The Dependency Pair Framework for Automated Complexity Analysis of Term Rewrite Systems*

Lars Noschinski[1], Fabian Emmes[2], and Jürgen Giesl[2]

[1] Institut für Informatik, TU Munich, Germany
[2] LuFG Informatik 2, RWTH Aachen University, Germany

Abstract. We present a modular framework to analyze the innermost runtime complexity of term rewrite systems (TRSs) automatically. Our method is based on the dependency pair (DP) framework for termination analysis. In contrast to previous work, we developed a *direct* adaptation of successful termination techniques from the DP framework in order to use them for complexity analysis. By extensive experimental results, we demonstrate the power of our method compared to existing techniques.

1 Introduction

In practice, termination is often not sufficient, but one also has to ensure that algorithms terminate in *reasonable* (e.g., polynomial) *time*. While termination of TRSs is well studied, only recently first ground-breaking results were obtained which adapt termination techniques in order to obtain polynomial complexity bounds automatically, e.g., [2–5,7,9,13,14,16–18,20–22]. Here, [3,13,14] consider the *DP method* [1, 10–12], which is one of the most popular termination techniques for TRSs.[3] Moreover, [22] introduces a modular approach for complexity analysis based on relative rewriting, which has similarities to the DP method.

In this paper, we present a fresh adaptation of the DP framework for *innermost runtime complexity analysis* [13]. In contrast to [3, 13, 14], we follow the original DP framework closely. This allows us to directly adapt the several termination techniques ("processors") of the DP framework for complexity analysis. Like [22], our method is modular. But in contrast to [22], which allows to investigate *derivational complexity* [15] we focus on innermost runtime complexity. Hence, we can inherit the modularity aspects of the DP framework and benefit from its transformation techniques, which increases power significantly.

After introducing preliminaries in Sect. 2, in Sect. 3 we adapt the concept of *dependency pairs* from termination analysis to so-called *dependency tuples* for complexity analysis. While the *DP framework* for termination works on *DP problems*, we now work on *DT problems* (Sect. 4). Sect. 5 adapts the "processors" of the DP framework in order to analyze the complexity of DT problems. We implemented our contributions in the termination analyzer AProVE. Due to the results of this paper, in the *International Termination Competition* 2010,

* Supported by the DFG grant GI 274/5-3.

[3] In addition, there are also several approaches to characterize complexity classes using termination techniques like dependency pairs (e.g., [17]).

AProVE was the most powerful tool for innermost runtime complexity analysis. This is confirmed by our experiments in Sect. 6, where we compare our technique with previous approaches. All proofs can be found in the appendix.

2 Runtime Complexity of Term Rewriting

See e.g. [6] for the basics of term rewriting. Let $\mathcal{T}(\Sigma, \mathcal{V})$ be the set of all terms over a signature Σ and a set of variables $\mathcal{V}$ where we just write $\mathcal{T}$ if Σ and $\mathcal{V}$ are clear from the context. The *arity* of a function symbol $f \in \Sigma$ is denoted by $\mathrm{ar}(f)$ and the size of a term is $|x| = 1$ for $x \in \mathcal{V}$ and $|f(t_1, \ldots, t_n)| = 1 + |t_1| + \ldots + |t_n|$. The *derivation length* of a term t w.r.t. a relation $\to$ is the length of the longest sequence of $\to$-steps starting with t, i.e., $\mathrm{dl}(t, \to) = \sup\{\, n \mid \exists t' \in \mathcal{T},\, t \to^n t' \,\}$, cf. [15]. Here, for any set $M \subseteq \mathbb{N} \cup \{\infty\}$, "$\sup M$" is the least upper bound of M. Thus, $\mathrm{dl}(t, \to) = \infty$ if t starts an infinite sequence of $\to$-steps.

As an example, consider $\mathcal{R} = \{\mathsf{dbl}(\mathsf{0}) \to \mathsf{0},\ \mathsf{dbl}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{dbl}(x)))\}$. Then $\mathrm{dl}(\mathsf{dbl}(\mathsf{s}^n(\mathsf{0})), \to_\mathcal{R}) = n + 1$, but $\mathrm{dl}(\mathsf{dbl}^n(\mathsf{s}(\mathsf{0})), \to_\mathcal{R}) = 2^n + n - 1$.

For a TRS $\mathcal{R}$ with *defined symbols* $\Sigma_d = \{\, \mathrm{root}(\ell) \mid \ell \to r \in \mathcal{R} \,\}$, a term $f(t_1, \ldots, t_n)$ is *basic* if $f \in \Sigma_d$ and $t_1, \ldots, t_n$ do not contain symbols from Σ_d. So for $\mathcal{R}$ above, the basic terms are $\mathsf{dbl}(\mathsf{s}^n(\mathsf{0}))$ and $\mathsf{dbl}(\mathsf{s}^n(x))$ for $n \in \mathbb{N}$, $x \in \mathcal{V}$. The *(innermost) runtime complexity function* $\mathrm{rc}_\mathcal{R}$ maps any $n \in \mathbb{N}$ to the length of the longest sequence of $\xrightarrow{\mathrm{i}}_\mathcal{R}$-steps starting with a basic term t with $|t| \leq n$. Here, "$\xrightarrow{\mathrm{i}}_\mathcal{R}$" is the innermost rewrite relation and $\mathcal{T}_B$ is the set of all basic terms.

Definition 1 (Runtime Complexity [13]). *For a TRS $\mathcal{R}$, its* runtime complexity function $\mathrm{rc}_\mathcal{R} : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ *is* $\mathrm{rc}_\mathcal{R}(n) = \sup\{\, \mathrm{dl}(t, \xrightarrow{\mathrm{i}}_\mathcal{R}) \mid t \in \mathcal{T}_B, |t| \leq n \,\}$.

If one only considers evaluations of basic terms, the runtime complexity of the dbl-TRS is linear ($\mathrm{rc}_\mathcal{R}(n) = n - 1$ for $n \geq 2$). But if one also permits evaluations starting with terms like $\mathsf{dbl}^n(\mathsf{s}(\mathsf{0}))$, the complexity of the dbl-TRS is exponential.

When analyzing the complexity of *programs*, one is typically only interested in (innermost) evaluations where a defined function like dbl is applied to data objects (i.e., to terms without defined symbols). Therefore, *runtime complexity* corresponds to the usual notion of "complexity" for programs [4, 5]. So for any TRS $\mathcal{R}$, our goal is to determine the *asymptotic complexity* of the function $\mathrm{rc}_\mathcal{R}$.

Definition 2 (Asymptotic Complexities). *Let $\mathfrak{C} = \{\mathcal{P}ol_0, \mathcal{P}ol_1, \mathcal{P}ol_2, ..., \infty\}$ with the order $\mathcal{P}ol_0 \sqsubset \mathcal{P}ol_1 \sqsubset \mathcal{P}ol_2 \sqsubset \ldots \sqsubset \infty$. Let $\sqsubseteq$ be the reflexive closure of $\sqsubset$. For any function $f : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ we define its* complexity $\iota(f) \in \mathfrak{C}$ *as follows: $\iota(f) = \mathcal{P}ol_k$ if k is the smallest number with $f(n) \in \mathcal{O}(n^k)$ and $\iota(f) = \infty$ if there is no such k. For any TRS $\mathcal{R}$, we define its* complexity $\iota_\mathcal{R}$ *as* $\iota(\mathrm{rc}_\mathcal{R})$.

So the dbl-TRS $\mathcal{R}$ has linear complexity, i.e., $\iota_\mathcal{R} = \mathcal{P}ol_1$. As another example, consider the following TRS $\mathcal{R}$ where "m" stands for "minus".

Example 3.

$$
\begin{array}{lll}
\mathsf{m}(x, y) \to \mathsf{if}(\mathsf{gt}(x, y), x, y) & \mathsf{gt}(\mathsf{0}, k) \to \mathsf{false} & \mathsf{p}(\mathsf{0}) \to \mathsf{0} \\
\mathsf{if}(\mathsf{true}, x, y) \to \mathsf{s}(\mathsf{m}(\mathsf{p}(x), y)) & \mathsf{gt}(\mathsf{s}(n), \mathsf{0}) \to \mathsf{true} & \mathsf{p}(\mathsf{s}(n)) \to n \\
\mathsf{if}(\mathsf{false}, x, y) \to \mathsf{0} & \mathsf{gt}(\mathsf{s}(n), \mathsf{s}(k)) \to \mathsf{gt}(n, k) &
\end{array}
$$

The terms $\mathsf{m}(\mathsf{s}^n(\mathsf{0}), \mathsf{s}^k(\mathsf{0}))$ start evaluations of quadratic length. So $\iota_\mathcal{R} = \mathcal{P}ol_2$.

