\pi \in \Pi(\mathcal{P})$ and extend it to a minimal π' such that f_1's rules are weakly decreasing. Then π' is extended such that f_2's rules are weakly decreasing, etc. Here, f_1 is considered before f_2 if $f_1 >_d f_2$. When we have $\Pi^t(\mathcal{P})$'s first element π_1, we check whether the constraints (a) and (b) of Thm. 14 are satisfiable with π_1. In case of success, we do not compute further elements of $\Pi^t(\mathcal{P})$. Only if the constraints are not satisfiable with π_1, we determine $\Pi^t(\mathcal{P})$'s next element, etc. The advantage of this approach is that $\Pi(\mathcal{P})$ is usually small, since it only contains filterings that satisfy a *strict* inequality. Thus, by taking $\Pi(\mathcal{P})$'s restrictions into account, only a fraction of the search space is examined.

So for the QUOT-cycle $\mathcal{P}$ in Ex. 1, we start with computing the first element of $Ex_{\mathsf{quot}}(\Pi(\mathcal{P}))$. To this end, the partial filterings $\pi \in \Pi(\mathcal{P})$ have to be extended to filterings that are also defined on quot and $\mathsf{0}$ where $\pi(\mathsf{quot}) \in \{[\,], 1, [1], [1,2]\}$ and if $\pi(\mathsf{s}) = [\,]$, then $\pi(\mathsf{quot})$ may also be 2 or [2]. So $|Ex_{\mathsf{quot}}(\Pi(\mathcal{P}))| = 54$. In $\Pi^t(\mathcal{P}) = Ex_{\mathsf{minus}}(Ex_{\mathsf{quot}}(\Pi(\mathcal{P})))$, all filterings π with $\pi(\mathsf{minus}) = [\,]$ are eliminated, since they contradict the weak decrease of the first minus-rule. Thus, while there exist $6 \cdot 6 \cdot 6 \cdot 3 \cdot 1 = 648$ possible argument filterings for the symbols minus, quot, QUOT, s, and $\mathsf{0}$, our algorithm reduces this set to only $|\Pi^t(\mathcal{P})| = 24$ candidates. For TRSs with more function symbols, the reduction of the search space is of course even more dramatic. Moreover, in successful proofs we only compute a small subset of $\Pi^t(\mathcal{P})$, since its elements are determined step by step in a depth-first search until a proof is found.

For innermost termination, the set of constraints to be satisfied depends on the argument filtering used. If $f \geq_d g$, then when orienting the rules of f, we do not necessarily have to orient the rules of g as well, since all occurrences of g in f-rules may have been deleted by the argument filtering, cf. Thm. 19. To formalize this, we define a relation "$\vdash_{\mathcal{P}}$" on sets of argument filterings. Let us extend rp_π to *partial* argument filterings by defining $rp_\pi(f) = \varnothing$ for all $f \notin DOM(\pi)$. Now $\mathcal{U}_{\mathcal{R}}(\mathcal{P}, \pi)$ is also defined for partial filterings by simply disregarding all subterms of function symbols where π is not defined.

For a partial argument filtering π, whenever $Rls_{\mathcal{R}}(f)$ is included in the usable rules $\mathcal{U}_{\mathcal{R}}(\mathcal{P}, \pi)$ for the cycle $\mathcal{P}$, then the relation "$\vdash_{\mathcal{P}}$" can extend π in order to make the f-rules weakly decreasing. We label each argument filtering by the set of those function symbols whose rules are already guaranteed to be weakly decreasing.

Definition 31 ($\vdash_{\mathcal{P}}$) *Each argument filtering π is labelled with a set $\mathcal{G} \subseteq \mathcal{D}$ and we denote a labelled argument filtering by $\pi_{\mathcal{G}}$. For sets of labelled argument* *filterings, we define the relation "$\vdash_{\mathcal{P}}$": $\mathit{\Pi} \uplus \{\pi_{\mathcal{G}}\} \vdash_{\mathcal{P}} \mathit{\Pi} \cup \{\pi'_{\mathcal{G}\cup\{f\}} \mid \pi' \in Ex_f(\pi)\}$, if $f \in \mathcal{D} \setminus \mathcal{G}$ and $Rls_{\mathcal{R}}(f) \subseteq \mathcal{U}_{\mathcal{R}}(\mathcal{P}, \pi)$.*

When proving innermost termination, we only regard argument filterings that result from $\mathit{\Pi}(\mathcal{P})$ by applying $\vdash_{\mathcal{P}}$-reductions as long as possible. So here we extend each $\pi \in \mathit{\Pi}(\mathcal{P})$ individually by $\vdash_{\mathcal{P}}$ instead of building $Ex_f(\mathit{\Pi}(\mathcal{P}))$ as in the termination case. The advantage of $\vdash_{\mathcal{P}}$ is that only those filterings π are extended to include f in their domain where this is is required by the usable rules $\mathcal{U}_{\mathcal{R}}(\mathcal{P}, \pi)$. For instance in Ex. 17, for cycles $\mathcal{P}$ containing the dependency pair $\mathsf{REV2}(x, \mathsf{cons}(y, l)) \to \mathsf{REV}(\mathsf{cons}(x, \mathsf{rev}(\mathsf{rev2}(y, l))))$, the filterings in $\mathit{\Pi}(\mathcal{P})$ are not defined on $\mathsf{rev1}$. When performing $\vdash_{\mathcal{P}}$-reductions, those $\pi \in \mathit{\Pi}(\mathcal{P})$ which eliminate the first argument of cons will never be extended to $\mathsf{rev1}$, whereas this is necessary for other filterings in $\mathit{\Pi}(\mathcal{P})$.

The following lemma is needed to prove that normal forms w.r.t. $\vdash_{\mathcal{P}}$ are unique. It states that $Ex_f(\pi)$ always consists of pairwise incompatible argument filterings. Here, two argument filterings π_1 and π_2 are *compatible* if $\pi_1(f) = \pi_2(f)$ for all $f \in DOM(\pi_1) \cap DOM(\pi_2)$, cf. [20].

Lemma 32 (Incompatibility) *Let T be a finite set of terms.*

(a) Let π, π_1, π_2 be (partial) argument filterings. Let $\pi_1, \pi_2 \in \{\pi' \mid \pi \sqsubseteq \pi'$ and π' is sufficiently defined for $T\}$, where π_1 is a $\sqsubseteq$-minimal element of this set. If π_1 and π_2 are compatible, then $\pi_1 \sqsubseteq \pi_2$.

(b) If $\pi_1, \pi_2 \in Ex_f(\pi)$ with $\pi_1 \neq \pi_2$, then π_1 and π_2 are incompatible, i.e., $Ex_f(\pi)$ consists of pairwise incompatible argument filterings.

(c) If $\mathit{\Pi}$ consists of pairwise incompatible argument filterings, then $Ex_f(\mathit{\Pi})$ consists of pairwise incompatible argument filterings, too.

Proof. (a) We perform induction on T using the (multi)set version of the proper subterm relation. If $T = \varnothing$, then the only minimal extension of π that is sufficiently defined for T is $\pi_1 = \pi$. Hence, $\pi_1 = \pi \sqsubseteq \pi_2$.

Next let $T = T' \uplus \{x\}$ for a variable x. Clearly, both π_1 and π_2 are also sufficiently defined for T' and moreover, π_1 is a minimal extension of π that is sufficiently defined for T'. Thus, the claim follows from the induction hypothesis.

If $T = T' \uplus \{f(t_1, \ldots, t_n)\}$, then $f \in DOM(\pi_1)$. Let $T'' = T' \cup \{t_j \mid j \in rp_{\pi_1}(f)\}$. Both π_1 and π_2 are sufficiently defined for T'' (for π_2 this follows from $\pi_2(f) = \pi_1(f)$ by compatibility of π_1 and π_2). If π_1 is a minimal extension of π that is sufficiently defined for T'', then the claim is implied by the induction hypothesis. Otherwise, we have $f \notin DOM(\pi)$ and we obtain the following minimal extension π'_1 of π that is sufficiently defined for T'': $DOM(\pi'_1) = DOM(\pi_1) \setminus \{f\}$ and $\pi'_1(g) = \pi_1(g)$ for all $g \in DOM(\pi'_1)$. Then the induction hypothesis implies $\pi'_1 \sqsubseteq \pi_2$. Since π_1 only differs from π'_1 on the function symbol f and since $\pi_1(f) = \pi_2(f)$, we obtain $\pi_1 \sqsubseteq \pi_2$.

(b) Let $\pi_1, \pi_2 \in Ex_f(\pi)$ be compatible. As both filterings are minimal extensions of π that are sufficiently defined for the terms on left- or right-hand sides of rules from $Rls_{\mathcal{R}}(f)$, we use (a) to conclude both $\pi_1 \sqsubseteq \pi_2$ and $\pi_2 \sqsubseteq \pi_1$, which implies $\pi_1 = \pi_2$.

(c) Let $\pi_1 \in Ex_f(\pi'_1)$ and $\pi_2 \in Ex_f(\pi'_2)$, where $\pi'_1, \pi'_2 \in \mathit{\Pi}$. If $\pi'_1 = \pi'_2$, then π_1 and π_2 are incompatible by (b). Otherwise $\pi'_1 \neq \pi'_2$, and by the assumption about Π we obtain that π'_1 and π'_2 are incompatible. As $\pi'_1 \sqsubseteq \pi_1$ and $\pi'_2 \sqsubseteq \pi_2$, this implies that π_1 and π_2 are incompatible as well. $\square$

The next theorem shows the desired properties of the relation $\vdash_{\mathcal{P}}$.

Theorem 33 *$\vdash_{\mathcal{P}}$ is terminating and confluent.*

Proof. The termination of $\vdash_{\mathcal{P}}$ is obvious as the labellings increase in every $\vdash_{\mathcal{P}}$-step. Hence for confluence, it suffices to show local confluence. The only crucial non-determinism in the definition of $\vdash_{\mathcal{P}}$ is the choice of f. Suppose that $f_0, f_1 \in \mathcal{D} \setminus \mathcal{G}$ with $f_0 \neq f_1$ and $Rls_{\mathcal{R}}(f_0) \cup Rls_{\mathcal{R}}(f_1) \subseteq \mathcal{U}_{\mathcal{R}}(\mathcal{P}, \pi)$ for some filtering π. This leads to two possible reduction steps

$$\begin{aligned} \Pi \uplus \{\pi_{\mathcal{G}}\} &\vdash_{\mathcal{P}} \Pi \cup \Pi_0, \quad \text{where } \Pi_0 = \{\pi^0_{\mathcal{G}\cup\{f_0\}} \mid \pi^0 \in Ex_{f_0}(\pi)\} \\ \Pi \uplus \{\pi_{\mathcal{G}}\} &\vdash_{\mathcal{P}} \Pi \cup \Pi_1, \quad \text{where } \Pi_1 = \{\pi^1_{\mathcal{G}\cup\{f_1\}} \mid \pi^1 \in Ex_{f_1}(\pi)\} \end{aligned}$$

Note that $\mathcal{U}_{\mathcal{R}}(\mathcal{P}, \pi) \subseteq \mathcal{U}_{\mathcal{R}}(\mathcal{P}, \pi^i)$ holds for all $\pi^i \in Ex_{f_i}(\pi)$. Thus, for all filterings $\pi^i_{\mathcal{G}\cup\{f_i\}} \in \Pi_i$, we have $f_{1-i} \in \mathcal{D}\setminus(\mathcal{G}\cup\{f_i\})$ and $Rls_{\mathcal{R}}(f_{1-i}) \subseteq \mathcal{U}_{\mathcal{R}}(\mathcal{P}, \pi^i)$. Hence, we can build the following reductions (where we also allow the application of Ex_f to labelled argument filterings by simply ignoring their labels).

$$\begin{array}{llll} \Pi \cup \Pi_0 & \overset{|\Pi_0|}{\vdash_{\mathcal{P}}} & (\Pi \setminus \Pi_0) & \cup \left\{\pi'_{\mathcal{G}\cup\{f_0,f_1\}} \mid \pi' \in Ex_{f_1}(Ex_{f_0}(\pi))\right\} \\ & \overset{|\Pi\cap\Pi_1|}{\vdash_{\mathcal{P}}} & (\Pi \setminus (\Pi_0 \cup \Pi_1)) & \cup \left\{\pi'_{\mathcal{G}\cup\{f_0,f_1\}} \mid \pi' \in Ex_{f_1}(Ex_{f_0}(\pi))\right\} \\ & & & \cup \left\{\pi'_{\mathcal{G}\cup\{f_1,f_0\}} \mid \pi' \in Ex_{f_0}(\Pi \cap \Pi_1)\right\} \end{array}$$

$$\begin{array}{llll} \Pi \cup \Pi_1 & \overset{|\Pi_1|}{\vdash_{\mathcal{P}}} & (\Pi \setminus \Pi_1) & \cup \left\{\pi'_{\mathcal{G}\cup\{f_1,f_0\}} \mid \pi' \in Ex_{f_0}(Ex_{f_1}(\pi))\right\} \\ & \overset{|\Pi\cap\Pi_0|}{\vdash_{\mathcal{P}}} & (\Pi \setminus (\Pi_1 \cup \Pi_0)) & \cup \left\{\pi'_{\mathcal{G}\cup\{f_1,f_0\}} \mid \pi' \in Ex_{f_0}(Ex_{f_1}(\pi))\right\} \\ & & & \cup \left\{\pi'_{\mathcal{G}\cup\{f_0,f_1\}} \mid \pi' \in Ex_{f_1}(\Pi \cap \Pi_0)\right\} \end{array}$$

where $Ex_{f_0}(\Pi \cap \Pi_1) \subseteq Ex_{f_0}(Ex_{f_1}(\pi))$, $Ex_{f_1}(\Pi \cap \Pi_0) \subseteq Ex_{f_1}(Ex_{f_0}(\pi))$. So to finish the proof we have to show $Ex_{f_0}(Ex_{f_1}(\pi)) = Ex_{f_1}(Ex_{f_0}(\pi))$. By symmetry, it suffices to prove $Ex_{f_0}(Ex_{f_1}(\pi)) \subseteq Ex_{f_1}(Ex_{f_0}(\pi))$. To this end, we only have to show that for every $\pi_{01} \in Ex_{f_0}(Ex_{f_1}(\pi))$ there exists a $\pi_{10} \in Ex_{f_1}(Ex_{f_0}(\pi))$ with $\pi_{10} \sqsubseteq \pi_{01}$. The reason is that in an analogous way one can show that for π_{10} there also exists a $\pi'_{01} \in Ex_{f_0}(Ex_{f_1}(\pi))$ with $\pi'_{01} \sqsubseteq \pi_{10}$. Hence, we have $\pi'_{01} \sqsubseteq \pi_{10} \sqsubseteq \pi_{01}$ and by Lemma 32 (b) and (c), this implies $\pi'_{01} = \pi_{10} = \pi_{01}$.