3 Dependency Tuples

In the DP method, for every $f \in \Sigma_d$ one introduces a fresh symbol $f^\sharp$ with $\mathrm{ar}(f) = \mathrm{ar}(f^\sharp)$. For a term $t = f(t_1, \ldots, t_n)$ with $f \in \Sigma_d$ we define $t^\sharp = f^\sharp(t_1, \ldots, t_n)$ and let $\mathcal{T}^\sharp = \{\, t^\sharp \mid t \in \mathcal{T}, \mathrm{root}(t) \in \Sigma_d \,\}$. Let $\mathcal{P}os(t)$ contain all positions of t and let $\mathcal{P}os_d(t) = \{\, \pi \mid \pi \in \mathcal{P}os(t), \mathrm{root}(t|_\pi) \in \Sigma_d \,\}$. Then for every rule $\ell \to r$ with $\mathcal{P}os_d(r) = \{\pi_1, \ldots, \pi_n\}$, its *dependency pairs* are $\ell^\sharp \to r|^\sharp_{\pi_1}, \ldots, \ell^\sharp \to r|^\sharp_{\pi_n}$.

While DPs are useful for termination, for complexity we have to regard all defined functions in a right-hand side *at once*. Thus, we extend the concept of *weak dependency pairs* [13, 14] and only build a single *dependency tuple* $\ell \to [r|^\sharp_{\pi_1}, \ldots, r|^\sharp_{\pi_n}]$ for each $\ell \to r$. To avoid handling tuples, for every $n \geq 0$, we introduce a fresh *compound symbol* Com_n of arity n and use $\ell \to \mathrm{Com}_n(r|^\sharp_{\pi_1}, \ldots, r|^\sharp_{\pi_n})$.

Definition 4 (Dependency Tuple). *A* dependency tuple *is a rule of the form $s^\sharp \to \mathrm{Com}_n(t^\sharp_1, \ldots, t^\sharp_n)$ for $s^\sharp, t^\sharp_1, \ldots, t^\sharp_n \in \mathcal{T}^\sharp$. Let $\ell \to r$ be a rule with $\mathcal{P}os_d(r) = \{\pi_1, \ldots, \pi_n\}$. Then $DT(\ell \to r)$ is defined[4] to be $\ell^\sharp \to \mathrm{Com}_n(r|^\sharp_{\pi_1}, \ldots, r|^\sharp_{\pi_n})$. For a TRS $\mathcal{R}$, let $DT(\mathcal{R}) = \{DT(\ell \to r) \mid \ell \to r \in \mathcal{R}\}$.*

Example 5. For the TRS $\mathcal{R}$ from Ex. 3, $DT(\mathcal{R})$ is the following set of rules.

$$\mathsf{m}^\sharp(x, y) \to \mathrm{Com}_2(\mathsf{if}^\sharp(\mathsf{gt}(x, y), x, y), \mathsf{gt}^\sharp(x, y)) \quad (1)$$

$$\mathsf{if}^\sharp(\mathsf{true}, x, y) \to \mathrm{Com}_2(\mathsf{m}^\sharp(\mathsf{p}(x), y), \mathsf{p}^\sharp(x)) \quad (2)$$

$$\mathsf{if}^\sharp(\mathsf{false}, x, y) \to \mathrm{Com}_0 \quad (3)$$

$$\mathsf{p}^\sharp(\mathsf{0}) \to \mathrm{Com}_0 \quad (4)$$

$$\mathsf{p}^\sharp(\mathsf{s}(n)) \to \mathrm{Com}_0 \quad (5)$$

$$\mathsf{gt}^\sharp(\mathsf{0}, k) \to \mathrm{Com}_0 \quad (6)$$

$$\mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{0}) \to \mathrm{Com}_0 \quad (7)$$

$$\mathsf{gt}^\sharp(\mathsf{s}(n), \mathsf{s}(k)) \to \mathrm{Com}_1(\mathsf{gt}^\sharp(n, k)) \quad (8)$$

For termination, one analyzes *chains* of DPs, which correspond to sequences of function calls that can occur in reductions. Since DTs represent *several* DPs, we now obtain *chain trees*. (This is analogous to the *path detection* in [14]).

Definition 6 (Chain Tree). *Let $\mathcal{D}$ be a set of DTs and $\mathcal{R}$ be a TRS. Let T be a (possibly infinite) tree whose nodes are labeled with both a DT from $\mathcal{D}$ and a substitution. Let the root node be labeled with $(s^\sharp \to \mathrm{Com}_n(\ldots) \mid \sigma)$. Then T is a $(\mathcal{D}, \mathcal{R})$-*chain tree *for $s^\sharp\sigma$ if the following holds for all nodes of T: If a node is labeled with $(u^\sharp \to \mathrm{Com}_m(v^\sharp_1, \ldots, v^\sharp_m) \mid \mu)$, then $u^\sharp\mu$ is in normal form w.r.t. $\mathcal{R}$. Moreover, if this node has the children $(p^\sharp_1 \to \mathrm{Com}_{m_1}(\ldots) \mid \tau_1), \ldots, (p^\sharp_k \to \mathrm{Com}_{m_k}(\ldots) \mid \tau_k)$, then there are pairwise different $i_1, \ldots, i_k \in \{1, \ldots, m\}$ with $v^\sharp_{i_j}\mu \xrightarrow{\mathrm{i}}^*_{\mathcal{R}} p^\sharp_j\tau_j$ for all $j \in \{1, \ldots, k\}$. A path in the chain tree is called a* chain.[5]

Example 7. For the TRS $\mathcal{R}$ from Ex. 3 and its DTs from Ex. 5, the tree in Fig. 1 is a $(DT(\mathcal{R}), \mathcal{R})$-chain tree for $\mathsf{m}^\sharp(\mathsf{s}(\mathsf{0}), \mathsf{0})$. Here, we use substitutions with $\sigma(x) = \mathsf{s}(\mathsf{0})$ and $\sigma(y) = \mathsf{0}$, $\tau(x) = \tau(y) = \mathsf{0}$, and $\mu(n) = \mu(k) = \mathsf{0}$.

For any term $s^\sharp \in \mathcal{T}^\sharp$, we define its *complexity* as the maximal number of nodes in any chain tree for $s^\sharp$. However, sometimes we do not want to count *all* DTs in the chain tree, but only the DTs from some subset $\mathcal{S}$. This will be crucial to adapt termination techniques for complexity, cf. Sect. 5.2 and 5.4.

[4] To make $DT(\ell \to r)$ unique, we use a total order $<$ on positions where $\pi_1 < \ldots < \pi_n$.

[5] These *chains* correspond to the "*innermost chains*" in the DP framework [1, 10, 11].

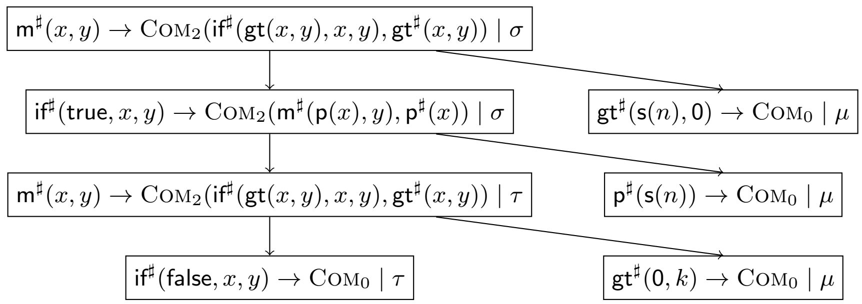


Fig. 1. Chain Tree for the TRS from Ex. 3

Definition 8 (Complexity of Terms, $\mathcal{C}plx_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$). *Let $\mathcal{D}$ be a set of dependency tuples, $\mathcal{S} \subseteq \mathcal{D}$, $\mathcal{R}$ a TRS, and $s^\sharp \in \mathcal{T}^\sharp$. Then $\mathcal{C}plx_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}(s^\sharp) \in \mathbb{N} \cup \{\infty\}$ is the maximal number of nodes from $\mathcal{S}$ occurring in any $(\mathcal{D},\mathcal{R})$-chain tree for $s^\sharp$. If there is no $(\mathcal{D},\mathcal{R})$-chain tree for $s^\sharp$, then $\mathcal{C}plx_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}(s^\sharp) = 0$.*

Example 9. For $\mathcal{R}$ from Ex. 3, we have $\mathcal{C}plx_{\langle DT(\mathcal{R}),DT(\mathcal{R}),\mathcal{R}\rangle}(\mathsf{m}^\sharp(\mathsf{s}(\mathsf{0}),\mathsf{0})) = 7$, since the maximal tree for $\mathsf{m}^\sharp(\mathsf{s}(\mathsf{0}),\mathsf{0})$ in Fig. 1 has 7 nodes. In contrast, if $\mathcal{S}$ is $DT(\mathcal{R})$ without the $\mathsf{gt}^\sharp$-DTs (6) – (8), then $\mathcal{C}plx_{\langle DT(\mathcal{R}),\mathcal{S},\mathcal{R}\rangle}(\mathsf{m}^\sharp(\mathsf{s}(\mathsf{0}),\mathsf{0})) = 5$.

Thm. 10 shows how dependency tuples can be used to approximate the derivation lengths of terms. More precisely, $\mathcal{C}plx_{\langle DT(\mathcal{R}),DT(\mathcal{R}),\mathcal{R}\rangle}(t^\sharp)$ is an upper bound for t's derivation length, provided that t is in *argument normal form*.