Let $\pi_{01} \in Ex_{f_0}(Ex_{f_1}(\pi))$. By the definition of Ex, there must be a $\pi_1 \in Ex_{f_1}(\pi)$ and a reduction pair $(\succsim, \succ) \in \mathcal{RP}$ with $\pi_1 \sqsubseteq \pi_{01}$ and $\pi_{01}(l) \succsim \pi_{01}(r)$ for all $l \to r \in Rls_{\mathcal{R}}(f_0)$. As $\pi_1 \in Ex_{f_1}(\pi)$, we may conclude in the same way that $\pi \sqsubseteq \pi_1$ and $\pi_1(l) \succsim' \pi_1(r)$ for all f_1-rules and some reduction pair $(\succsim', \succ') \in \mathcal{RP}$. Since $\pi \sqsubseteq \pi_{01}$ and since the f_0-rules can be oriented in a weakly decreasing way using π_{01}, there exists a $\pi_0 \in Ex_{f_0}(\pi)$ with $\pi \sqsubseteq \pi_0 \sqsubseteq \pi_{01}$ such that the f_0-rules can also be oriented using π_0. Since $\pi_0 \sqsubseteq \pi_{01}$ and since the f_1-rules can be oriented with π_{01}, there is a $\pi_{10} \in Ex_{f_1}(\pi_0)$ with $\pi_0 \sqsubseteq \pi_{10} \sqsubseteq \pi_{01}$ such that π_{10} also permits an orientation of the f_1-rules. As explained above, this suffices to prove $Ex_{f_0}(Ex_{f_1}(\pi)) \subseteq Ex_{f_1}(Ex_{f_0}(\pi))$. $\square$

Now we can define the set of argument filterings that are regarded for innermost termination proofs.

Definition 34 ($\Pi^i(\mathcal{P})$) *Let $Nf_{\vdash_\mathcal{P}}(\Pi)$ denote the normal form of Π w.r.t. $\vdash_\mathcal{P}$. Then we define $\Pi^i(\mathcal{P}) = Nf_{\vdash_\mathcal{P}}(\{\pi_\varnothing \mid \pi \in \Pi(\mathcal{P})\})$.*

To compute $\Pi^i(\mathcal{P})$, we again perform a depth-first search and start with some $\pi \in \Pi(\mathcal{P})$. Now π only has to be extended in order to make the rules for a symbol f weakly decreasing if the f-rules are contained in $\mathcal{U}_\mathcal{R}(\mathcal{P}, \pi)$. If by this extension, the rules for some new symbol g become usable, then a subsequent extension with Ex_g is also necessary, etc. Similar to the termination case, in this way the first elements of $\Pi^i(\mathcal{P})$ can be determined very quickly. If for one of these elements, the constraints (c) and (d) of Thm. 19 can already be solved, then no further consideration of the cycle $\mathcal{P}$ is necessary.

Thm. 35 states that $\Pi^t(\mathcal{P})$ resp. $\Pi^i(\mathcal{P})$ indeed contain all argument filterings which could possibly solve the dependency pair constraints. Here, $\mathcal{P}$ may also result from narrowing, rewriting, and instantiating dependency pairs. In this way the set of argument filterings is reduced dramatically and thus, efficiency is increased. For example, for a TRS from [3, Ex. 3.11] computing quicksort, $\Pi^t(\mathcal{P})$ reduces the number of argument filterings from more than 26 million to 3734 and with $\Pi^i(\mathcal{P})$ we obtain a reduction from more than 1.4 million to 783.

Theorem 35 *Let $\mathcal{P}$ be a cycle. If the constraints (a) and (b) of Thm. 14 for termination are satisfied by some reduction pair from $\mathcal{RP}$ and argument filtering π, then $\pi' \sqsubseteq \pi$ for some $\pi' \in \Pi^t(\mathcal{P})$. If the constraints (c) and (d) of Thm. 19 for innermost termination are satisfied by some reduction pair from $\mathcal{RP}$ and argument filtering π, then $\pi' \sqsubseteq \pi$ for some $\pi' \in \Pi^i(\mathcal{P})$.*

Proof. Let π be an argument filtering and let $(\succsim, \succ) \in \mathcal{RP}$ be a reduction pair that solve the constraints (a) and (b) from Thm. 14 or the constraints (c) and (d) from Thm. 19 for a cycle $\mathcal{P}$, respectively.

We first consider the termination case. There must be a minimal argument filtering $\pi_0 \in \Pi(\mathcal{P})$ with $\pi_0 \sqsubseteq \pi$ that solves the constraints in (a) using $(\succsim, \succ)$. Let $\mathcal{P}$ originate from $DP(\mathcal{R}_i)$, where $\mathcal{R}'_i$ has the defined symbols $\{f_1, \ldots, f_k\}$. As $\pi_0 \sqsubseteq \pi$ and $\pi(l) \succsim \pi(r)$ for all f_1-rules $l \to r$, there must be a filtering $\pi_1 \in Ex_{f_1}(\pi_0)$ with $\pi_1 \sqsubseteq \pi$. We continue in this way and obtain an argument filtering $\pi_k \in Ex_{f_k}(\ldots Ex_{f_1}(\Pi(\mathcal{P}))\ldots) = \Pi^t(\mathcal{P})$ with $\pi_k \sqsubseteq \pi$.

In the innermost case, let $\Pi(\mathcal{P}) = \Pi_0 \vdash_\mathcal{P} \Pi_1 \vdash_\mathcal{P} \ldots \vdash_\mathcal{P} \Pi_n = \Pi^i(\mathcal{P})$ be a $\vdash_\mathcal{P}$-reduction to normal form. We show that for all $0 \leq j \leq n$ there is a $\pi_j \in \Pi_j$ with $\pi_j \sqsubseteq \pi$ by induction on j. For $j = 0$, since π solves the constraints in (c), by definition there is again a minimal argument filtering $\pi_0 \in \Pi(\mathcal{P})$ with $\pi_0 \sqsubseteq \pi$. For $j > 0$, we assume that there is a $\pi_{j-1} \in \Pi_{j-1}$ with $\pi_{j-1} \sqsubseteq \pi$. Thus, we either have $\pi_{j-1} \in \Pi_j$ as well or else, Π_j results from Π_{j-1} by replacing π_{j-1} by all elements of $Ex_f(\pi_{j-1})$ for some f with $Rls_\mathcal{R}(f) \subseteq \mathcal{U}_\mathcal{R}(\mathcal{P}, \pi_{j-1})$. Since $\pi_{j-1} \sqsubseteq \pi$, we have $\mathcal{U}_\mathcal{R}(\mathcal{P}, \pi_{j-1}) \subseteq \mathcal{U}_\mathcal{R}(\mathcal{P}, \pi)$ and thus, π also makes the f-rules weakly decreasing by the constraints in (d). This implies that there must be a $\pi_j \in Ex_f(\pi_{j-1}) \subseteq \Pi_j$ with $\pi_j \sqsubseteq \pi$. ☐

The converse directions of this theorem do not hold, since in the computation of $\Pi^t(\mathcal{P})$ and $\Pi^i(\mathcal{P})$, when extending argument filterings, one does not take the orders into account. So even if $Ex_f(Ex_g(\ldots)) \neq \varnothing$, it could be that there is no reduction pair such that both f- and g-rules are weakly decreasing w.r.t. the *same* reduction pair from $\mathcal{RP}$.

However, the technique of this section can be extended by storing both argument filterings and corresponding parameters of the order in the sets $\Pi(\mathcal{P})$ and $Ex_f(\ldots)$. For example, if $\mathcal{RP}$ is the set of all LPOs, then $\Pi(\mathcal{P})$ would now contain all (minimal) pairs of argument filterings π and precedences such that $\pi(s) \succ_{LPO} \pi(t)$ resp. $\pi(s) \succsim_{LPO} \pi(t)$ holds for $s \to t \in \mathcal{P}$. When extending argument filterings, one would also have to extend the corresponding precedence. Of course, such an extension is only permitted if the extended precedence is still irreflexive (and hence, well founded). For other path orders like RPO(S), extensions can be defined in an analogous way. Then $\Pi^t(\mathcal{P})$ (resp. $\Pi^i(\mathcal{P})$) is non-empty iff the constraints for (innermost) termination are satisfiable for $\mathcal{P}$. Thus, after computing $\Pi^t(\mathcal{P})$ resp. $\Pi^i(\mathcal{P})$, no further checking of orders and constraints is necessary anymore.

7 Heuristics

Now we present heuristics to improve the efficiency of the search for argument filterings and base orders further. In contrast to the improvements of the preceding sections, these heuristics affect the power of the method, i.e., there exist examples whose (innermost) termination can no longer be proved when following the heuristics.

7.1 Type Inference for Argument Filterings

To reduce the sets $\Pi^t(\mathcal{P})$ and $\Pi^i(\mathcal{P})$ of potential argument filterings, we have developed the following heuristic based on *type inference*. In natural examples, termination of a function is usually due to the decrease of arguments of the *same* type. Of course, this type may be different for the different functions in a TRS. So termination of f may be due to arguments that are natural numbers, while termination of g may be due to arguments that are lists. We use a (monomorphic) type inference algorithm to transform a TRS into a sorted TRS (i.e., a TRS with rules $l \to r$ where l and r are well-typed terms of the same type). Then as a good heuristic to reduce the set of possible argument filterings further, one can require that for every symbol f, either no argument position is eliminated or all non-eliminated argument positions are of the same type. In other words, if f is n-ary, then $\pi(f) = [1, \ldots, n]$, $\pi(f) \in \{1, \ldots, n\}$, or $\pi(f) = [i_1, \ldots, i_k]$ where the argument positions $i_1, \ldots, i_k$ all have the same type. Our experiments show that all examples in the collections of [3, 9, 32] that can be solved using LPO as a base order can still be solved when using this heuristic.

7.2 Embedding Order for Dependency Pairs

To increase efficiency in our depth-first algorithm of Sect. 6, a successful heuristic is to only use the embedding order when orienting the constraints $\pi(s) \succ \pi(t)$ and $\pi(s) \succsim \pi(t)$ for dependency pairs $s \to t$. Only for constraints $\pi(l) \succsim \pi(r)$ for rules $l \to r$, one may apply more complicated quasi-orders like LPO, RPO(S), etc. The advantage of this approach is that now $\Pi(\mathcal{P})$ is much smaller than when using more powerful orders. Thus, the depth-first search starting with $\Pi(\mathcal{P})$ can be performed very quickly. Our experiments show that due to the improvements in Sect. 3 and 4, this heuristic succeeds for more than 96 % of those examples from [3, 9, 32] where a full LPO was successful, while reducing runtimes by at least 63 %.

7.3 Bottom-Up Heuristic

To determine argument filterings in Sect. 6, we start with the dependency pairs and treat the constraints for rules afterwards, where f-rules are considered before g-rules if $f >_d g$. In contrast, now we suggest a bottom-up approach which starts with determining an argument filtering for constructors and then moves upwards through the recursion hierarchy where g is treated before f if $f >_d g$. While in Sect. 6, we determined *sets* of argument filterings, now we only determine one single argument filtering, even if several ones are possible. To obtain an efficient technique, no backtracking takes place, i.e., if at some point one selects the "wrong" argument filtering, then the proof can fail.

More precisely, we first guess an argument filtering π which is only defined for constructors. For every n-ary constructor c we define $\pi(c) = [1, \ldots, n]$ or we let π filter away all arguments of c that do not have the same type as c's result. Afterwards, for every function symbol f, we try to extend π on f such that $\pi(l) \succsim \pi(r)$ for all f-rules $l \to r$. We consider functions according to the recursion hierarchy $>_d$. So when extending π on f, π is already defined on all $g <_d f$. Among the extensions of π which permit an orientation of the f-rules, we choose $\pi(f)$ such that it eliminates as many arguments of f as possible. Of course, this is just one of the potential argument filterings for f. If we have chosen the "wrong" argument filtering for f, the (innermost) termination proof might fail, although there would have been a solution with a different argument filtering. If we are not able to orient the rules of f, then we mark f as not orientable. Finally, after having treated all rules, the filtering is extended to the tuple symbols by trying to orient the dependency pairs as well (where at least one dependency pair must be strictly decreasing). Of course, this extension is done separately for every SCC or cycle, respectively.