Theorem 10 ($\mathcal{C}plx$ bounds Derivation Length). *Let $\mathcal{R}$ be a TRS. Let $t = f(t_1,\ldots,t_n) \in \mathcal{T}$ be in* argument normal form*, i.e., all t_i are normal forms w.r.t. $\mathcal{R}$. Then we have $\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \le \mathcal{C}plx_{\langle DT(\mathcal{R}),DT(\mathcal{R}),\mathcal{R}\rangle}(t^\sharp)$. If $\mathcal{R}$ is confluent, we even have $\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = \mathcal{C}plx_{\langle DT(\mathcal{R}),DT(\mathcal{R}),\mathcal{R}\rangle}(t^\sharp)$.*

Note that DTs are much closer to the original DP method than the *weak DPs* of [13,14] which only consider the *topmost* defined function symbols in right-hand sides of rules. Hence, [13, 14] does not use DP concepts when defined functions occur nested on right-hand sides (as in the m- and the first if-rule) and thus, it cannot fully benefit from the advantages of the DP technique. Instead, [13, 14] has to impose several restrictions which are not needed in our approach, cf. Footnote 9. The close analogy of our approach to the DP method allows us to adapt the termination techniques of the DP framework in order to work on DTs (i.e., in order to analyze $\mathcal{C}plx_{\langle DT(\mathcal{R}),DT(\mathcal{R}),\mathcal{R}\rangle}(t^\sharp)$ for all base terms t of a certain size). By Thm. 10, this yields an upper bound for the complexity $\iota_{\mathcal{R}}$ of the TRS $\mathcal{R}$. Note that there exist non-confluent TRSs[6] where $\mathcal{C}plx_{\langle DT(\mathcal{R}),DT(\mathcal{R}),\mathcal{R}\rangle}(t^\sharp)$ is exponentially larger than $\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}})$ (this is in contrast to [13, 14], where the step from TRSs to weak DPs does not change the complexity). However, our main interest is in TRSs corresponding to "typical" (confluent) *programs*. Here, the step from TRSs to DTs does not "lose" anything (i.e., one has equality in Thm. 10).

[6] Consider the TRS $\mathsf{f}(\mathsf{s}(x)) \to \mathsf{f}(\mathsf{g}(x))$, $\mathsf{g}(x) \to x$, $\mathsf{g}(x) \to \mathsf{a}(\mathsf{f}(x))$. Its runtime complexity is linear, but for any $n > 0$, we have $\mathcal{C}plx_{\langle DT(\mathcal{R}),DT(\mathcal{R}),\mathcal{R}\rangle}(\mathsf{f}^\sharp(\mathsf{s}^n(\mathsf{0}))) = 2^{n+1} - 2$.

4 DT Problems

Our goal is to find out automatically how large $\mathcal{C}plx_{\langle\mathcal{D},\mathcal{S},\mathcal{R}\rangle}(t^\sharp)$ could be for basic terms t of size n. To this end, we will repeatedly replace the triple $\langle\mathcal{D},\mathcal{S},\mathcal{R}\rangle$ by "simpler" triples $\langle\mathcal{D}',\mathcal{S}',\mathcal{R}'\rangle$ and examine $\mathcal{C}plx_{\langle\mathcal{D}',\mathcal{S}',\mathcal{R}'\rangle}(t^\sharp)$ instead.

This is similar to the DP framework where termination problems are represented by so-called DP problems (consisting of a set of DPs and a set of rules) and where DP problems are transformed into "simpler" DP problems repeatedly. For complexity analysis, we consider "DT problems" instead of "DP problems".

Definition 11 (DT Problem). *Let $\mathcal{R}$ be a TRS, $\mathcal{D}$ a set of DTs, $\mathcal{S} \subseteq \mathcal{D}$. Then $\langle\mathcal{D},\mathcal{S},\mathcal{R}\rangle$ is a* DT problem *and $\mathcal{R}$'s* canonical DT problem *is $\langle DT(\mathcal{R}),DT(\mathcal{R}),\mathcal{R}\rangle$.*

Thm. 10 showed the connection between the derivation length of a term and the maximal number of nodes in a chain tree. This leads to the definition of the *complexity of a DT problem* $\langle\mathcal{D},\mathcal{S},\mathcal{R}\rangle$. It is defined as the asymptotic complexity of the function $\mathrm{rc}_{\langle\mathcal{D},\mathcal{S},\mathcal{R}\rangle}$ which maps any number n to the maximal number of $\mathcal{S}$-nodes in any $(\mathcal{D},\mathcal{R})$-chain tree for $t^\sharp$, where t is a basic term of at most size n.

Definition 12 (Complexity of DT Problems). *For a DT problem $\langle\mathcal{D},\mathcal{S},\mathcal{R}\rangle$, its* complexity function *is $\mathrm{rc}_{\langle\mathcal{D},\mathcal{S},\mathcal{R}\rangle}(n) = \sup\{\, \mathcal{C}plx_{\langle\mathcal{D},\mathcal{S},\mathcal{R}\rangle}(t^\sharp) \mid t \in \mathcal{T}_B, |t| \leq n\,\}$. We define the* complexity *$\iota_{\langle\mathcal{D},\mathcal{S},\mathcal{R}\rangle}$ of the DT problem as $\iota(\mathrm{rc}_{\langle\mathcal{D},\mathcal{S},\mathcal{R}\rangle})$.*

Example 13. Consider the TRS $\mathcal{R}$ of Ex. 3 and let $\mathcal{D} = DT(\mathcal{R}) = \{(1),\ldots,(8)\}$. For $t \in \mathcal{T}_B$ with $|t| = n$, the maximal chain tree for $t^\sharp$ has approximately n^2 nodes, i.e., $\mathrm{rc}_{\langle\mathcal{D},\mathcal{D},\mathcal{R}\rangle}(n) \in \mathcal{O}(n^2)$. Thus, $\langle\mathcal{D},\mathcal{D},\mathcal{R}\rangle$'s complexity is $\iota_{\langle\mathcal{D},\mathcal{D},\mathcal{R}\rangle} = \mathcal{P}ol_2$.

Thm. 14 shows that to analyze the complexity of a TRS $\mathcal{R}$, it suffices to analyze the complexity of its canonical DT problem: By Def. 2, $\iota_\mathcal{R}$ is the complexity of the runtime complexity function $\mathrm{rc}_\mathcal{R}$ which maps n to the length of the longest innermost rewrite sequence starting with a basic term of at most size n. By Thm. 10, this length is less than or equal to the size $\mathcal{C}plx_{\langle DT(\mathcal{R}),DT(\mathcal{R}),\mathcal{R}\rangle}(t^\sharp)$ of the maximal chain tree for any basic term t of at most size n, i.e., to $\mathrm{rc}_{\langle DT(\mathcal{R}),DT(\mathcal{R}),\mathcal{R}\rangle}(n)$.

Theorem 14 (Upper bound for TRSs via Canonical DT Problems). *Let $\mathcal{R}$ be a TRS and let $\langle\mathcal{D},\mathcal{D},\mathcal{R}\rangle$ be the corresponding canonical DT problem. Then we have $\iota_\mathcal{R} \sqsubseteq \iota_{\langle\mathcal{D},\mathcal{D},\mathcal{R}\rangle}$ and if $\mathcal{R}$ is confluent, we even have $\iota_\mathcal{R} = \iota_{\langle\mathcal{D},\mathcal{D},\mathcal{R}\rangle}$.*

Now we can introduce our notion of processors which is analogous to the "DP processors" for termination [10, 11]. They transform a DT problem P to a pair (c, P') of an asymptotic complexity $c \in \mathfrak{C}$ and a DT problem P', such that P's complexity is bounded by the maximum of c and of the complexity of P'.

Definition 15 (Processor, $\oplus$). *A* DT processor $\textsc{Proc}$ *is a function* $\textsc{Proc}(P) = (c, P')$ *mapping any DT problem P to a complexity $c \in \mathfrak{C}$ and a DT problem P'. A processor is* sound *if $\iota_P \sqsubseteq c \oplus \iota_{P'}$. Here, "$\oplus$" is the "maximum" function on $\mathfrak{C}$, i.e., for any $c, d \in \mathfrak{C}$, we define $c \oplus d = d$ if $c \sqsubseteq d$ and $c \oplus d = c$ otherwise.*

To analyze the complexity $\iota_\mathcal{R}$ of a TRS $\mathcal{R}$, we start with the canonical DT problem $P_0 = \langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R}\rangle$. Then we apply a sound processor to P_0 which yields a result (c_1, P_1). Afterwards, we apply another (possibly different) sound processor to P_1 which yields (c_2, P_2), etc. This is repeated until we obtain a *solved* DT problem (whose complexity is obviously $\mathcal{P}ol_0$).