In termination proofs, if $f \in \mathcal{R}_j$ is not orientable, then all symbols in $\mathcal{R}_i \geq_d \mathcal{R}_j$ as well as all dependency pairs resulting from $\mathcal{R}_i \geq_d \mathcal{R}_j$ are also not orientable. In innermost termination proofs, if f is not orientable, then a symbol that depends on f can still be orientable if one can extend the argument filtering in such a way that all occurrences of f in its rules are eliminated. Similarly, dependency pairs can still be orientable if the argument filtering eliminates all occurrences of f. Thus, here the bottom-up approach has the advantage that we already know that certain argument positions must be eliminated when extending the argument filtering to new function symbols.

This algorithm can also be modified by determining both the argument filtering and the reduction pair step by step. For example, if one uses reduction pairs based on LPO, then one always tries to extend the current precedence in a minimal way when proceeding from one function symbol to the next symbol in the recursion hierarchy. The combination with other orders works in a similar way.

The bottom-up algorithm reduces the search space enormously. The number of TRSs from [3, 9, 32] where the bottom-up algorithm succeeds is more than 77 % of the number achieved by the full dependency pair approach with LPO, but runtime is reduced to 32 %.

8 Using Polynomial Orders for Dependency Pairs

In Sect. 6 and 7 we showed how to mechanize the dependency pair approach with argument filterings and monotonic orders like LPO, RPO(S), KBO, etc. Now we discuss how to use reduction pairs based on polynomial orders instead, which are not necessarily monotonic if one also permits the coefficient 0 in polynomials. In contrast to RPO(S) and KBO, it is undecidable whether a set of constraints is satisfied by a polynomial order, and thus one can only use sufficient criteria when automating these orders. However, polynomial orders are often very powerful: with the results of this section, even very restrictive linear polynomial interpretations with coefficients from $\{0,1\}$ are sufficient to prove innermost termination of all examples in [3].

An advantage of polynomial orders is that one does not need any extra argument filtering anymore, since argument filtering can be simulated directly by the corresponding polynomials. If

$$\mathcal{P}ol(f(x_1, ..., x_n)) = a_1\, x_1^{b_{1,1}} \ldots x_n^{b_{1,n}} + \ldots + a_m\, x_1^{b_{m,1}} \ldots x_n^{b_{m,n}} \tag{43}$$

for coefficients $a_i \geq 0$, then this corresponds to the argument filtering $\pi_{\mathcal{P}ol}$ with $\pi_{\mathcal{P}ol}(f) = [j \mid a_i > 0 \wedge b_{i,j} > 0 \text{ for some } 1 \leq i \leq m]$. However, disregarding argument filterings is a problem in our new technique for innermost termination (Thm. 19), because the argument filtering π is needed in order to compute the constraints resulting from the usable rules $\mathcal{U}_{\mathcal{R}}(\mathcal{P}, \pi)$. Therefore, in Sect. 8.1 we show how to adapt this technique in the case of polynomial orders. In Sect. 8.2 we present another improvement of the dependency pair technique which can only be used for polynomial orders. It eliminates the indeterminism in the constraints of type (a) or (c) where one has to find a strictly decreasing dependency pair in each cycle.

8.1 Innermost Termination With Polynomial Orders

When automating Thm. 19 with polynomial orders, one fixes the degree of the polynomials in the polynomial interpretation and then suitable coefficients for the polynomials have to be found automatically. So one starts with an *abstract* polynomial interpretation $\mathcal{P}ol$. For every function symbol f, $\mathcal{P}ol(f(x_1, \ldots, x_n))$ is as in (43), but m and $b_{i,j}$ are fixed numbers, whereas a_i are *variable* coefficients. Then the constraints (c) of Thm. 19 would take the form

$$\begin{array}{ll} \bullet\ \mathcal{P}ol(s) - \mathcal{P}ol(t) > 0 & \text{for one pair } s \to t \text{ of } \mathcal{P} \\ \bullet\ \mathcal{P}ol(s) - \mathcal{P}ol(t) \geq 0 & \text{for all other pairs } s \to t \text{ of } \mathcal{P} \end{array} \tag{44}$$

An abstract polynomial interpretation $\mathcal{P}ol$ can be turned into a concrete polynomial interpretation by assigning a natural number to each variable coefficient a_i. We denote such *assignments* by α and let $\alpha(\mathcal{P}ol)$ denote the concrete polynomial interpretation resulting from this assignment. A set of constraints of the form $p \;{}_{(}\!\geq_{)}\; 0$ as above is *satisfiable* iff there exists an assignment α such that all instantiated constraints $\alpha(p) \;{}_{(}\!\geq_{)}\; 0$ hold. These instantiated constraints still contain the variables $x, y, \ldots$ that occurred in the dependency pairs and we say that $\alpha(p) \;{}_{(}\!\geq_{)}\; 0$ *holds* iff the inequality is true for all instantiations of the variables $x, y, \ldots$ by natural numbers. For example, $a_1 x + a_2 - a_3 y > 0$ is satisfied by the assignment α where $\alpha(a_1) = 1$, $\alpha(a_2) = 1$, and $\alpha(a_3) = 0$. The reason is that α turns the above constraint into $x + 1 > 0$ which holds for all instantiations of x and y with natural numbers.

The constraints of type (d) in Thm. 19 require $l \succsim r$ for all rules $l \to r \in \mathcal{U}_\mathcal{R}(\mathcal{P}, \pi_{\alpha(\mathcal{P}ol)})$. The problem is how to determine these constraints in the case of polynomial orders where the argument filtering $\pi_{\alpha(\mathcal{P}ol)}$ is not given explicitly but depends on the assignment α of natural numbers to variable coefficients a_i. Thus, $\pi_{\alpha(\mathcal{P}ol)}$ is not available yet when building the constraints although we would have to know it in order to compute $\mathcal{U}_\mathcal{R}(\mathcal{P}, \pi_{\alpha(\mathcal{P}ol)})$.

The solution is to translate the constraints of type (d) to polynomial constraints of the following form:

$$q \cdot (\mathcal{P}ol(l) - \mathcal{P}ol(r)) \geq 0 \;\text{ for all rules } l \to r \text{ of } \mathcal{U}_\mathcal{R}(\mathcal{P}) \tag{45}$$

Here, q will be a polynomial containing only variable coefficients a_i but no variables $x, y, \ldots$ from the rules of $\mathcal{R}$. So for any assignment α, $\alpha(q)$ is a number. We generate the constraints such that for *any* assignment α, we have $\alpha(q) = 0$ if $l \to r \notin \mathcal{U}_\mathcal{R}(\mathcal{P}, \pi_{\alpha(\mathcal{P}ol)})$. Thus, the constraints (45) are equivalent to requiring $\mathcal{P}ol(l) - \mathcal{P}ol(r) \geq 0$ for the rules $l \to r$ of $\mathcal{U}_\mathcal{R}(\mathcal{P}, \pi_{\alpha(\mathcal{P}ol)})$, but the advantage is that the constraints (45) with the variable coefficients a_i can be constructed before determining the assignment α of these variable coefficients.

Let $\mathcal{P}ol$ be an abstract polynomial interpretation as in (43). To generate the constraints (45), we first define a polynomial which sums up the coefficients of those monomials of $\mathcal{P}ol(f(x_1, ..., x_n))$ that contain x_j.

$$rp_{\mathcal{P}ol}(f, j) = \sum\nolimits_{1 \leq i \leq m,\, b_{i,j} > 0} \; a_i$$

So for any assignment α, $\alpha(rp_{\mathcal{P}ol}(f, j))$ is a number and this number is greater than 0 iff $j \in rp_{\pi_{\alpha(\mathcal{P}ol)}}(f)$. Now the set of constraints which corresponds to (d) in Thm. 19 can be built in an analogous way to the definition of usable rules (Def. 16).

Definition 36 (Usable Rules for Polynomial Orders) *Let $\mathcal{P}ol$ be an abstract polynomial interpretation. Again, let $\mathcal{R}' = \mathcal{R} \setminus Rls_\mathcal{R}(f)$. For any term t, we define the* usable rule constraints *$Con_\mathcal{R}(t, \mathcal{P}ol)$ as*

- $Con_\mathcal{R}(x, \mathcal{P}ol) = \varnothing$ *for $x \in \mathcal{V}$ and*
- $Con_\mathcal{R}(f(t_1, \ldots, t_n), \mathcal{P}ol) = \{\mathcal{P}ol(l) - \mathcal{P}ol(r) \geq 0 \mid l \to r \in Rls_\mathcal{R}(f)\}$
  $\cup \;\; \bigcup_{l \to r \in Rls_\mathcal{R}(f)} Con_{\mathcal{R}'}(r, \mathcal{P}ol)$
  $\cup \;\; \bigcup_{1 \leq j \leq n} \{rp_{\mathcal{P}ol}(f, j) \cdot p \;\geq\; 0 \mid p \geq 0 \in Con_{\mathcal{R}'}(t_j, \mathcal{P}ol)\}.$

For any term s, we define $Con^s_\mathcal{R}(t, \mathcal{P}ol) = \varnothing$ if t is a subterm of s and otherwise, it is defined like $Con_\mathcal{R}(f(t_1, ..., t_n), \mathcal{P}ol)$ where however, "$Con_{\mathcal{R}'}(t_j, \mathcal{P}ol)$" is replaced by "$Con^s_{\mathcal{R}'}(t_j, \mathcal{P}ol)$". For sets $\mathcal{P}$ of dependency pairs, let $Con_\mathcal{R}(\mathcal{P}, \mathcal{P}ol) = \bigcup_{s \to t \in \mathcal{P}} Con^s_\mathcal{R}(t, \mathcal{P}ol)$.

To improve efficiency, constraints like $Con_{\mathcal{R}}(t, \mathcal{P}ol)$ can often be simplified using equivalence-preserving transformations.

Example 37 Consider the TRS for list reversal from Ex. 17 again. We use a linear abstract polynomial interpretation $\mathcal{P}ol$ where $\mathcal{P}ol(\mathsf{nil}) = a_{\mathsf{nil}}$, $\mathcal{P}ol(f(x)) = a_{f,0} + a_{f,1}\, x$ for $f \in \{\mathsf{rev}, \mathsf{REV}\}$, and $\mathcal{P}ol(f(x, y)) = a_{f,0} + a_{f,1}\, x + a_{f,2}\, y$ for all other (binary) symbols f. Thus, $rp_{\mathcal{P}ol}(f, j) = a_{f,j}$ for all symbols f and positions j.

We now compute the usable rule constraints for the right-hand side of the dependency pair $\mathsf{REV2}(x, \mathsf{cons}(y, z)) \to \mathsf{REV}(\mathsf{rev2}(y, z))$. Let $\mathcal{R}' = \mathcal{R} \setminus Rls_{\mathcal{R}}(\mathsf{rev2})$ and $\mathcal{R}'' = \mathcal{R}' \setminus Rls_{\mathcal{R}'}(\mathsf{rev})$.

$$\begin{array}{ll}
Con_{\mathcal{R}}(\mathsf{REV}(\mathsf{rev2}(y,z)), \mathcal{P}ol) & = \{a_{\mathsf{REV},1} \cdot p \geq 0 \mid p \geq 0 \in Con_{\mathcal{R}}(\mathsf{rev2}(y,z), \mathcal{P}ol)\} \\
Con_{\mathcal{R}}(\mathsf{rev2}(y,z), \mathcal{P}ol) & = \{\mathcal{P}ol(l) - \mathcal{P}ol(r) \geq 0 \mid l \to r \in Rls_{\mathcal{R}}(\mathsf{rev2})\} \\
 & \quad \cup\ Con_{\mathcal{R}'}(\mathsf{rev}(\mathsf{cons}(x, ...)), \mathcal{P}ol) \\
Con_{\mathcal{R}'}(\mathsf{rev}(\mathsf{cons}(x, ...)), \mathcal{P}ol) & = \{\mathcal{P}ol(l) - \mathcal{P}ol(r) \geq 0 \mid l \to r \in Rls_{\mathcal{R}'}(\mathsf{rev})\} \\
 & \quad \cup\ Con_{\mathcal{R}''}(\mathsf{cons}(\mathsf{rev1}(...), ...), \mathcal{P}ol) \\
Con_{\mathcal{R}''}(\mathsf{cons}(\mathsf{rev1}(...), ...), \mathcal{P}ol) & = \{a_{\mathsf{cons},1} \cdot (\mathcal{P}ol(l) - \mathcal{P}ol(r)) \geq 0 \mid l \to r \in Rls_{\mathcal{R}''}(\mathsf{rev1})\}
\end{array}$$

So $Con_{\mathcal{R}}(\mathsf{REV}(\mathsf{rev2}(y, z)), \mathcal{P}ol)$ contains $a_{\mathsf{REV},1} \cdot (\mathcal{P}ol(l) - \mathcal{P}ol(r)) \geq 0$ for $\mathsf{rev2}$- and rev-rules and $a_{\mathsf{REV},1} \cdot a_{\mathsf{cons},1} \cdot (\mathcal{P}ol(l) - \mathcal{P}ol(r)) \geq 0$ for $\mathsf{rev1}$-rules.

This indicates that if cons is mapped to a polynomial which disregards its first argument (i.e., if $a_{\mathsf{cons},1} = 0$), then one does not have to require that the $\mathsf{rev1}$-rules are weakly decreasing. As shown in Ex. 20, this observation is crucial for the success of the innermost termination proof. It turns out that all constraints (for all cycles) are satisfied by the assignment α which maps $a_{\mathsf{cons},0}$, $a_{\mathsf{cons},2}$, $a_{\mathsf{rev},1}$, $a_{\mathsf{rev2},2}$, $a_{\mathsf{REV2},2}$, and $a_{\mathsf{REV},1}$ to 1 and all remaining variable coefficients to 0. So α turns $\mathcal{P}ol$ into a concrete polynomial interpretation where nil and $\mathsf{rev1}(x, y)$ are mapped to 0, $\mathsf{cons}(x, y)$ is mapped to $1 + y$, $\mathsf{rev}(x)$ and $\mathsf{REV}(x)$ are mapped to x, and both $\mathsf{rev2}(x, y)$ and $\mathsf{REV2}(x, y)$ are mapped to y.

The following lemma shows that $Con_{\mathcal{R}}$ indeed corresponds to the constraints resulting from the usable rules.

Lemma 38 (*Con* **and** $\mathcal{U}$) *Let $\mathcal{P}ol$ be an abstract polynomial interpretation and t be a term. An assignment α for $\mathcal{P}ol$'s coefficients satisfies $Con_{\mathcal{R}}(t, \mathcal{P}ol)$ iff α satisfies $\mathcal{P}ol(l) - \mathcal{P}ol(r) \geq 0$ for all $l \to r \in \mathcal{U}_{\mathcal{R}}(t, \pi_{\alpha(\mathcal{P}ol)})$. An analogous statement holds for $Con^{s}_{\mathcal{R}}(t, \mathcal{P}ol)$ and $\mathcal{U}^{s}_{\mathcal{R}}(t, \pi_{\alpha(\mathcal{P}ol)})$.*

Proof. We use induction over the sizes of $\mathcal{R}$ and t. If $t \in \mathcal{V}$, then the claim is trivial. Otherwise, let $t = f(t_1, \ldots, t_n)$. The assignment α satisfies $Con_{\mathcal{R}}(f(t_1, ..., t_n), \mathcal{P}ol)$ iff it satisfies $\mathcal{P}ol(l) - \mathcal{P}ol(r) \geq 0$ and $Con_{\mathcal{R}'}(r)$ for all $l \to r \in Rls_{\mathcal{R}}(f)$, and if it also satisfies $rp_{\mathcal{P}ol}(f, j) \cdot p \geq 0$ for all constraints $p \geq 0$ from $Con_{\mathcal{R}'}(t_j, \mathcal{P}ol)$ where $j \in \{1, \ldots, n\}$.

We have $\alpha(rp_{\mathcal{P}ol}(f, j)) \geq 0$ for all j, and $\alpha(rp_{\mathcal{P}ol}(f, j)) > 0$ iff $j \in rp_{\pi_{\alpha(\mathcal{P}ol)}}(f)$. So α satisfies the last requirement iff it satisfies $Con_{\mathcal{R}'}(t_j, \mathcal{P}ol)$ for $j \in rp_{\pi_{\alpha(\mathcal{P}ol)}}(f)$. Now the claim follows from the induction hypothesis. The proof for $Con^{s}_{\mathcal{R}}(t, \mathcal{P}ol)$ and $\mathcal{U}^{s}_{\mathcal{R}}(t, \pi_{\alpha(\mathcal{P}ol)})$ is analogous. $\square$

Now Thm. 19 for innermost termination can be reformulated to permit the use of reduction pairs based on polynomial orders.

Theorem 39 (Improved Innermost Termination with Polynomials) $\mathcal{R}$ *is innermost terminating if for any cycle* $\mathcal{P}$ *of the innermost dependency graph, there is an abstract polynomial interpretation* $\mathcal{P}ol$ *and an assignment* α *satisfying the constraints (44) and* $Con_{\mathcal{R}}(\mathcal{P}, \mathcal{P}ol)$*.*

Proof. The theorem follows from Thm. 19 using the polynomial order given by $\alpha(\mathcal{P}ol)$ as reduction pair and the argument filtering $\pi_{\alpha(\mathcal{P}ol)}$. Then (44) clearly corresponds to the constraints (c) in Thm. 19 and $Con_{\mathcal{R}}(\mathcal{P}, \mathcal{P}ol)$ corresponds to the constraints (d) by Lemma 38. □

8.2 Finding Strict Constraints Automatically

For (innermost) termination with dependency pairs and polynomial orders, we have to solve constraints like (44) which have the form

$$p_i \geq 0 \text{ for all } 1 \leq i \leq n \qquad \text{and} \qquad p_i > 0 \text{ for one } 1 \leq i \leq n \tag{46}$$

for polynomials p_i. The reason is that all dependency pairs in a cycle must be weakly decreasing but at least one has to be strictly decreasing. The basic approach for such constraints is to iterate over all n possible choices for the strict constraint. So in the worst case, the satisfiability checker for polynomial inequalities is called n times in order to find an assignment α satisfying (46). We present an equivalent, but much more efficient method where the satisfiability checker is only called once. The solution is to transform (46) into the following constraint.

$$p_i \geq 0 \text{ for all } 1 \leq i \leq n \qquad \text{and} \qquad \textstyle\sum_{1 \leq i \leq n} p_i > 0 \tag{47}$$

Theorem 40 ((46) iff (47)) *Let the* p_i *be variable disjoint, except for variable coefficients. An assignment* α *satisfies (46) iff it satisfies (47).*

Proof. If α satisfies (46), then w.l.o.g. we have $\alpha(p_1) > 0$. Using $\alpha(p_i) \geq 0$ for all $i \in \{2, \ldots n\}$ we immediately obtain $\alpha(\sum_{1 \leq i \leq n} p_i) > 0$.

For the other direction, let α satisfy (47) and assume that $\alpha(p_i) \not> 0$ for all $i \in \{1, \ldots, n\}$. Hence, for all i there exists a variable assignment β_i of the variables $x, y, \ldots$ in $\alpha(p_i)$ such that $\beta_i(\alpha(p_i)) = 0$. Since the polynomials $\alpha(p_i)$ are pairwise variable disjoint, the assignments β_i can be combined to one assignment β which coincides with each β_i on β_i's domain. Thus, $\beta(\alpha(p_i)) = 0$ for all i and therefore $\beta(\alpha(\sum_{1 \leq i \leq n} p_i)) = 0$. But then α cannot satisfy $\sum_{1 \leq i \leq n} p_i > 0$ which gives a contradiction. □

A well-known method to find variable assignments α satisfying polynomial constraints is the method of *partial derivation* [12, 26]. It transforms polynomial constraints as above into inequalities containing only variable coefficients, but no variables $x, y, \ldots$ from the terms or dependency pairs anymore. The main idea is that instead of $p_i(x_1, \ldots, x_n) \mathrel{\genfrac{}{}{0pt}{}{}{(\geq)}} 0$ one can impose the stronger requirements $p_i(0, x_2, \ldots, x_n) \mathrel{\genfrac{}{}{0pt}{}{}{(\geq)}} 0$ and $\frac{\partial p(x_1, \ldots, x_n)}{\partial x_1} \geq 0$. Repeating this process finally leads to a set of polynomial constraints without variables $x_1, \ldots, x_n$ which ensure that all partial derivations of all p_i are at least 0 and $p_i(0, \ldots, 0) \mathrel{\genfrac{}{}{0pt}{}{}{(\geq)}} 0$.

If the satisfiability checker for polynomial constraints uses the above partial derivation technique, then (47) can be simplified further to

$$p_i \geq 0 \text{ for all } 1 \leq i \leq n \qquad \text{and} \qquad \sum_{1 \leq i \leq n} p_i(0, \ldots, 0) > 0 \tag{48}$$

The reason is that then the constraints $p_i \geq 0$ ensure that all partial derivations of p_i must be at least 0. But then, the partial derivations of $\sum_{1 \leq i \leq n} p_i$ are also at least 0. Thus, it suffices to require $\sum_{1 \leq i \leq n} p_i > 0$ only for the instantiation of all variables $x, y, \ldots$ by 0.

Example 41 Regard the following cycle of the TRS from Ex. 15.

$$\mathsf{INT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{INT}(x, y)$$
$$\mathsf{INT}(\mathsf{0}, \mathsf{s}(y')) \to \mathsf{INT}(\mathsf{s}(\mathsf{0}), \mathsf{s}(y'))$$

We use a linear interpretation $\mathcal{P}ol(\mathsf{INT}(x, y)) = a_{\mathsf{INT},0} + a_{\mathsf{INT},1}\ x + a_{\mathsf{INT},2}\ y$, $\mathcal{P}ol(\mathsf{s}(x)) = a_{\mathsf{s},0} + a_{\mathsf{s},1}\ x$, and $\mathcal{P}ol(\mathsf{0}) = a_{\mathsf{0}}$. To make the dependency pairs decreasing, we obtain constraints like (46).

$$a_{\mathsf{INT},1}(a_{\mathsf{s},0} + a_{\mathsf{s},1}\ x - x) + a_{\mathsf{INT},2}(a_{\mathsf{s},0} + a_{\mathsf{s},1}\ y - y) \ {}_{(}\geq_{)}\ 0 \tag{49}$$
$$a_{\mathsf{INT},1}(a_{\mathsf{0}} - a_{\mathsf{s},0} - a_{\mathsf{s},1}\ a_{\mathsf{0}}) \ {}_{(}\geq_{)}\ 0 \tag{50}$$

Instead of choosing one of these constraints to be strict, with our refinement in (47) one obtains the above constraints with weak inequalities (i.e., with "$\geq$") and the additional constraint

$$a_{\mathsf{INT},1}(a_{\mathsf{s},1}\ x - x + a_{\mathsf{0}} - a_{\mathsf{s},1}\ a_{\mathsf{0}}) + a_{\mathsf{INT},2}(a_{\mathsf{s},0} + a_{\mathsf{s},1}\ y - y) > 0$$

If the satisfiability checker uses the partial derivation method, then the above constraint may be simplified by instantiating x and y with 0:

$$a_{\mathsf{INT},2}\ a_{\mathsf{s},0} + a_{\mathsf{INT},1}(a_{\mathsf{0}} - a_{\mathsf{s},1}\ a_{\mathsf{0}}) > 0 \tag{51}$$

Applying the method of partial derivation finally leads to the constraints (50), (51), and the following constraints arising from (49).

$$a_{\mathsf{INT},1}\ a_{\mathsf{s},0}\ +\ a_{\mathsf{INT},2}\ a_{\mathsf{s},0} \geq 0 \quad a_{\mathsf{INT},1}(a_{\mathsf{s},1} - 1) \geq 0 \quad a_{\mathsf{INT},2}(a_{\mathsf{s},1} - 1) \geq 0$$

These constraints are for example satisfied by the assignment α which maps $a_{\mathsf{s},0}$, $a_{\mathsf{s},1}$, and $a_{\mathsf{INT},2}$ to 1 and all other variable coefficients to 0.

9 Conclusion and Empirical Results

We presented improvements of the dependency pair approach which significantly reduce the sets of constraints $\pi(l) \succsim \pi(r)$ for both termination and innermost termination proofs. Moreover, we extended the applicability of dependency pair transformations and developed a criterion to ensure that their application is terminating without compromising the power of the approach in almost all examples. Subsequently, we introduced new techniques to implement the approach with polynomial orders and with monotonic orders like RPOS, respectively. For the latter type of orders, we developed an algorithm for computing argument filterings which is tailored to the improvements of dependency pairs presented before. In addition, we presented heuristics to increase efficiency which proved successful in large case studies.

A preliminary version of this paper appeared in [15]. The present article extends [15] substantially, e.g., by detailed proofs, by improved usable rules $\mathcal{U}^s_{\mathcal{R}}$ and the new dependency graph estimation EIDG**, by an improved technique for instantiating dependency pairs, by the extension of our results on completeness of dependency pair transformations to EDG* and EIDG**, by a new section on automating dependency pairs with polynomial orders, by a detailed description of our experiments, and by several additional explanations and examples.