Definition 16 (Proof Chain, Solved DT Problem). *We call a DT problem* $P = \langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ solved*, if* $\mathcal{S} = \varnothing$. *A* proof chain[7] *is a finite sequence* $P_0 \overset{c_1}{\leadsto} P_1 \overset{c_2}{\leadsto} \ldots \overset{c_k}{\leadsto} P_k$ *ending with a solved DT problem* P_k*, such that for all* $0 \leq i < k$ *there exists a sound processor* $\textsc{Proc}_i$ *with* $\textsc{Proc}_i(P_i) = (c_{i+1}, P_{i+1})$.

By Def. 15 and 16, for every P_i in a proof chain, $c_{i+1} \oplus \ldots \oplus c_k$ is an upper bound for its complexity ι_{P_i}. Here, the empty sum (for $i = k$) is defined as $\mathcal{P}ol_0$.

Theorem 17 (Approximating Complexity by Proof Chain). *Let* $P_0 \overset{c_1}{\leadsto} P_1 \overset{c_2}{\leadsto} \ldots \overset{c_k}{\leadsto} P_k$ *be a proof chain. Then* $\iota_{P_0} \sqsubseteq c_1 \oplus \ldots \oplus c_k$.

Thm. 14 and 17 now imply that our approach for complexity analysis is correct.

Corollary 18 (Correctness of Approach). *If* P_0 *is the canonical DT problem for a TRS* $\mathcal{R}$ *and* $P_0 \overset{c_1}{\leadsto} \ldots \overset{c_k}{\leadsto} P_k$ *is a proof chain, then* $\iota_{\mathcal{R}} \sqsubseteq c_1 \oplus \ldots \oplus c_k$.

5 DT Processors

In this section, we present several processors to simplify DT problems automatically. To this end, we adapt the processors of the DP framework for termination.

The *usable rules processor* (Sect. 5.1) simplifies a problem $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ by deleting rules from $\mathcal{R}$. The *reduction pair processor* (Sect. 5.2) removes DTs from $\mathcal{S}$, based on term orders. In Sect. 5.3 we introduce the *dependency graph*, on which the *leaf removal* and *knowledge propagation processor* (Sect. 5.4) are based. Finally, Sect. 5.5 adapts processors based on transformations like *narrowing*.

5.1 Usable Rules Processor

As in termination analysis (and in [13]), we can restrict ourselves to those rewrite rules that can be used to reduce right-hand sides of DTs (when instantiating their variables with normal forms). This leads to the notion of *usable rules*.

Definition 19 (Usable Rules $\mathcal{U}_{\mathcal{R}}$ [1]). *For a TRS* $\mathcal{R}$ *and any symbol* f*, let* $\mathit{Rls}_{\mathcal{R}}(f) = \{\ell \to r \mid \mathrm{root}(\ell) = f\}$. *For any term* t, $\mathcal{U}_{\mathcal{R}}(t)$ *is the smallest set with*

- $\mathcal{U}_{\mathcal{R}}(x) = \varnothing$ *if* $x \in \mathcal{V}$ *and*
- $\mathcal{U}_{\mathcal{R}}(f(t_1, \ldots, t_n)) = \mathit{Rls}_{\mathcal{R}}(f) \;\cup\; \bigcup_{\ell \to r \,\in\, \mathit{Rls}_{\mathcal{R}}(f)} \mathcal{U}_{\mathcal{R}}(r) \cup \bigcup_{1 \leq i \leq n} \mathcal{U}_{\mathcal{R}}(t_i)$

For any set $\mathcal{D}$ *of DTs, we define* $\mathcal{U}_{\mathcal{R}}(\mathcal{D}) = \bigcup_{s \to t \,\in\, \mathcal{D}} \mathcal{U}_{\mathcal{R}}(t)$.

So for $\mathcal{R}$ and $DT(\mathcal{R})$ in Ex. 3 and 5, $\mathcal{U}_{\mathcal{R}}(DT(\mathcal{R}))$ contains just the gt- and the p-rules. The following processor removes non-usable rules from DT problems.[8]

Theorem 20 (Usable Rules Processor). *Let* $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ *be a DT problem.*

---

[7] Of course, one could also define DT processors that transform a DT problem P into a complexity c and a *set* $\{P'_1, \ldots, P'_n\}$ such that $\iota_P \sqsubseteq c \oplus \iota_{P'_1} \oplus \ldots \oplus \iota_{P'_n}$. Then instead of a proof chain one would obtain a proof tree.

[8] While Def. 19 is the most basic definition of *usable rules*, the processor of Thm. 20 can also be used with more sophisticated definitions of "usable rules" (e.g., as in [11]).

Then the following processor is sound: $\textsc{Proc}(\langle \mathcal{D}, \mathcal{S}, \mathcal{R}\rangle) = (\mathcal{P}ol_0, \langle \mathcal{D}, \mathcal{S}, \mathcal{U}_{\mathcal{R}}(\mathcal{D})\rangle)$.

So when applying the usable rules processor on the canonical DT problem $\langle \mathcal{D}, \mathcal{D}, \mathcal{R}\rangle$ of $\mathcal{R}$ from Ex. 3, we obtain $\langle \mathcal{D}, \mathcal{D}, \mathcal{R}_1\rangle$ where $\mathcal{R}_1$ are the gt- and p-rules.

5.2 Reduction Pair Processor

The use of orders is one of the most important methods for termination or complexity analysis. In the most basic approach, one tries to find a well-founded order $\succ$ such that every reduction step (strictly) decreases w.r.t. $\succ$. This proves termination and most reduction orders also imply some complexity bound, cf. e.g. [7,15]. However, *direct* applications of orders have two main drawbacks: The obtained bounds are often far too high to be useful and there are many TRSs that cannot be oriented strictly with standard orders amenable to automation.

Therefore, the *reduction pair processor* of the DP framework only requires a strict decrease (w.r.t. $\succ$) for at least one DP, while for all other DPs and rules, a weak decrease (w.r.t. $\succsim$) suffices. Then the strictly decreasing DPs can be deleted. Afterwards one can use other orders (or termination techniques) to solve the remaining DP problem. To adapt the reduction pair processor for complexity analysis, we have to restrict ourselves to $\textsc{Com}$-*monotonic* orders.[9]

Definition 21 (Reduction Pair). *A* reduction pair $(\succsim, \succ)$ *consists of a stable monotonic quasi-order $\succsim$ and a stable well-founded order $\succ$ which are compatible (i.e., $\succsim \circ \succ \circ \succsim \; \subseteq \; \succ$). An order $\succ$ is* $\textsc{Com}$-monotonic *iff* $\textsc{Com}_n(s_1^\sharp, ..., s_i^\sharp, ..., s_n^\sharp) \succ \textsc{Com}_n(s_1^\sharp, ..., t^\sharp, ..., s_n^\sharp)$ *for all $n \in \mathbb{N}$, all $1 \le i \le n$, and all $s_1^\sharp, \ldots, s_n^\sharp, t^\sharp \in \mathcal{T}^\sharp$ with $s_i^\sharp \succ t^\sharp$. A reduction pair $(\succsim, \succ)$ is $\textsc{Com}$-monotonic iff $\succ$ is $\textsc{Com}$-monotonic.*

For a DT problem $(\mathcal{D}, \mathcal{S}, \mathcal{R})$, we orient $\mathcal{D} \cup \mathcal{R}$ by $\succsim$ or $\succ$. But in contrast to the processor for termination, if a DT is oriented strictly, we may not remove it from $\mathcal{D}$, *but only from* $\mathcal{S}$. So the DT is not counted anymore for complexity, but it may still be used in reductions.[10] We will improve this later in Sect. 5.4.

Example 22. This TRS $\mathcal{R}$ shows why DTs may not be removed from $\mathcal{D}$.

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

Let $\mathcal{D} = DT(\mathcal{R}) = \{\mathsf{f}^\sharp(\mathsf{0}) \to \textsc{Com}_0,\ \mathsf{f}^\sharp(\mathsf{s}(x)) \to \textsc{Com}_2(\mathsf{f}^\sharp(\mathsf{id}(x)), \mathsf{id}^\sharp(x)),\ \mathsf{id}^\sharp(\mathsf{0}) \to \textsc{Com}_0,\ \mathsf{id}^\sharp(\mathsf{s}(x)) \to \textsc{Com}_1(\mathsf{id}^\sharp(x))\}$, *where $\mathcal{U}_{\mathcal{R}}(\mathcal{D})$ are just the* id*-rules. For the DT problem $\langle \mathcal{D}, \mathcal{S}, \mathcal{U}_{\mathcal{R}}(\mathcal{D})\rangle$ with $\mathcal{S} = \mathcal{D}$, there is a linear* polynomial interpretation $[\cdot]$ *that orients the first two DTs strictly and the remaining DTs and usable rules weakly:* $[\mathsf{0}] = 0, [\mathsf{s}](x) = x + 1, [\mathsf{id}](x) = x, [\mathsf{f}^\sharp](x) = x + 1, [\mathsf{id}^\sharp](x) =$

[9] In [13] "$\textsc{Com}$-monotonic" is called "safe". Note that our reduction pair processor is much closer to the original processor of the DP framework than [13]. In the main theorem of [13], all (weak) DPs have to be oriented strictly in one go. Moreover, one even has to orient the (usable) rules strictly. Finally, one is either restricted to non-duplicating TRSs or one has to use orderings $\succ$ that are monotonic on *all* symbols.