We implemented the results of the paper in the system AProVE (Automated Program Verification Environment), available at `http://www-i2.informatik.rwth-aachen.de/AProVE`. The tool is written in Java and proofs can be performed both in a fully automated or in an interactive mode via a graphical user interface. We tested AProVE 1.0 on the examples of [3, 9, 32] (108 TRSs for termination, 151 TRSs for innermost termination) with the following techniques:

- *Normal* is the method of Sect. 3 – 6 with reduction pairs based on LPO or the embedding order. For LPO we allow different symbols to be equal in the precedence. Moreover, when computing the sets $\Pi^t(\mathcal{P})$ and $\Pi^i(\mathcal{P})$, we determine both the argument filterings and the precedences of the LPO, as illustrated at the end of Sect. 6. However, we do not yet apply the heuristics of Sect. 7.
- *Old+T* is *Normal*, but with Thm. 12 instead of Thm. 14 and 19.
- *Old* is like *Old+T*, but without the transformations of Sect. 5.
- *Type* is *Normal*, but with the type inference heuristic (Sect. 7.1).
- *Emb* is like *Normal*, but it applies the heuristic to use the embedding order for dependency pairs (Sect. 7.2).
- *Bottom-Up* uses the heuristic of Sect. 7.3 where we determine both the argument filtering and the reduction pair step by step.
- *Combi* is the following algorithm. For every SCC, it combines the heuristics of Sect. 7 as a pre-processing step and only calls the full dependency pair approach for SCCs where the heuristics fail. In this case, we use linear polynomial orders with coefficients from $\{0, 1\}$ according to the technique of Sect. 8.
  1. Safe transformations with Cases (1) and (2) of Def. 27
  2. Bottom-up heuristic of Sect. 7.3. For every SCC, we first use polynomial orders and if they fail, we use LPO.
  3. Heuristics of Sect. 7.1 and Sect. 7.2 with LPO as base order
  4. Full dependency pair approach with polynomial orders (Sect. 8)
  5. Remaining safe transformations according to Def. 27.
     If at least one transformation was applied, go back to 1.

  When the constraints for the SCC are solved, the algorithm is called recursively with the SCCs of those remaining pairs which were only weakly decreasing.

The following table shows success rates and runtimes for the different techniques and heuristics. The "Power" column contains the percentage of those examples in the collection where the proof attempt was successful. The "Time" column gives the overall time for running the system on all examples of the collection (also on the ones where the proof attempt failed). For each example we used a time-out of 30 seconds on a Pentium IV with 2.4 GHz and 1 GB memory. The detailed results of our experiments can be found in the appendix.

		Termination		Innermost T.	
Algorithm	Order	Power	Time	Power	Time
Old	EMB	37.0 %	18.7 s	51.0 %	27.9 s
Old+T	EMB	48.1 %	95.2 s	66.9 %	155.1 s
Normal	EMB	59.3 %	102.2 s	80.1 %	131.4 s
Old	LPO	63.0 %	144.4 s	64.9 %	205.1 s
Old+T	LPO	75.9 %	235.0 s	84.8 %	334.9 s
Normal	LPO	78.7 %	219.1 s	86.8 %	298.8 s
Type	LPO	78.7 %	203.0 s	87.4 %	275.1 s
Emb	LPO	75.9 %	79.8 s	84.8 %	102.5 s
Bottom-Up	LPO	61.1 %	67.2 s	74.8 %	96.0 s
Combi		94.4 %	16.4 s	98.0 %	34.8 s

Comparing the results for "*Old*" and "*Old+T*" indicates that transforming dependency pairs according to our heuristics in Sect. 5 increases power by 20 – 30 %. The step from "*Old+T*" to "*Normal*" shows the benefits of the results from Sect. 3 and 4, i.e., that Thm. 14 and 19 indeed improve upon Thm. 12 in practice. If one uses simple reduction pairs like the embedding order where orientability can be checked very efficiently, then compared to Thm. 12, Thm. 14 and 19 increase power by almost 20 % on the examples from [3, 9, 32]. For innermost termination, runtimes are decreased by about 15 %, while for termination one keeps approximately the same runtimes. If the reduction pairs are more complex (i.e., LPO), then Thm. 14 and 19 reduce runtime (by about 7 % for termination and about 11 % for innermost termination), while power is increased moderately.

The remainder of the table illustrates the usefulness of the heuristics of Sect. 7 and the combination algorithm described before. The type inference heuristic on its own does not improve the performance very much, but it also does not reduce the set of examples where the method is successful. With the embedding order heuristic from Sect. 7.2 we only lose a few examples in comparison to the full algorithm, but we need significantly less time. Using the bottom-up heuristic, there are several examples where we can no longer prove (innermost) termination, but we are at least three times faster than with the full approach. Finally, with the combined algorithm, we obtain the best of all methods. With this algorithm that integrates all results of the paper, our system succeeded on 98 % of the innermost termination examples (including all of [3]) and on 94.4 % of the examples for termination. The automated proof for the whole collection took 34.8 seconds for innermost termination and 16.4 seconds for termination. These results indicate that the contributions of the paper are indeed very useful in practice.

References

1. T. Arts. System description: The dependency pair method. In *Proc. 11th RTA*, pages 261–264, 2000. LNCS 1833.
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[5], RWTH, 2001.
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*, pages 346–364, 2000. LNAI 1831.
6. E. Contejean, C. Marché, B. Monate, and X. Urbain. Cime version 2, 2000. Available from `http://cime.lri.fr`.
7. N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
8. N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.
9. N. Dershowitz. 33 examples of termination. In *Proc. French Spring School of Theoretical Computer Science*, pages 16–26, 1995. LNCS 909.
10. N. Dershowitz, N. Lindenstrauss, Y. Sagiv, and A. Serebrenik. A general framework for automatic termination analysis of logic programs. *Appl. Algebra in Engineering, Communication and Computing*, 12(1,2):117–156, 2001.
11. 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, 2002.
12. J. Giesl. Generating polynomial orderings for termination proofs. In *Proc. 6th RTA*, volume 914 of *LNCS*, pages 426–431, 1995.
13. J. Giesl and T. Arts. Verification of Erlang processes by dependency pairs. *Appl. Algebra in Engineering, Communication & Comp.*, 12(1,2):39–72, 2001.
14. J. Giesl, T. Arts, and E. Ohlebusch. Modular termination proofs for rewriting using dependency pairs. *J. Symbolic Computation*, 34(1):21–58, 2002.
15. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Improving dependency pairs. In *Proc. 10th LPAR*, pages 165–179, 2003. LNAI 2850.
16. B. Gramlich. Abstract relations between restricted termination and confluence properties of rewrite systems. *Fundamenta Informaticae*, 24:3–23, 1995.
17. B. Gramlich. On proving termination by innermost termination. In *Proc. 7th RTA*, pages 97–107, 1996. LNCS 1103.
18. B. Gramlich. *Termination and Confluence Properties of Structured Rewrite Systems*. PhD thesis, Universität Kaiserslautern, Germany, 1996.
19. N. Hirokawa and A. Middeldorp. Approximating dependency graphs without using tree automata techniques. In *Proc. 6th WST*, pages 9–11, 2003.
20. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. In *Proc. 19th CADE*, 2003. LNAI 2741.
21. N. Hirokawa and A. Middeldorp. Tsukuba termination tool. In *Proc. 14th RTA*, pages 311–320, 2003. LNCS 2706.
22. G. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and System Sciences*, 25:239–299, 1982.
23. S. Kamin and J. J. Lévy. Two generalizations of the recursive path ordering. Unpublished Manuscript, University of Illinois, IL, USA, 1980.
24. D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. 1970.
25. K. Kusakari, M. Nakamura, and Y. Toyama. Argument filtering transformation. In *Proc. 1st PPDP*, pages 48–62, 1999. LNCS 1702.
26. D. Lankford. On proving term rewriting systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.
27. C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In *Proc. POPL '01*, pages 81–92, 2001.
28. A. Middeldorp. Approximating dependency graphs using tree automata techniques. In *Proc. IJCAR 2001*, pages 593–610, 2001. LNAI 2083.
29. A. Middeldorp. Approximations for strategies and termination. In *Proc. 2nd WRS*, 2002. ENTCS 70(6).
30. E. Ohlebusch. Hierarchical termination revisited. *Information Processing Letters*, 84(4):207–214, 2002.

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

31. E. Ohlebusch, C. Claves, and C. Marché. TALP: A tool for termination analysis of logic programs. In *Proc. 11th RTA*, pages 270–273, 2000. LNCS 1833.
32. J. Steinbach. Automatic termination proofs with transformation orderings. In *Proc. 6th RTA*, LNCS, pages 11–25, 1995. Full version appeared as Technical Report SR-92-23, Universität Kaiserslautern, Germany.
33. J. Steinbach. Simplification orderings: History of results. *Fundamenta Informaticae*, 24:47–87, 1995.
34. R. Thiemann and J. Giesl. Size-change termination for term rewriting. In *Proc. 14th RTA*, pages 264–278, 2003. LNCS 2706.
35. Y. Toyama. Counterexamples to the termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, 1987.
36. X. Urbain. Automated incremental termination proofs for hierarchically defined term rewriting systems. In *Proc. IJCAR 2001*, pages 485–498, 2001. LNAI 2083.

A Detailed Experiments

The following tables show the detailed results of our experiments on the examples of [3, 9, 32]. In the left column of each table, we mention the number of the TRS. Here, D.x is example x of [9], S.x is example x of [32], and $n.x$ is example x in Section n of [3]. Since some TRSs are contained in several collections and since we regarded every TRS only once, there are some numbers missing.

For each of the methods discussed in Sect. 9 there are two columns. The first one denotes how much time was spent for the proof attempt and the second one denotes the result of the proof attempt: an "**OK**" denotes a successful proof, the infinity sign "∞" denotes a timeout after 30 seconds, and a "**-**" denotes a failure of the proof attempt. At the end of each column we wrote the total time needed for the corresponding method and the number of successful proof attempts.

Table 1. Termination

Algorithm Order	Old EMB	Old+T EMB	Normal EMB	Old LPO	Old+T LPO	Normal LPO	Type LPO	Emb LPO	Bottom-Up LPO	Combi
3.1	0.06 OK	0.07 OK	0.07 OK	0.06 OK	0.08 OK	0.08 OK	0.08 OK	0.08 OK	0.06 OK	0.06 OK
3.2	0.06 OK	0.08 OK	0.08 OK	0.07 OK	0.09 OK	0.09 OK	0.09 OK	0.09 OK	0.07 OK	0.05 OK
3.3	0.14 -	0.44 -	0.47 -	0.39 OK	0.46 OK	0.33 OK	0.32 OK	0.24 OK	0.17 OK	0.13 OK
3.4	0.05 -	0.05 -	0.04 -	0.46 OK	0.47 OK	0.13 OK	0.13 OK	0.15 OK	0.07 -	0.10 OK
3.5	0.46 -	1.10 -	0.46 OK	1.43 OK	1.50 OK	1.48 OK	0.99 OK	0.51 OK	0.18 OK	0.15 OK
3.5a	0.47 -	1.88 -	0.47 OK	1.44 OK	1.52 OK	1.49 OK	1.01 OK	0.54 OK	0.20 OK	0.15 OK
3.5b	0.18 -	0.48 -	4.81 -	3.41 OK	3.56 OK	3.12 OK	1.55 OK	0.66 OK	0.27 OK	0.20 OK
3.6	0.43 -	3.38 -	3.44 -	4.62 -	30.00 ∞	30.00 ∞	30.00 ∞	3.73 -	2.46 -	0.22 OK
3.6a	0.41 -	1.67 -	1.64 -	4.26 -	21.22 -	20.92 -	12.40 -	1.84 -	2.02 -	0.20 OK
3.6b	0.21 -	0.69 -	1.26 -	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	7.00 -	3.04 -	0.27 OK
3.7	0.04 OK	0.06 OK	0.06 OK	0.05 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.04 OK
3.8	0.09 OK	0.12 OK	0.12 OK	0.10 OK	0.14 OK	0.14 OK	0.14 OK	0.13 OK	0.11 OK	0.09 OK
3.8a	0.10 OK	0.13 OK	0.13 OK	0.12 OK	0.15 OK	0.15 OK	0.15 OK	0.15 OK	0.12 OK	0.10 OK
3.8b	0.18 -	0.66 -	0.50 -	0.96 OK	1.07 OK	0.49 OK	0.45 OK	0.40 OK	0.25 OK	0.19 OK
3.9	0.25 -	1.38 -	0.34 OK	0.80 OK	0.92 OK	0.67 OK	0.63 OK	0.39 OK	0.23 OK	0.19 OK
3.10	0.50 -	1.47 -	9.31 -	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	10.44 -	9.36 -	0.92 OK
3.11	0.38 -	1.37 -	0.73 OK	2.13 OK	2.39 OK	1.63 OK	1.43 OK	0.88 OK	0.59 OK	0.47 OK
3.12	0.08 -	0.26 -	0.24 -	0.24 -	0.82 -	0.82 -	0.78 -	0.30 -	0.21 -	0.09 OK
3.13	7.78 -	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	0.78 OK
3.14	0.12 -	1.28 -	1.27 -	0.33 -	1.28 OK	1.28 OK	1.29 OK	1.38 -	0.95 OK	0.09 OK
3.15	0.03 -	0.03 -	0.03 -	0.05 -	0.04 -	0.04 -	0.04 -	0.04 -	0.04 -	0.03 OK
3.16	0.06 -	0.07 -	0.04 -	0.13 OK	0.13 OK	0.09 OK	0.09 OK	0.09 OK	0.05 OK	0.05 OK
3.17	0.05 -	0.05 -	0.05 -	0.79 -	1.39 OK	0.87 OK	0.88 OK	0.43 OK	0.29 -	0.08 OK
3.17a	0.32 -	0.34 -	0.05 -	2.57 -	30.00 ∞	30.00 ∞	30.00 ∞	1.99 -	0.49 -	0.09 OK
3.18	0.05 -	0.05 -	0.05 -	0.26 OK	0.27 OK	0.10 OK	0.11 OK	0.11 OK	0.07 OK	0.09 OK
3.19	0.05 -	0.05 -	0.05 -	0.30 OK	0.36 OK	0.19 OK	0.19 OK	0.19 OK	0.13 OK	0.15 OK
3.20	0.04 -	0.10 OK	0.09 OK	0.18 -	0.24 OK	0.24 OK	0.24 OK	0.11 OK	0.09 OK	0.03 OK
3.21	0.06 -	0.11 OK	0.11 OK	0.40 -	0.46 OK	0.46 OK	0.46 OK	0.12 OK	0.10 -	0.29 OK
3.22	0.06 -	0.07 -	0.04 -	0.21 -	0.22 -	2.74 -	2.74 -	0.29 -	0.11 -	0.52 -