[10] A related idea is used in [22]. However, [22] focuses on derivational complexity instead of (innermost) runtime complexity, and it operates directly on TRSs and not on DPs or DTs. Therefore, [22] has to impose stronger restrictions (it requires $\succ$ to be monotonic on *all* symbols) and it does not use other DP- resp. DT-based processors.

$0, [\textsc{Com}_0] = 0, [\textsc{Com}_1](x) = x, [\textsc{Com}_2](x, y) = x + y$. *If one would remove the first two DTs from $\mathcal{D}$, there is another linear polynomial interpretation that orients the remaining DTs strictly (e.g., by $[\mathsf{id}^\sharp](x) = x + 1$). Then, one would falsely conclude that the whole TRS has linear runtime complexity.*

Hence, the first two DTs should only be removed from $\mathcal{S}$, not from $\mathcal{D}$. This results in $\langle \mathcal{D}, \mathcal{S}', \mathcal{U}_\mathcal{R}(\mathcal{D}) \rangle$ where $\mathcal{S}'$ consists of the last two DTs. These DTs can occur quadratically often in reductions with $\mathcal{D} \cup \mathcal{U}_\mathcal{R}(\mathcal{D})$. Hence, when trying to orient $\mathcal{S}'$ strictly and the remaining DTs and usable rules weakly, we have to use a quadratic polynomial interpretation (e.g., $[\mathsf{0}] = 0, [\mathsf{s}](x) = x + 2, [\mathsf{id}](x) = x, [\mathsf{f}^\sharp](x) = x^2, [\mathsf{id}^\sharp](x) = x + 1, [\textsc{Com}_0] = 0, [\textsc{Com}_1](x) = x, [\textsc{Com}_2](x, y) = x + y$). Hence, now we (correctly) conclude that the TRS has quadratic runtime complexity (indeed, $\mathrm{dl}(\mathsf{f}(\mathsf{s}^n(\mathsf{0})), \xrightarrow{\mathrm{i}}_\mathcal{R}) = \frac{(n+1)\cdot(n+2)}{2}$).

So when applying the reduction pair processor to $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$, we obtain $(c, \langle \mathcal{D}, \mathcal{S} \setminus \mathcal{D}_\succ, \mathcal{R} \rangle)$. Here, $\mathcal{D}_\succ$ are the strictly decreasing DTs from $\mathcal{D}$ and c is an upper bound for the number of $\mathcal{D}_\succ$-steps in innermost reductions with $\mathcal{D} \cup \mathcal{R}$.

Theorem 23 (Reduction Pair Processor). *Let $P = \langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ be a DT problem and $(\succsim, \succ)$ be a* Com*-monotonic reduction pair. Let $\mathcal{D} \subseteq {\succsim} \cup {\succ}$, $\mathcal{R} \subseteq {\succsim}$, and $c \sqsupseteq \iota(\mathrm{rc}_\succ)$ for the function $\mathrm{rc}_\succ(n) = \sup\{ \mathrm{dl}(t^\sharp, \succ) \mid t \in \mathcal{T}_B, |t| \le n\}$.*[11] *Then the following processor is sound:* $\textsc{Proc}(\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle) = (c, \langle \mathcal{D}, \mathcal{S} \setminus \mathcal{D}_\succ, \mathcal{R} \rangle)$.

To automate Thm. 23, we need reduction pairs where an upper bound c for $\iota(\mathrm{rc}_\succ)$ can be computed easily. This holds for reduction pairs based on *polynomial interpretations* with coefficients from $\mathbb{N}$ (which are well suited for automation). For Com-monotonicity, we restrict ourselves to *complexity polynomial interpretations* $[\cdot]$ where $[\textsc{Com}_n](x_1, \ldots, x_n) = x_1 + \ldots + x_n$ for all $n \in \mathbb{N}$. This is the smallest polynomial which is monotonic in $x_1, \ldots, x_n$. As $\textsc{Com}_n$ only occurs on right-hand sides of $\succ$- and $\succsim$-inequalities, $[\textsc{Com}_n]$ should be as small as possible.

Moreover, a *complexity polynomial interpretation* interprets constructor symbols $f \in \Sigma \setminus \Sigma_d$ by polynomials $[f](x_1, \ldots, x_n)$ of the form $a_1 x_1 + \ldots + a_n x_n + b$ where $b \in \mathbb{N}$ and $a_i \in \{0, 1\}$. This ensures that the mapping from constructor ground terms $t \in \mathcal{T}(\Sigma \setminus \Sigma_d, \varnothing)$ to their interpretations is in $\mathcal{O}(|t|)$, cf. [7,15]. Note that the interpretations in Ex. 22 were *complexity polynomial interpretations*.

Thm. 24 shows how such interpretations can be used[12] for the processor of Thm. 23. Here, as an upper bound c for $\iota(\mathrm{rc}_\succ)$, one can simply take $\mathcal{P}ol_m$, where m is the maximal degree of the polynomials in the interpretation.

Theorem 24 (Reduction Pair Processor with Polynomial Interpretations). *Let $P = \langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ be a DT problem and let $\succsim$ and $\succ$ be induced by a*

[11] As noted by [19], this can be weakened by replacing "$\mathrm{dl}(t^\sharp, \succ)$" with $\mathrm{dl}(t^\sharp, {\succ} \cap \xrightarrow{\mathrm{i}}_{\mathcal{D}/\mathcal{R}})$, where ${\to_{\mathcal{D}/\mathcal{R}}} = {\to_\mathcal{R}^* \circ \to_\mathcal{D} \circ \to_\mathcal{R}^*}$ and $\xrightarrow{\mathrm{i}}_{\mathcal{D}/\mathcal{R}}$ is the restriction of $\to_{\mathcal{D}/\mathcal{R}}$ where in each rewrite step with $\to_\mathcal{R}$ or $\to_\mathcal{D}$, the arguments of the redex must be in $(\mathcal{D} \cup \mathcal{R})$-normal form, cf. [3]. Such a weakening is required to use reduction pairs based on path orders where a term $t^\sharp$ may start $\succ$-decreasing sequences of arbitrary (finite) length.

[12] Alternatively, our reduction pair processor can also use matrix interpretations [8,16, 18,20,21], polynomial path orders (POP^* [3]), etc. For POP^*, we would extend $\mathfrak{C}$ by a complexity $\mathcal{P}ol_*$ for polytime computability, where $\mathcal{P}ol_n \sqsubset \mathcal{P}ol_* \sqsubset \infty$ for all $n \in \mathbb{N}$.

complexity polynomial interpretation $[\cdot]$. *Let* $m \in \mathbb{N}$ *be the maximal degree of all polynomials* $[f^\sharp]$, *for all* $f^\sharp$ *with* $f \in \Sigma_d$. *Let* $\mathcal{D} \subseteq \succsim \cup \succ$ *and* $\mathcal{R} \subseteq \succsim$. *Then the following processor is sound:* $\text{Proc}(\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle) = (\mathcal{P}ol_m,\ \langle \mathcal{D},\ \mathcal{S} \setminus \mathcal{D}_\succ,\ \mathcal{R} \rangle)$.

Example 25. This TRS [1] illustrates Thm. 24, where $\mathsf{q}(x, y, y)$ *computes* $\lfloor \frac{x}{y} \rfloor$.

$$\mathsf{q}(\mathsf{0}, \mathsf{s}(y), \mathsf{s}(z)) \to \mathsf{0} \quad \mathsf{q}(\mathsf{s}(x), \mathsf{s}(y), z) \to \mathsf{q}(x, y, z) \quad \mathsf{q}(x, \mathsf{0}, \mathsf{s}(z)) \to \mathsf{s}(\mathsf{q}(x, \mathsf{s}(z), \mathsf{s}(z)))$$

The dependency tuples $\mathcal{D}$ *of this TRS are*

$$\mathsf{q}^\sharp(\mathsf{0}, \mathsf{s}(y), \mathsf{s}(z)) \to \text{Com}_0 \quad (9) \qquad \mathsf{q}^\sharp(\mathsf{s}(x), \mathsf{s}(y), z) \to \text{Com}_1(\mathsf{q}^\sharp(x, y, z)) \quad (10)$$

$$\mathsf{q}^\sharp(x, \mathsf{0}, \mathsf{s}(z)) \to \text{Com}_1(\mathsf{q}^\sharp(x, \mathsf{s}(z), \mathsf{s}(z))) \quad (11)$$

As the usable rules are empty, Thm. 20 transforms the canonical DT problem to $\langle \mathcal{D}, \mathcal{D}, \varnothing \rangle$. *Consider the complexity polynomial interpretation* $[\mathsf{0}] = 0$, $[\mathsf{s}](x) = x + 1$, $[\mathsf{q}^\sharp](x, y, z) = x + 1$, $[\text{Com}_0] = 0$, $[\text{Com}_1](x) = x$. *With the corresponding reduction pair, the DTs* (9) *and* (10) *are strictly decreasing and* (11) *is weakly decreasing. Moreover, the degree of* $[\mathsf{q}^\sharp]$ *is 1. Hence, the reduction pair processor returns* $(\mathcal{P}ol_1, \langle \mathcal{D}, \{(11)\}, \varnothing \rangle)$. *Unfortunately, no reduction pair based on complexity polynomial interpretations orients* (11) *strictly and both* (9) *and* (10) *weakly. So for the moment we cannot simplify this problem further.*