Algorithm Order	Old EMB	Old+T EMB	Normal EMB	Old LPO	Old+T LPO	Normal LPO	Type LPO	Emb LPO	Bottom-Up LPO	Combi
3.23	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK
3.24	0.04 -	0.16 -	0.16 -	0.07 -	0.19 -	0.19 -	0.19 -	0.18 -	0.17 -	0.04 OK
3.25	0.03 OK	0.04 OK	0.04 OK	0.04 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.09 -	0.07 OK
3.26	0.03 -	0.13 -	0.13 -	0.03 OK	0.04 OK	0.04 OK	0.04 OK	0.13 -	0.03 OK	0.02 OK
3.27	0.03 -	0.08 -	0.08 -	0.03 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.07 -	0.08 OK
3.28	0.07 -	0.08 -	0.08 -	0.15 OK	0.16 OK	0.16 OK	0.14 OK	0.17 OK	0.07 -	0.49 OK
3.29	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
3.30	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK
3.31	0.02 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK
3.32	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
3.33	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.01 OK
3.34	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.03 OK	0.03 OK	0.02 OK	0.02 OK	0.03 -	0.04 OK
3.35	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.02 OK
3.36	0.08 -	0.68 -	0.57 OK	0.22 OK	0.33 OK	0.33 OK	0.33 OK	0.65 OK	0.41 OK	0.11 OK
3.37	0.04 OK	0.05 OK	0.04 OK	0.04 OK	0.05 OK	0.05 OK	0.05 OK	0.04 OK	0.04 OK	0.03 OK
3.38	0.12 -	0.17 -	0.13 OK	0.55 OK	0.59 OK	0.53 OK	0.54 OK	0.15 OK	0.11 OK	0.08 OK
3.39	0.09 -	0.09 -	0.32 -	0.48 -	0.49 -	0.69 -	0.70 -	0.37 -	0.24 -	0.27 OK
3.40	0.10 -	0.10 -	1.06 -	0.38 -	0.39 -	2.93 -	2.94 -	1.18 -	0.80 -	0.40 OK
3.41	0.03 -	0.03 OK	0.03 OK	0.04 -	0.04 OK	0.04 OK	0.03 OK	0.04 OK	0.04 OK	0.03 OK
3.42	0.07 -	0.13 OK	0.13 OK	0.09 -	0.16 OK	0.16 OK	0.16 OK	0.15 OK	0.15 OK	0.18 OK
3.43	0.03 -	0.04 OK	0.04 OK	0.05 -	0.06 OK	0.06 OK	0.06 OK	0.04 OK	0.04 OK	0.08 OK
3.44	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.02 OK	0.01 OK	0.02 OK	0.01 OK	0.01 OK
3.45	0.04 OK	0.05 OK	0.05 OK	0.04 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.04 OK
3.46	0.02 -	0.02 OK	0.02 OK	0.02 -	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.01 OK
3.47	0.05 OK	0.06 OK	0.06 OK	0.09 OK	0.10 OK	0.10 OK	0.09 OK	0.10 OK	0.04 OK	0.03 OK
3.48	0.15 -	0.16 -	0.07 -	2.83 OK	2.90 OK	1.17 OK	0.96 OK	0.81 -	0.07 -	0.68 OK
3.49	0.07 -	0.09 -	0.09 -	0.17 -	0.19 -	0.19 -	0.14 -	0.09 -	0.07 -	0.13 OK
3.50	0.03 -	0.04 -	0.03 OK	0.09 OK	0.09 OK	0.03 OK	0.03 OK	0.03 OK	0.04 -	0.06 OK
3.51	0.05 -	0.05 -	0.05 -	0.30 OK	0.30 OK	0.13 OK	0.13 OK	0.13 OK	0.06 -	0.10 OK
3.52	0.03 OK	0.04 OK	0.04 OK	0.05 OK	0.06 OK	0.06 OK	0.06 OK	0.05 OK	0.04 OK	0.02 OK

Algorithm Order	Old EMB	Old+T EMB	Normal EMB	Old LPO	Old+T LPO	Normal LPO	Type LPO	Emb LPO	Bottom-Up LPO	Combi
3.53	0.29 -	1.49 -	1.48 -	0.51 -	2.34 -	2.34 -	2.36 -	1.61 -	1.45 -	0.27 OK
3.53a	0.03 -	0.05 -	0.06 -	0.03 -	0.06 -	0.07 -	0.06 -	0.07 -	0.09 -	0.02 OK
3.53b	0.03 OK	0.04 OK	0.04 OK	0.05 OK	0.06 OK	0.06 OK	0.06 OK	0.05 OK	0.05 OK	0.02 OK
3.54	0.03 OK	0.05 OK	0.05 OK	0.04 OK	0.06 OK	0.06 OK	0.07 OK	0.05 OK	0.09 -	0.07 OK
3.55	0.53 -	1.69 -	0.87 OK	2.26 OK	2.69 OK	1.94 OK	1.66 OK	1.04 OK	0.70 OK	0.56 OK
3.56	0.07 -	0.11 -	0.08 OK	0.12 -	0.18 -	0.10 OK	0.09 OK	0.09 OK	0.05 -	0.14 OK
3.57	0.47 -	0.50 -	0.06 -	3.42 -	5.70 OK	0.98 OK	0.99 OK	0.65 OK	0.10 -	0.18 OK
D.1	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK
D.2	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.02 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK
D.3	0.02 OK	0.02 OK	0.03 OK	0.02 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK	0.01 OK
D.6	0.02 -	0.02 -	0.02 -	0.02 OK	0.02 OK	0.03 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK
D.7	0.05 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.05 OK	0.05 OK	0.05 OK	0.05 -	0.07 OK
D.8	0.09 OK	0.11 OK	0.10 OK	0.09 OK	0.10 OK	0.11 OK	0.11 OK	0.10 OK	0.12 OK	0.09 OK
D.9	0.03 -	0.03 -	0.03 -	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 -	0.04 OK
D.11	0.40 OK	0.41 OK	0.41 OK	0.41 OK	0.45 OK	0.45 OK	0.46 OK	0.45 OK	0.76 OK	0.41 OK
D.12	0.04 -	0.05 -	0.05 -	0.04 -	0.05 -	0.05 -	0.05 -	0.05 -	0.07 -	0.09 -
D.13	0.07 -	0.07 -	0.06 -	0.10 -	0.11 -	0.10 -	0.09 -	0.10 -	0.06 -	0.16 -
D.17	0.04 -	0.04 -	0.04 -	0.18 OK	0.19 OK	0.12 OK	0.12 OK	0.10 OK	0.05 OK	0.06 OK
D.18	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK
D.20	0.13 -	30.00 ∞	30.00 ∞	0.17 OK	0.79 OK	0.79 OK	0.78 OK	0.70 OK	0.69 OK	0.68 OK
D.21	0.07 -	0.09 OK	0.09 OK	0.08 -	0.09 OK	0.09 OK	0.09 OK	0.09 OK	0.10 OK	0.07 OK
D.28	0.06 -	0.07 -	0.07 -	0.05 OK	0.06 OK	0.05 OK	0.05 OK	0.05 OK	0.06 -	0.07 OK
D.29	0.06 -	0.44 -	0.10 OK	0.06 OK	0.10 OK	0.10 OK	0.10 OK	0.11 OK	0.10 OK	0.08 OK
D.30	0.08 -	5.52 -	5.51 -	0.23 OK	0.41 OK	0.41 OK	0.41 OK	0.25 OK	5.63 -	0.26 OK
D.32	0.16 OK	0.19 OK	0.20 OK	0.20 OK	0.24 OK	0.23 OK	0.21 OK	0.21 OK	0.17 OK	0.11 OK
D.33	0.07 -	0.08 -	0.23 -	3.08 -	3.49 -	1.48 -	1.25 -	0.57 -	0.09 -	0.79 -
S.1	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
S.2	0.03 -	0.10 OK	0.10 OK	3.47 -	6.44 OK	6.36 OK	3.61 OK	2.88 OK	0.05 -	0.62 OK
S.3	0.06 OK	0.07 OK	0.07 OK	0.07 OK	0.07 OK	0.07 OK	0.08 OK	0.07 OK	0.07 OK	0.05 OK
S.4	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK

Algorithm Order	Old EMB	Old+T EMB	Normal EMB	Old LPO	Old+T LPO	Normal LPO	Type LPO	Emb LPO	Bottom-Up LPO	Combi
S.5	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.04 OK	0.03 OK	0.03 OK	0.03 OK
S.6	0.01 OK	0.02 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
S.7	0.04 OK	0.04 OK	0.05 OK	0.04 OK	0.05 OK	0.05 OK	0.04 OK	0.05 OK	0.04 OK	0.02 OK
S.10	0.03 -	0.03 OK	0.04 OK	0.03 -	0.04 OK	0.04 OK	0.04 OK	0.03 OK	0.04 OK	0.03 OK
S.11	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
S.12	0.02 OK	0.01 OK	0.02 OK	0.01 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.01 OK
S.14	0.08 -	0.09 -	0.09 -	1.09 -	1.12 -	1.12 -	1.12 -	0.10 -	0.07 -	0.16 -
S.15	0.04 -	0.10 -	0.10 -	0.05 -	0.12 -	0.12 -	0.12 -	0.11 -	0.11 -	0.16 -
S.17	0.06 -	0.08 -	0.08 -	0.13 -	0.14 -	0.15 -	0.12 -	0.09 -	0.06 -	0.15 OK
S.18	0.05 -	0.06 -	0.06 -	0.07 -	0.08 -	0.08 -	0.08 -	0.07 -	0.09 -	0.13 OK
S.22	0.07 -	0.13 OK	0.13 OK	0.09 -	0.16 OK	0.16 OK	0.16 OK	0.14 OK	0.15 OK	0.16 OK
S.24	0.20 -	0.64 -	0.16 OK	3.15 -	7.49 -	2.27 OK	1.82 OK	0.18 OK	0.13 OK	0.11 OK
S.25	0.03 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.03 OK
S.26	0.44 -	0.89 -	0.42 OK	2.02 -	3.33 -	0.89 OK	0.90 OK	0.47 OK	0.17 OK	0.11 OK
S.27	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
S.28	0.05 -	0.05 -	0.05 -	0.13 -	0.14 -	0.14 -	0.14 -	0.06 -	0.06 -	0.20 OK
S.29	0.07 OK	0.08 OK	0.08 OK	0.07 OK	0.08 OK	0.09 OK	0.08 OK	0.08 OK	0.08 OK	0.06 OK
S.30	0.23 -	0.09 OK	0.09 OK	0.33 OK	0.10 OK	0.10 OK	0.10 OK	0.10 OK	0.10 OK	0.07 OK
S.31	0.17 -	1.55 OK	1.56 OK	0.49 -	2.79 OK	2.79 OK	2.04 OK	1.72 OK	0.61 -	1.09 OK
Total: 108	18.71 40	95.22 52	102.25 64	144.35 68	234.99 82	219.12 85	202.97 85	79.81 82	67.20 66	16.41 102

Table 2. Innermost Termination

Algorithm Order	Old EMB	Old+T EMB	Normal EMB	Old LPO	Old+T LPO	Normal LPO	Type LPO	Emb LPO	Bottom-Up LPO	Combi
3.1	0.06 OK	0.07 OK	0.07 OK	0.11 OK	0.08 OK	0.08 OK	0.08 OK	0.08 OK	0.06 OK	0.05 OK
3.2	0.06 OK	0.08 OK	0.08 OK	0.07 OK	0.09 OK	0.09 OK	0.09 OK	0.09 OK	0.07 OK	0.05 OK
3.3	0.15 -	0.45 -	0.47 -	0.44 OK	0.45 OK	0.33 OK	0.31 OK	0.24 OK	0.17 OK	0.13 OK
3.4	0.12 -	0.60 -	0.17 OK	0.14 OK	0.18 OK	0.18 OK	0.19 OK	0.19 OK	0.14 OK	0.12 OK
3.5	0.46 -	1.10 -	0.46 OK	1.46 OK	1.50 OK	1.46 OK	0.98 OK	0.51 OK	0.18 OK	0.14 OK
3.5a	0.47 -	1.88 -	0.47 OK	1.47 OK	1.50 OK	1.48 OK	1.00 OK	0.54 OK	0.21 OK	0.16 OK
3.5b	0.18 -	0.48 -	4.83 -	3.49 OK	3.50 OK	3.09 OK	1.54 OK	0.66 OK	0.27 OK	0.20 OK
3.6	0.43 -	3.34 -	3.44 -	4.74 -	30.00 ∞	30.00 ∞	30.00 ∞	3.72 -	2.43 -	0.22 OK
3.6a	0.41 -	1.66 -	1.65 -	4.38 -	20.80 -	20.83 -	12.34 -	1.83 -	2.01 -	0.20 OK
3.6b	0.21 -	0.69 -	1.26 -	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	6.96 -	3.00 -	0.28 OK
3.7	0.04 OK	0.06 OK	0.06 OK	0.05 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.04 OK
3.8	0.09 OK	0.12 OK	0.12 OK	0.11 OK	0.13 OK	0.14 OK	0.14 OK	0.13 OK	0.11 OK	0.09 OK
3.8a	0.10 OK	0.13 OK	0.13 OK	0.12 OK	0.15 OK	0.15 OK	0.15 OK	0.15 OK	0.12 OK	0.10 OK
3.8b	0.17 -	0.65 -	0.50 -	0.97 OK	1.04 OK	0.48 OK	0.45 OK	0.39 OK	0.25 OK	0.20 OK
3.9	0.25 -	1.38 -	0.34 OK	0.80 OK	0.90 OK	0.67 OK	0.63 OK	0.39 OK	0.24 OK	0.19 OK
3.10	0.59 -	1.49 -	9.31 -	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	10.28 -	9.15 -	0.90 OK
3.11	0.36 -	1.41 -	0.73 OK	2.25 OK	2.33 OK	1.62 OK	1.40 OK	0.87 OK	0.54 OK	0.45 OK
3.12	0.08 -	0.25 -	0.24 -	0.28 -	0.80 -	0.81 -	0.77 -	0.29 -	0.22 -	0.07 OK
3.13	7.77 -	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	30.00 ∞	0.75 OK
3.14	0.12 -	1.28 -	1.27 -	0.34 -	1.27 OK	1.28 OK	1.27 OK	1.36 -	0.94 OK	0.09 OK
3.15	0.04 -	0.05 -	0.04 -	0.04 -	0.05 -	0.05 -	0.05 -	0.05 -	0.06 -	0.04 OK
3.16	0.05 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.07 OK	0.06 OK
3.17	0.15 -	0.74 -	0.73 -	0.30 -	1.32 OK	1.34 OK	1.34 OK	0.68 OK	0.57 -	0.13 OK
3.17a	0.21 -	1.43 -	1.43 -	0.36 -	30.00 ∞	30.00 ∞	30.00 ∞	2.21 -	1.14 -	0.22 OK
3.18	0.12 -	1.02 -	0.18 OK	0.13 OK	0.19 OK	0.21 OK	0.21 OK	0.19 OK	0.19 OK	0.15 OK
3.19	0.17 -	1.06 -	0.23 OK	0.18 OK	0.25 OK	0.27 OK	0.27 OK	0.25 OK	0.22 OK	0.19 OK
3.20	0.04 -	0.10 OK	0.11 OK	0.19 -	0.24 OK	0.24 OK	0.24 OK	0.11 OK	0.09 OK	0.03 OK
3.21	0.06 -	0.12 OK	0.11 OK	0.41 -	0.46 OK	0.46 OK	0.46 OK	0.12 OK	0.10 -	0.18 OK
3.22	0.05 OK	0.05 OK	0.05 OK	0.07 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.06 OK	0.04 OK

Algorithm Order	Old EMB	Old+T EMB	Normal EMB	Old LPO	Old+T LPO	Normal LPO	Type LPO	Emb LPO	Bottom-Up LPO	Combi
3.23	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK
3.24	0.04 -	0.17 -	0.16 -	0.07 -	0.18 -	0.19 -	0.19 -	0.17 -	0.17 -	0.04 OK
3.25	0.03 OK	0.05 OK	0.04 OK	0.04 OK	0.05 OK	0.05 OK	0.05 OK	0.04 OK	0.09 -	0.07 OK
3.26	0.03 -	0.23 -	0.23 -	0.08 OK	0.05 OK	0.05 OK	0.05 OK	0.25 -	0.05 OK	0.04 OK
3.27	0.01 OK	0.01 OK	0.01 OK	0.05 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.02 OK	0.01 OK
3.28	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.07 OK	0.08 OK	0.07 OK	0.09 -	0.11 OK
3.29	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
3.30	0.01 OK	0.01 OK	0.01 OK	0.02 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK
3.31	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK
3.32	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
3.33	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.01 OK
3.34	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
3.35	0.03 OK	0.04 OK	0.03 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.02 OK
3.36	0.08 -	0.68 -	0.57 OK	0.23 OK	0.31 OK	0.33 OK	0.33 OK	0.65 OK	0.41 OK	0.11 OK
3.37	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.05 OK	0.05 OK	0.05 OK	0.04 OK	0.04 OK	0.03 OK
3.38	0.12 -	0.16 -	0.14 OK	0.56 OK	0.58 OK	0.53 OK	0.53 OK	0.15 OK	0.11 OK	0.07 OK
3.39	0.09 OK	0.12 OK	0.11 OK	0.14 OK	0.12 OK	0.13 OK	0.13 OK	0.12 OK	0.11 OK	0.08 OK
3.40	0.13 OK	0.15 OK	0.15 OK	0.14 OK	0.16 OK	0.16 OK	0.16 OK	0.16 OK	0.14 OK	0.12 OK
3.41	0.03 -	0.03 OK	0.03 OK	0.03 -	0.04 OK	0.04 OK	0.04 OK	0.03 OK	0.04 OK	0.02 OK
3.42	0.07 -	0.14 OK	0.14 OK	0.09 -	0.15 OK	0.16 OK	0.16 OK	0.14 OK	0.15 OK	0.16 OK
3.43	0.03 -	0.05 OK	0.05 OK	0.07 -	0.07 OK	0.07 OK	0.07 OK	0.06 OK	0.06 OK	0.09 OK
3.44	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.03 OK	0.02 OK	0.03 OK	0.02 OK	0.02 OK	0.01 OK
3.45	0.04 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.04 OK
3.46	0.02 -	0.02 OK	0.02 OK	0.08 -	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.01 OK
3.47	0.05 OK	0.05 OK	0.05 OK	0.06 OK	0.06 OK	0.06 OK	0.07 OK	0.06 OK	0.05 OK	0.04 OK
3.48	0.58 -	5.04 -	4.24 -	1.43 OK	1.53 OK	0.72 OK	0.50 OK	4.60 -	4.36 -	0.86 OK
3.49	0.09 -	0.11 -	0.10 -	0.19 -	0.20 -	0.20 -	0.15 -	0.11 -	0.09 -	0.12 OK
3.50	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.01 OK
3.51	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.05 OK	0.03 OK
3.52	0.03 OK	0.04 OK	0.04 OK	0.04 OK	0.05 OK	0.06 OK	0.06 OK	0.05 OK	0.04 OK	0.03 OK

Algorithm Order	Old EMB	Old+T EMB	Normal EMB	Old LPO	Old+T LPO	Normal LPO	Type LPO	Emb LPO	Bottom-Up LPO	Combi
3.53	0.29 -	1.49 -	1.49 -	0.54 -	2.26 -	2.32 -	2.44 -	1.59 -	1.32 -	0.29 OK
3.53a	0.01 OK	0.01 OK	0.01 OK	0.02 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK
3.53b	0.03 OK	0.05 OK	0.05 OK	0.05 OK	0.06 OK	0.06 OK	0.07 OK	0.05 OK	0.04 OK	0.04 OK
3.54	0.03 OK	0.05 OK	0.05 OK	0.04 OK	0.05 OK	0.06 OK	0.06 OK	0.05 OK	0.11 -	0.07 OK
3.55	0.52 -	1.69 -	0.87 OK	2.33 OK	2.63 OK	1.93 OK	1.64 OK	1.03 OK	0.69 OK	0.59 OK
3.56	0.05 OK	0.07 OK	0.06 OK	0.06 OK	0.07 OK	0.07 OK	0.06 OK	0.06 OK	0.09 -	0.08 OK
3.57	0.28 -	0.88 -	0.89 -	0.44 -	1.52 OK	1.55 OK	1.55 OK	0.88 OK	0.71 -	0.33 OK
4.1	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK
4.2	0.03 OK	0.04 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK
4.3	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
4.4	0.02 OK	0.01 OK	0.02 OK	0.04 OK	0.01 OK	0.02 OK	0.01 OK	0.02 OK	0.02 OK	0.01 OK
4.4a	0.01 OK	0.02 OK	0.01 OK	0.01 OK	0.01 OK	0.02 OK	0.01 OK	0.02 OK	0.02 OK	0.01 OK
4.5	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
4.6	0.03 -	0.03 OK	0.02 OK	0.06 -	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.01 OK
4.7	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK
4.8	0.07 -	0.05 OK	0.05 OK	0.19 -	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.04 OK
4.9	0.06 -	0.06 OK	0.06 OK	0.14 -	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.06 OK	0.04 OK
4.10	0.02 OK	0.02 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK
4.11	0.03 OK	0.05 OK	0.04 OK	0.03 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.05 OK
4.12	0.04 -	0.03 OK	0.02 OK	0.08 -	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.03 OK
4.12a	0.04 -	0.03 OK	0.02 OK	0.08 -	0.02 OK	0.03 OK	0.02 OK	0.02 OK	0.03 OK	0.02 OK
4.13	0.01 OK	0.02 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.02 OK
4.14	0.03 OK	0.04 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK
4.15	0.05 OK	0.05 OK	0.04 OK	0.03 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.08 -	0.09 OK
4.16	0.03 OK	0.04 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK
4.17	0.06 -	0.05 OK	0.04 OK	0.17 -	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK
4.18	0.03 -	0.03 OK	0.02 OK	0.03 -	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK
4.19	0.07 OK	0.10 OK	0.08 OK	0.12 OK	0.10 OK	0.10 OK	0.09 OK	0.09 OK	0.08 OK	0.09 OK
4.20	0.02 OK	0.03 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK
4.20a	0.03 OK	0.06 OK	0.03 OK	0.04 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK

Algorithm Order	Old EMB	Old+T EMB	Normal EMB	Old LPO	Old+T LPO	Normal LPO	Type LPO	Emb LPO	Bottom-Up LPO	Combi
4.21	0.02 OK	0.03 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK
4.22	0.04 OK	0.05 OK	0.05 OK	0.04 OK	0.05 OK	0.05 OK	0.05 OK	0.05 OK	0.06 OK	0.05 OK
4.23	0.09 -	0.27 -	0.10 OK	0.17 OK	0.14 OK	0.14 OK	0.14 OK	0.11 OK	0.10 OK	0.10 OK
4.25	0.02 -	0.02 OK	0.02 OK	0.03 -	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.01 OK
4.26	0.10 -	0.67 -	0.55 OK	0.44 -	1.28 OK	1.20 OK	0.83 OK	0.64 OK	0.57 OK	0.88 OK
4.27	0.08 -	0.10 OK	0.09 OK	0.10 -	0.09 OK	0.10 OK	0.10 OK	0.10 OK	0.10 OK	0.08 OK
4.28	0.05 -	0.13 OK	0.11 OK	0.06 -	0.13 OK	0.13 OK	0.13 OK	0.11 OK	0.11 OK	0.15 OK
4.29	0.23 -	0.83 -	0.65 OK	3.23 -	4.58 OK	4.24 OK	3.11 OK	0.73 OK	0.81 OK	1.05 OK
4.30	0.21 -	0.81 -	0.70 OK	1.64 -	3.18 OK	3.08 OK	2.01 OK	0.79 OK	0.58 OK	0.72 OK
4.30a	0.09 -	0.12 OK	0.12 OK	0.12 -	0.12 OK	0.13 OK	0.13 OK	0.13 OK	0.11 OK	0.10 OK
4.30b	0.45 -	1.23 -	1.12 OK	3.63 -	5.46 OK	5.06 OK	3.94 OK	1.26 OK	0.62 OK	0.92 OK
4.30c	0.25 -	1.63 -	1.58 -	2.50 -	9.61 -	9.39 -	7.11 -	1.73 -	1.48 -	1.28 OK
4.31	0.09 -	0.23 OK	0.20 OK	1.52 -	1.59 OK	1.61 OK	1.61 OK	0.22 OK	0.16 OK	0.36 OK
4.32	0.05 OK	0.06 OK	0.06 OK	0.06 OK	0.07 OK	0.07 OK	0.07 OK	0.06 OK	0.06 OK	0.05 OK
4.33	0.12 OK	0.15 OK	0.15 OK	0.13 OK	0.16 OK	0.16 OK	0.16 OK	0.17 OK	0.15 OK	0.10 OK
4.34	0.16 -	0.58 -	0.57 -	0.58 OK	0.36 OK	0.28 OK	0.26 OK	0.76 -	0.42 -	0.36 OK
4.35	3.75 -	30.00 ∞	5.50 OK	30.00 ∞	30.00 ∞	30.00 ∞	28.20 OK	3.71 OK	15.24 -	3.57 OK
4.36	1.71 -	3.74 -	2.06 OK	30.00 ∞	30.00 ∞	5.62 OK	3.02 OK	2.26 OK	0.71 OK	0.54 OK
4.37	0.05 OK	0.06 OK	0.06 OK	0.08 OK	0.07 OK	0.07 OK	0.06 OK	0.06 OK	0.09 -	0.11 OK
4.37a	0.05 OK	0.07 OK	0.07 OK	0.06 OK	0.07 OK	0.07 OK	0.07 OK	0.07 OK	0.11 -	0.09 OK
D.1	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.03 OK	0.02 OK
D.2	0.02 OK	0.03 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK
D.3	0.01 OK	0.02 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
D.6	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK	0.03 OK	0.01 OK
D.7	0.05 OK	0.06 OK	0.06 OK	0.05 OK	0.06 OK	0.06 OK	0.07 OK	0.06 OK	0.09 -	0.07 OK
D.8	0.09 OK	0.10 OK	0.09 OK	0.09 OK	0.10 OK	0.11 OK	0.10 OK	0.10 OK	0.12 OK	0.09 OK
D.9	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.07 -	0.04 OK
D.11	0.38 OK	0.42 OK	0.41 OK	0.40 OK	0.41 OK	0.46 OK	0.45 OK	0.46 OK	0.77 OK	0.42 OK
D.12	0.09 -	0.11 -	0.10 -	0.10 -	0.11 -	0.12 -	0.11 -	0.11 -	0.14 -	0.15 OK
D.13	0.12 -	0.58 -	0.58 -	0.12 -	0.59 -	0.63 -	0.62 -	0.62 -	0.16 -	0.74 -