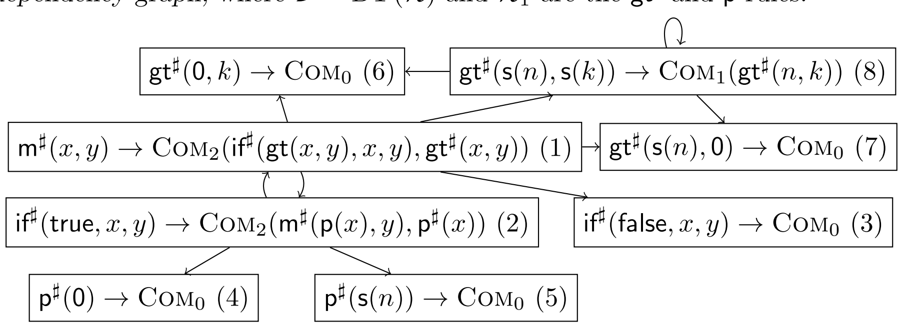
5.3 Dependency Graph Processors

As in the DP framework for termination, it is useful to have a finite representation of (a superset of) all possible chain trees.

Definition 26 (Dependency Graph). *Let* $\mathcal{D}$ *be a set of DTs and* $\mathcal{R}$ *a TRS. The* $(\mathcal{D}, \mathcal{R})$-dependency graph *is the directed graph whose nodes are the DTs in* $\mathcal{D}$ *and there is an edge from* $s \to t$ *to* $u \to v$ *in the dependency graph iff there is a chain tree with an edge from a node* $(s \to t \mid \sigma_1)$ *to a node* $(u \to v \mid \sigma_2)$.

Every $(\mathcal{D}, \mathcal{R})$-chain corresponds to a path in the $(\mathcal{D}, \mathcal{R})$-dependency graph. While dependency graphs are not computable in general, there are several techniques to compute over-approximations of dependency graphs for termination, cf. e.g. [1]. These techniques can also be applied for $(\mathcal{D}, \mathcal{R})$-dependency graphs.

Example 27. For the TRS $\mathcal{R}$ *from Ex. 3, we obtain the following* $(\mathcal{D}, \mathcal{R}_1)$*-dependency graph, where* $\mathcal{D} = DT(\mathcal{R})$ *and* $\mathcal{R}_1$ *are the* gt- *and* p-*rules.*



For termination analysis, one can regard each cycle of the graph separately and ignore nodes that are not on cycles. This is not possible for complexity analysis: If one regards the DTs $\mathcal{D}' = \{(1),(2)\}$ and $\mathcal{D}'' = \{(8)\}$ on the two cycles of the dependency graph separately, then both resulting DT problems $\langle \mathcal{D}', \mathcal{D}', \mathcal{R}_1 \rangle$ and $\langle \mathcal{D}'', \mathcal{D}'', \mathcal{R}_1 \rangle$ have linear complexity. However, this does not allow any conclusions on the complexity of $\langle \mathcal{D}, \mathcal{D}, \mathcal{R}_1 \rangle$ (which is quadratic). Nevertheless, it is possible to remove DTs $s \to t$ that are leaves *(i.e., $s \to t$ has no successors in the dependency graph). This yields $\langle \mathcal{D}_1, \mathcal{D}_1, \mathcal{R}_1 \rangle$, where $\mathcal{D}_1 = \{(1),(2),(8)\}$.*

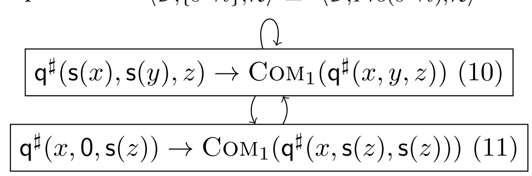
Theorem 28 (Leaf Removal Processor). *Let $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ be a DT problem and let $s \to t \in \mathcal{D}$ be a leaf in the $(\mathcal{D}, \mathcal{R})$-dependency graph. Then the following processor is sound:* $\textsc{Proc}(\,\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle\,) = (\mathcal{P}ol_0, \langle \mathcal{D} \setminus \{s \to t\}, \mathcal{S} \setminus \{s \to t\}, \mathcal{R} \rangle)$.

5.4 Knowledge Propagation

In the DP framework for termination, the reduction pair processor removes "strictly decreasing" DPs. While this is unsound for complexity analysis (cf. Ex. 22), we now show that by an appropriate *extension* of DT problems, one can obtain a similar processor also for complexity analysis.

Lemma 29 shows that we can estimate the complexity of a DT if we know the complexity of all its *predecessors* in the dependency graph.

Lemma 29 (Complexity Bounded by Predecessors). *Let $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ be a DT problem and $s \to t \in \mathcal{D}$. Let $Pre(s \to t) \subseteq \mathcal{D}$ be the predecessors of $s \to t$, i.e., $Pre(s \to t)$ contains all DTs $u \to v$ where there is an edge from $u \to v$ to $s \to t$ in the $(\mathcal{D}, \mathcal{R})$-dependency graph. Then $\iota_{\langle \mathcal{D}, \{s \to t\}, \mathcal{R} \rangle} \sqsubseteq \iota_{\langle \mathcal{D}, Pre(s \to t), \mathcal{R} \rangle}$.*



Example 30. *Consider the TRS from Ex. 25. By usable rules and reduction pairs, we obtained $\langle \mathcal{D}, \{(11)\}, \varnothing \rangle$ for $\mathcal{D} = \{(9),(10),(11)\}$. The leaf removal processor yields $\langle \mathcal{D}', \{(11)\}, \varnothing \rangle$ with $\mathcal{D}' = \{(10),(11)\}$. Consider the the $(\mathcal{D}', \varnothing)$-dependency graph above. We have $\iota_{\langle \mathcal{D}', \{(11)\}, \varnothing \rangle} \sqsubseteq \iota_{\langle \mathcal{D}', \{(10)\}, \varnothing \rangle}$ by Lemma 29, since* (10) *is the only predecessor of* (11)*. Thus, the complexity of $\langle \mathcal{D}', \{(11)\}, \varnothing \rangle$ does not matter for the overall complexity, if we can guarantee that we have already taken the complexity of $\langle \mathcal{D}', \{(10)\}, \varnothing \rangle$ into account.*

Therefore, we now extend the definition of DT problems by a set $\mathcal{K}$ of DTs with "known" complexity, i.e., the complexity of the DTs in $\mathcal{K}$ has already been taken into account. So a processor only needs to estimate the complexity of a set of DTs correctly if their complexity is higher than the complexity of the DTs in $\mathcal{K}$. Otherwise, the processor may return an arbitrary result. To this end, we introduce a "subtraction" operation $\ominus$ on complexities from $\mathfrak{C}$.

Definition 31 (Extended DT Problems, $\ominus$). *For $c, d, \in \mathfrak{C}$, let $c \ominus d = c$ if $d \sqsubset c$ and $c \ominus d = \mathcal{P}ol_0$ if $c \sqsubseteq d$. Let $\mathcal{R}$ be a TRS, $\mathcal{D}$ a set of DTs, and $\mathcal{S}, \mathcal{K} \subseteq \mathcal{D}$. Then $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle$ is an* extended DT problem *and $\langle DT(\mathcal{R}), DT(\mathcal{R}), \varnothing, \mathcal{R} \rangle$ is the* canonical extended DT problem *for $\mathcal{R}$. We define the complexity of an extended*

DT problem to be $\gamma_{\langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{R}\rangle} = \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \ominus \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle}$ *and also use* γ *instead of* ι *in the soundness condition for processors. So on extended DT problems, a processor with* $\textsc{Proc}(P) = (c, P')$ *is* sound *if* $\gamma_P \sqsubseteq c \oplus \gamma_{P'}$*. An extended DT problem* $\langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{R}\rangle$ *is* solved *if* $\mathcal{S} = \varnothing$*.*

So for $\mathcal{K} = \varnothing$, the definition of "complexity" for extended DT problems is equivalent to complexity for ordinary DT problems, i.e., $\gamma_{\langle \mathcal{D},\mathcal{S},\varnothing,\mathcal{R}\rangle} = \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$. Cor. 32 shows that our approach is still correct for extended DT problems.