Algorithm Order	Old EMB	Old+T EMB	Normal EMB	Old LPO	Old+T LPO	Normal LPO	Type LPO	Emb LPO	Bottom-Up LPO	Combi
D.17	0.07 OK	0.08 OK	0.08 OK	0.08 OK	0.08 OK	0.09 OK	0.08 OK	0.09 OK	0.09 OK	0.09 OK
D.18	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK
D.20	0.23 -	30.00 ∞	30.00 ∞	0.27 OK	1.43 OK	1.54 OK	1.53 OK	1.45 OK	1.45 OK	1.31 OK
D.21	0.07 -	0.09 OK	0.09 OK	0.08 -	0.09 OK	0.09 OK	0.09 OK	0.09 OK	0.10 OK	0.07 OK
D.28	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.05 OK	0.05 OK	0.04 OK	0.05 OK	0.05 -	0.06 OK
D.29	0.06 -	0.44 -	0.10 OK	0.06 OK	0.09 OK	0.10 OK	0.10 OK	0.10 OK	0.09 OK	0.08 OK
D.30	0.08 -	5.51 -	5.52 -	0.23 OK	0.40 OK	0.41 OK	0.41 OK	0.25 OK	5.68 -	0.26 OK
D.32	0.16 OK	0.19 OK	0.20 OK	0.20 OK	0.23 OK	0.23 OK	0.21 OK	0.21 OK	0.17 OK	0.12 OK
D.33	0.17 -	7.59 -	7.45 -	0.40 -	30.00 ∞	30.00 ∞	30.00 ∞	7.93 -	0.17 -	8.36 -
S.1	0.01 OK	0.01 OK	0.01 OK	0.02 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
S.2	0.03 -	0.04 OK	0.04 OK	0.05 -	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.03 OK
S.3	0.06 OK	0.07 OK	0.07 OK	0.07 OK	0.07 OK	0.07 OK	0.08 OK	0.07 OK	0.07 OK	0.05 OK
S.4	0.01 OK	0.01 OK	0.02 OK	0.02 OK	0.01 OK	0.02 OK	0.01 OK	0.02 OK	0.01 OK	0.01 OK
S.5	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.03 OK	0.02 OK
S.6	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
S.7	0.04 OK	0.05 OK	0.04 OK	0.04 OK	0.05 OK	0.06 OK	0.04 OK	0.05 OK	0.03 OK	0.02 OK
S.10	0.03 -	0.03 OK	0.03 OK	0.03 -	0.04 OK	0.04 OK	0.03 OK	0.04 OK	0.04 OK	0.03 OK
S.11	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
S.12	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.02 OK	0.01 OK
S.14	0.13 OK	0.15 OK	0.14 OK	0.14 OK	0.15 OK	0.16 OK	0.16 OK	0.16 OK	0.16 -	0.22 OK
S.15	0.05 -	0.11 -	0.10 -	0.05 -	0.12 -	0.12 -	0.12 -	0.10 -	0.11 -	0.16 -
S.17	0.10 -	0.47 -	0.47 -	0.15 -	0.62 -	0.64 -	0.57 -	0.51 -	0.42 -	0.21 OK
S.18	0.05 -	0.06 -	0.06 -	0.07 -	0.08 -	0.08 -	0.08 -	0.07 -	0.09 -	0.14 OK
S.22	0.07 -	0.14 OK	0.13 OK	0.09 -	0.15 OK	0.16 OK	0.16 OK	0.14 OK	0.14 OK	0.16 OK
S.24	0.20 -	0.65 -	0.16 OK	3.17 -	7.49 -	2.35 OK	1.79 OK	0.18 OK	0.15 OK	0.11 OK
S.25	0.03 OK	0.04 OK	0.04 OK	0.05 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.04 OK	0.03 OK
S.26	0.44 -	0.89 -	0.42 OK	2.07 -	3.28 -	0.88 OK	0.88 OK	0.47 OK	0.17 OK	0.11 OK
S.27	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.02 OK	0.01 OK	0.01 OK	0.01 OK	0.01 OK	0.00 OK
S.28	0.06 -	0.16 -	0.16 -	0.14 -	0.30 -	0.31 -	0.31 -	0.17 -	0.17 -	0.15 OK
S.29	0.07 OK	0.08 OK	0.08 OK	0.09 OK	0.08 OK	0.09 OK	0.08 OK	0.09 OK	0.08 OK	0.07 OK

Algorithm Order	Old EMB	Old+T EMB	Normal EMB	Old LPO	Old+T LPO	Normal LPO	Type LPO	Emb LPO	Bottom-Up LPO	Combi
S.30	0.23 -	0.10 OK	0.09 OK	0.33 OK	0.09 OK	0.10 OK	0.10 OK	0.10 OK	0.10 OK	0.08 OK
S.31	0.16 -	1.55 OK	1.56 OK	0.48 -	2.83 OK	2.77 OK	2.02 OK	1.71 OK	0.61 -	1.07 OK
Total: 151	27.94 77	155.06 101	131.36 121	205.12 98	334.93 128	298.76 131	275.06 132	102.48 128	95.96 113	34.82 148

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

- 95-11 * M. Staudt / K. von Thadden: Subsumption Checking in Knowledge Bases
- 95-12 * G.V. Zemanek / H.W. Nissen / H. Hubert / M. Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology
- 95-13 * M. Staudt / M. Jarke: Incremental Maintenance of Externally Materialized Views
- 95-14 * P. Peters / P. Szczurko / M. Jeusfeld: Business Process Oriented Information Management: Conceptual Models at Work
- 95-15 * S. Rams / M. Jarke: Proceedings of the Fifth Annual Workshop on Information Technologies & Systems
- 95-16 * W. Hans / St. Winkler / F. Sáenz: Distributed Execution in Functional Logic Programming
- 96-1 * Jahresbericht 1995
- 96-2 M. Hanus / Chr. Prehofer: Higher-Order Narrowing with Definitional Trees
- 96-3 * W. Scheufele / G. Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates
- 96-4 K. Pohl: PRO-ART: Enabling Requirements Pre-Traceability
- 96-5 K. Pohl: Requirements Engineering: An Overview
- 96-6 * M. Jarke / W. Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools
- 96-7 O. Chitil: The ς-Semantics: A Comprehensive Semantics for Functional Programs
- 96-8 * S. Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics
- 96-9 M. Hanus (Ed.): Proceedings of the Poster Session of ALP'96 — Fifth International Conference on Algebraic and Logic Programming
- 96-10 R. Conradi / B. Westfechtel: Version Models for Software Configuration Management
- 96-11 * C. Weise / D. Lenzkes: A Fast Decision Algorithm for Timed Refinement
- 96-12 * R. Dömges / K. Pohl / M. Jarke / B. Lohmann / W. Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models
- 96-13 * K. Pohl / R. Klamma / K. Weidenhaupt / R. Dömges / P. Haumer / M. Jarke: A Framework for Process-Integrated Tools
- 96-14 * R. Gallersdörfer / K. Klabunde / A. Stolz / M. Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996
- 96-15 * H. Schimpe / M. Staudt: VAREX: An Environment for Validating and Refining Rule Bases

96-16 * M. Jarke / M. Gebhardt, S. Jacobs, H. Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization

96-17 M. Jeusfeld / T. X. Bui: Decision Support Components on the Internet

96-18 M. Jeusfeld / M. Papazoglou: Information Brokering: Design, Search and Transformation

96-19 * P. Peters / M. Jarke: Simulating the impact of information flows in networked organizations

96-20 M. Jarke / P. Peters / M. Jeusfeld: Model-driven planning and design of cooperative information systems

96-21 * G. de Michelis / E. Dubois / M. Jarke / F. Matthes / J. Mylopoulos / K. Pohl / J. Schmidt / C. Woo / E. Yu: Cooperative information systems: a manifesto

96-22 * S. Jacobs / M. Gebhardt, S. Kethers, W. Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality

96-23 * M. Gebhardt / S. Jacobs: Conflict Management in Design

97-01 Jahresbericht 1996

97-02 J. Faassen: Using full parallel Boltzmann Machines for Optimization

97-03 A. Winter / A. Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems

97-04 M. Mohnen / S. Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler

97-05 * S. Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge

97-06 M. Nicola / M. Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries

97-07 P. Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen

97-08 D. Blostein / A. Schürr: Computing with Graphs and Graph Rewriting

97-09 C.-A. Krapp / B. Westfechtel: Feedback Handling in Dynamic Task Nets

97-10 M. Nicola / M. Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases

97-13 M. Mohnen: Optimising the Memory Management of Higher-Order Functional Programs

97-14 R. Baumann: Client/Server Distribution in a Structure-Oriented Database Management System

97-15 G. H. Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms

98-01 * Jahresbericht 1997

98-02 S. Gruner/ M. Nagel / A. Schürr: Fine-grained and Structure-oriented Integration Tools are Needed for Product Development Processes

98-03 S. Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

98-04 * O. Kubitz: Mobile Robots in Dynamic Environments

98-05 M. Leucker / St. Tobies: Truth — A Verification Platform for Distributed Systems

98-07 M. Arnold / M. Erdmann / M. Glinz / P. Haumer / R. Knoll / B. Paech / K. Pohl / J. Ryser / R. Studer / K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects

98-08 * H. Aust: Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen

98-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien

98-10 * M. Nicola / M. Jarke: Performance Modeling of Distributed and Replicated Databases

98-11 * A. Schleicher / B. Westfechtel / D. Jäger: Modeling Dynamic Software Processes in UML

98-12 * W. Appelt / M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web

98-13 K. Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation

99-01 * Jahresbericht 1998

99-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version

99-03 * R. Gallersdörfer / M. Jarke / M. Nicola: The ADR Replication Manager

99-04 M. Alpuente / M. Hanus / S. Lucas / G. Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing

99-07 Th. Wilke: CTL+ is exponentially more succinct than CTL

99-08 O. Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures

2000-01 * Jahresbericht 1999

2000-02 Jens Vöge / Marcin Jurdzinski: A Discrete Strategy Improvement Algorithm for Solving Parity Games

2000-04 Andreas Becks / Stefan Sklorz / Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach

2000-05 Mareike Schoop: Cooperative Document Management

2000-06 Mareike Schoop / Christoph Quix (eds.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling

2000-07 * Markus Mohnen / Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages

2000-08 Thomas Arts / Thomas Noll: Verifying Generic Erlang Client-Server Implementations

2001-01 * Jahresbericht 2000

2001-02 Benedikt Bollig / Martin Leucker: Deciding LTL over Mazurkiewicz Traces

2001-03 Thierry Cachat: The power of one-letter rational languages

2001-04 Benedikt Bollig / Martin Leucker / Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus

2001-05 Benedikt Bollig / Martin Leucker / Thomas Noll: Regular MSC Languages

2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic

2001-07 Martin Grohe / Stefan Wöhrle: An Existential Locality Theorem

2001-08 Mareike Schoop / James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling

2001-09 Thomas Arts / Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs

2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures

2001-11 Klaus Indermark / Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung

2002-01 * Jahresbericht 2001

2002-02 Jürgen Giesl / Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems

2002-03 Benedikt Bollig / Martin Leucker / Thomas Noll: Generalised Regular MSC Languages

2002-04 Jürgen Giesl / Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting

2002-05 Horst Lichter / Thomas von der Maßen / Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines

2002-06 Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata

2002-07 Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities

2002-08 Markus Mohnen: An Open Framework for Data-Flow Analysis in Java

2002-09 Markus Mohnen: Interfaces with Default Implementations in Java

2002-10 Martin Leucker: Logics for Mazurkiewicz traces

2002-11 Jürgen Giesl / Hans Zantema: Liveness in Rewriting

2003-01 * Jahresbericht 2002

2003-02 Jürgen Giesl / René Thiemann: Size-Change Termination for Term Rewriting

2003-03 Jürgen Giesl / Deepak Kapur: Deciding Inductive Validity of Equations

2003-04 Jürgen Giesl / René Thiemann / Peter Schneider-Kamp / Stephan Falke: Improving Dependency Pairs

2003-05 Christof Löding / Philipp Rohde: Solving the Sabotage Game is PSPACE-hard

2003-06 Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates

2003-07 Horst Lichter / Thomas von der Maßen / Alexander Nyßen / Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.