Corollary 32 (Correctness). *If* P_0 *is the canonical extended DT problem for a TRS* $\mathcal{R}$ *and* $P_0 \overset{c_1}{\rightsquigarrow} \ldots \overset{c_k}{\rightsquigarrow} P_k$ *is a proof chain, then* $\iota_\mathcal{R} = \gamma_{P_0} \sqsubseteq c_1 \oplus \ldots \oplus c_k$*.*

Now we introduce a processor which makes use of $\mathcal{K}$. It moves a DT $s \to t$ from $\mathcal{S}$ to $\mathcal{K}$ whenever the complexity of all predecessors of $s \to t$ in the dependency graph has already been taken into account.[13]

Theorem 33 (Knowledge Propagation Processor). *Let* $\langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{R}\rangle$ *be an extended DT problem,* $s \to t \in \mathcal{S}$*, and* $\mathit{Pre}(s \to t) \subseteq \mathcal{K}$*. Then the following processor is sound:* $\textsc{Proc}(\,\langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{R}\rangle\,) = (\mathcal{P}ol_0,\ \langle \mathcal{D},\ \mathcal{S}\setminus\{s \to t\},\ \mathcal{K}\cup\{s \to t\},\ \mathcal{R}\rangle\,)$.

Before we can illustrate this processor, we need to adapt the previous processors to *extended* DT problems. The adaption of the usable rules and leaf removal processors is straightforward. But now the reduction pair processor does not only delete DTs from $\mathcal{S}$, but it can also move them to $\mathcal{K}$. The reason is that the complexity of these DTs is bounded by the complexity value $c \in \mathfrak{C}$ returned by the processor. (Of course, the special case of the reduction pair processor with polynomial interpretations of Thm. 24 can be adapted analogously.)

Theorem 34 (Processors for Extended DT Problems). *Let* $P = \langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{R}\rangle$ *be an extended DT problem. Then the following processors are sound.*

- *The usable rules processor:* $\textsc{Proc}(P) = (\mathcal{P}ol_0, \langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{U}_\mathcal{R}(\mathcal{D})\rangle)$.
- *The leaf removal processor* $\textsc{Proc}(P) = (\mathcal{P}ol_0, \langle \mathcal{D} \setminus \{s \to t\}, \mathcal{S} \setminus \{s \to t\}, \mathcal{K} \setminus \{s \to t\}, \mathcal{R}\rangle)$*, if* $s \to t$ *is a leaf in the* $(\mathcal{D},\mathcal{R})$*-dependency graph.*
- *The reduction pair processor:* $\textsc{Proc}(P) = (c,\ \langle \mathcal{D},\ \mathcal{S} \setminus \mathcal{D}_\succ,\ \mathcal{K} \cup \mathcal{D}_\succ,\ \mathcal{R}\rangle)$*, if* $(\succsim, \succ)$ *is a* Com*-monotonic reduction pair,* $\mathcal{D} \subseteq \succsim \cup \succ$*,* $\mathcal{R} \subseteq \succsim$*, and* $c \sqsupseteq \iota(\mathrm{rc}_\succ)$ *for the function* $\mathrm{rc}_\succ(n) = \sup\{\,\mathrm{dl}(t^\sharp, \succ) \mid t \in \mathcal{T}_B, |t| \le n\}$*.*

Example 35. Reconsider the TRS $\mathcal{R}$ *for division from Ex. 25. Starting with its canonical extended DT problem, we now obtain the following proof chain.*

$$\begin{array}{llllll}
 & \langle\,\{(9),(10),(11)\}, & \{(9),(10),(11)\}, & \varnothing, & \mathcal{R}\rangle & \\
\overset{\mathcal{P}ol_0}{\rightsquigarrow} & \langle\,\{(10),(11)\}, & \{(10),(11)\}, & \varnothing, & \mathcal{R}\rangle & \textit{(leaf removal)}\\
\overset{\mathcal{P}ol_0}{\rightsquigarrow} & \langle\,\{(10),(11)\}, & \{(10),(11)\}, & \varnothing, & \varnothing\rangle & \textit{(usable rules)}\\
\overset{\mathcal{P}ol_1}{\rightsquigarrow} & \langle\,\{(10),(11)\}, & \{(11)\}, & \{(10)\}, & \varnothing\rangle & \textit{(reduction pair)}\\
\overset{\mathcal{P}ol_0}{\rightsquigarrow} & \langle\,\{(10),(11)\}, & \varnothing, & \{(10),(11)\}, & \varnothing\rangle & \textit{(knowledge propag.)}
\end{array}$$

For the last step we use $\mathit{Pre}((11)) = \{(10)\}$*, cf. Ex. 30. The last DT problem is solved. Thus,* $\iota_\mathcal{R} \sqsubseteq \mathcal{P}ol_0 \oplus \mathcal{P}ol_0 \oplus \mathcal{P}ol_1 \oplus \mathcal{P}ol_0 = \mathcal{P}ol_1$*, i.e.,* $\mathcal{R}$ *has linear complexity.*

[13] In particular, this means that nodes without predecessors (i.e., "roots" of the dependency graph that are not in any cycle) can always be moved from $\mathcal{S}$ to $\mathcal{K}$.

5.5 Transformation Processors

To increase power, the DP framework for termination analysis has several processors which *transform* a DP into new ones (by "narrowing", "rewriting", "instantiation", or "forward instantiation") [11]. We now show how to adapt such processors for complexity analysis. For reasons of space, we only present the narrowing processor (the other processors can be adapted in a similar way).

For a DT problem $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle$, let $s \to t \in \mathcal{D}$ with $t = \textsc{Com}_n(t_1, ..., t_i, ..., t_n)$. If there exists a (variable-renamed) $u \to v \in \mathcal{D}$ where t_i and u have an mgu μ and both $s\mu$ and $u\mu$ are in $\mathcal{R}$-normal form, then we call μ a *narrowing substitution* of t_i and define the corresponding *narrowing result* to be $t_i\mu$.

Moreover, if $s \to t$ has a successor $u \to v$ in the $(\mathcal{D}, \mathcal{R})$-dependency graph where t_i and u have no such mgu, then we obtain additional narrowing substitutions and narrowing results for t_i. The reason is that in any possible reduction $t_i\sigma \xrightarrow{\mathrm{i}}^*_{\mathcal{R}} u\tau$ in a chain, the term $t_i\sigma$ must be rewritten at least one step before it reaches $u\tau$. The idea of the narrowing processor is to already perform this first reduction step directly on the DT $s \to t$. Whenever a subterm $t_i|_\pi \notin \mathcal{V}$ of t_i unifies with the left-hand side of a (variable-renamed) rule $\ell \to r \in \mathcal{R}$ using an mgu μ where $s\mu$ is in $\mathcal{R}$-normal form, then μ is a *narrowing substitution* of t_i and the corresponding *narrowing result* is $w = t_i[r]_\pi\mu$.

If $\mu_1, \ldots, \mu_d$ are all narrowing substitutions of t_i with the corresponding narrowing results $w_1, \ldots, w_d$, then $s \to t$ can be replaced by $s\mu_j \to \textsc{Com}_n(t_1\mu_j, \ldots, t_{i-1}\mu_j, w_j, t_{i+1}\mu_j, \ldots, t_n\mu_j)$ for all $1 \leq j \leq d$.

However, there could be a t_k (with $k \neq i$) which was involved in a chain (i.e., $t_k\sigma \xrightarrow{\mathrm{i}}^*_{\mathcal{R}} u\tau$ for some $u \to v \in \mathcal{D}$ and some σ, τ), but this chain is no longer possible when instantiating t_k to $t_k\mu_1, \ldots, t_k\mu_d$. We say that t_k is *captured* by μ_1, $\ldots, \mu_d$ if for each narrowing substitution ρ of t_k, there is a μ_j that is more general (i.e., $\rho = \mu_j\, \rho'$ for some substitution ρ'). The narrowing processor has to add another DT $s \to \textsc{Com}_m(t_{k_1}, \ldots, t_{k_m})$ where $t_{k_1}, \ldots, t_{k_m}$ are all terms from t_1, $\ldots, t_n$ which are not captured by the narrowing substitutions $\mu_1, \ldots, \mu_d$ of t_i.

This leads to the following processor. For any sets $\mathcal{D}, \mathcal{M}$ of DTs, $\mathcal{D}[s \to t \,/\, \mathcal{M}]$ denotes the result of replacing $s \to t$ by the DTs in $\mathcal{M}$. So if $s \to t \in \mathcal{D}$, then $\mathcal{D}[s \to t \,/\, \mathcal{M}] = (DT \setminus \{s \to t\}) \cup \mathcal{M}$ and otherwise, $\mathcal{D}[s \to t \,/\, \mathcal{M}] = \mathcal{D}$.

Theorem 36 (Narrowing Processor). *Let $P = \langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle$ be an extended DT problem and let $s \to t \in \mathcal{D}$ with $t = \textsc{Com}_n(t_1, \ldots, t_i, \ldots, t_n)$. Let $\mu_1, \ldots, \mu_d$ be the narrowing substitutions of t_i with the corresponding narrowing results $w_1, \ldots, w_d$, where $d \geq 0$. Let $t_{k_1}, \ldots, t_{k_m}$ be the terms from $t_1, \ldots, t_n$ that are not captured by $\mu_1, \ldots, \mu_d$, where $k_1, \ldots, k_m$ are pairwise different. We define*

$$\begin{aligned}\mathcal{M} = \; & \{ s\mu_j \to \textsc{Com}_n(t_1\mu_j, \ldots, t_{i-1}\mu_j, w_j, t_{i+1}\mu_j, \ldots, t_n\mu_j) \mid 1 \leq j \leq d \} \\ \cup \; & \{ s \quad \to \textsc{Com}_m(t_{k_1}, \ldots, t_{k_m}) \}.\end{aligned}$$

Then the following processor is sound: $\textsc{Proc}(P) = (\mathcal{P}ol_0, \langle \mathcal{D}', \mathcal{S}', \mathcal{K}', \mathcal{R} \rangle)$*, where $\mathcal{D}' = \mathcal{D}[s \to t \,/\, \mathcal{M}]$ and $\mathcal{S}' = \mathcal{S}[s \to t \,/\, \mathcal{M}]$. $\mathcal{K}'$ results from $\mathcal{K}$ by removing $s \to t$ and all DTs that are reachable from $s \to t$ in the $(\mathcal{D}, \mathcal{R})$-dependency graph.*[14]

[14] We cannot define $\mathcal{K}' = \mathcal{K}[s \to t \,/\, \mathcal{M}]$, because the narrowing step performed on $s \to t$ does not necessarily correspond to an *innermost* reduction. Hence, there can

Example 37. *To illustrate the narrowing processor, consider the following TRS.*

$$\mathsf{f}(\mathsf{c}(n,x)) \to \mathsf{c}(\,\mathsf{f}(\mathsf{g}(\mathsf{c}(n,x))),\ \mathsf{f}(\mathsf{h}(\mathsf{c}(n,x)))\,) \qquad \mathsf{g}(\mathsf{c}(\mathsf{0},x)) \to x \qquad \mathsf{h}(\mathsf{c}(\mathsf{1},x)) \to x$$

So f *operates on "lists" of* 0s *and* 1s, *where* g *removes a leading* 0 *and* h *removes a leading* 1. *Since* g*'s and* h*'s applicability "exclude" each other, the TRS has linear (and not exponential) complexity. The leaf removal and usable rules processors yield the problem* $\langle\,\{(12)\},\ \{(12)\},\ \varnothing,\ \{\mathsf{g}(\mathsf{c}(\mathsf{0},x)) \to x,\ \mathsf{h}(\mathsf{c}(\mathsf{1},x)) \to x\}\,\rangle$ *with*

$$\mathsf{f}^\sharp(\mathsf{c}(n,x)) \to \textsc{Com}_4(\,\mathsf{f}^\sharp(\mathsf{g}(\mathsf{c}(n,x))),\ \mathsf{g}^\sharp(\mathsf{c}(n,x)),\ \mathsf{f}^\sharp(\mathsf{h}(\mathsf{c}(n,x))),\ \mathsf{h}^\sharp(\mathsf{c}(n,x))\,). \quad (12)$$

The only narrowing substitution of $t_1 = \mathsf{f}^\sharp(\mathsf{g}(\mathsf{c}(n,x)))$ *is* $[n/\mathsf{0}]$ *and the corresponding narrowing result is* $\mathsf{f}^\sharp(x)$. *However,* $t_3 = \mathsf{f}^\sharp(\mathsf{h}(\mathsf{c}(n,x)))$ *is not captured by the substitution* $[n/\mathsf{0}]$, *since* $[n/\mathsf{0}]$ *is not more general than* t_3*'s narrowing substitution* $[n/\mathsf{1}]$. *Hence, the DT (12) is replaced by the following two new DTs:*

$$\mathsf{f}^\sharp(\mathsf{c}(\mathsf{0},x)) \to \textsc{Com}_4(\,\mathsf{f}^\sharp(x),\ \mathsf{g}^\sharp(\mathsf{c}(\mathsf{0},x)),\ \mathsf{f}^\sharp(\mathsf{h}(\mathsf{c}(\mathsf{0},x))),\ \mathsf{h}^\sharp(\mathsf{c}(\mathsf{0},x))\,) \quad (13)$$

$$\mathsf{f}^\sharp(\mathsf{c}(n,x)) \to \textsc{Com}_1(\,\mathsf{f}^\sharp(\mathsf{h}(\mathsf{c}(n,x)))\,) \quad (14)$$

Another application of the narrowing processor replaces (14) by $\mathsf{f}^\sharp(\mathsf{c}(\mathsf{1},x)) \to \textsc{Com}_1(\mathsf{f}^\sharp(x))$.[15] *Now* $\iota_{\mathcal{R}} \sqsubseteq \mathcal{P}ol_1$ *is easy to show by the reduction pair processor.*

Example 38. *Reconsider the TRS of Ex. 3. The canonical extended DT problem is transformed to* $\langle \mathcal{D}_1, \mathcal{D}_1, \varnothing, \mathcal{R}_1\rangle$, *where* $\mathcal{D}_1 = \{(1),(2),(8)\}$ *and* $\mathcal{R}_1$ *are the* gt- *and* p-*rules, cf. Ex. 27. In* $\mathsf{m}^\sharp(x,y) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{gt}(x,y),x,y), \mathsf{gt}^\sharp(x,y))$ (1), *one can narrow* $t_1 = \mathsf{if}^\sharp(\mathsf{gt}(x,y),x,y)$. *Its narrowing substitutions are* $[x/\mathsf{0}, y/k]$, $[x/\mathsf{s}(n), y/\mathsf{0}], [x/\mathsf{s}(n), y/\mathsf{s}(k)]$. *Note that* $t_2 = \mathsf{gt}^\sharp(x,y)$ *is captured, as its only narrowing substitution is* $[x/\mathsf{s}(n), y/\mathsf{s}(k)]$. *So (1) can be replaced by*

$$\mathsf{m}^\sharp(\mathsf{0},k) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{false},\mathsf{0},k), \mathsf{gt}^\sharp(\mathsf{0},k)) \quad (15)$$

$$\mathsf{m}^\sharp(\mathsf{s}(n),\mathsf{0}) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{true},\mathsf{s}(n),\mathsf{0}), \mathsf{gt}^\sharp(\mathsf{s}(n),\mathsf{0})) \quad (16)$$

$$\mathsf{m}^\sharp(\mathsf{s}(n),\mathsf{s}(k)) \to \textsc{Com}_2(\mathsf{if}^\sharp(\mathsf{gt}(n,k),\mathsf{s}(n),\mathsf{s}(k)), \mathsf{gt}^\sharp(\mathsf{s}(n),\mathsf{s}(k))) \quad (17)$$

$$\mathsf{m}^\sharp(x,y) \to \textsc{Com}_0 \quad (18)$$

The leaf removal processor deletes (15), (18) *and yields* $\langle \mathcal{D}_2, \mathcal{D}_2, \varnothing, \mathcal{R}_1\rangle$ *with* $\mathcal{D}_2 = \{(16),(17),(2),(8)\}$. *We replace* $\mathsf{if}^\sharp(\mathsf{true},x,y) \to \textsc{Com}_2(\mathsf{m}^\sharp(\mathsf{p}(x),y), \mathsf{p}^\sharp(x))$ (2) *by*

$$\mathsf{if}^\sharp(\mathsf{true},\mathsf{0},y) \to \textsc{Com}_2(\mathsf{m}^\sharp(\mathsf{0},y), \mathsf{p}^\sharp(\mathsf{0})) \quad (19)$$

$$\mathsf{if}^\sharp(\mathsf{true},\mathsf{s}(n),y) \to \textsc{Com}_2(\mathsf{m}^\sharp(n,y), \mathsf{p}^\sharp(\mathsf{s}(n))) \quad (20)$$

by the narrowing processor. The leaf removal processor deletes (19) and the usable rules processor removes the p-*rules from* $\mathcal{R}_1$. *This yields* $\langle \mathcal{D}_3, \mathcal{D}_3, \varnothing, \mathcal{R}_2\rangle$,

---

be $(\mathcal{D}',\mathcal{R})$-chains that correspond to non-innermost reductions with $\mathcal{D}\cup\mathcal{R}$. So there may exist terms whose maximal $(\mathcal{D}',\mathcal{R})$-chain tree is larger than their maximal $(\mathcal{D},\mathcal{R})$-chain tree and thus, $\iota_{\langle \mathcal{D}',\mathcal{K}[s\to t/\mathcal{M}],\mathcal{R}\rangle} \sqsupseteq \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle}$. But we need $\iota_{\langle \mathcal{D}',\mathcal{K}',\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle}$ in order to guarantee the soundness of the processor, i.e., to ensure that $\gamma_{\langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{R}\rangle} = \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \ominus \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D}',\mathcal{S}',\mathcal{R}\rangle} \ominus \iota_{\langle \mathcal{D}',\mathcal{K}',\mathcal{R}\rangle} = \gamma_{\langle \mathcal{D}',\mathcal{S}',\mathcal{K}',\mathcal{R}\rangle}$.

[15] One can also simplify (13) further by narrowing. Its subterm $\mathsf{g}^\sharp(\mathsf{c}(\mathsf{0},x))$ has no narrowing substitutions. This (empty) set of narrowing substitutions captures $\mathsf{f}^\sharp(\mathsf{h}(\mathsf{c}(\mathsf{0},x)))$ and $\mathsf{h}^\sharp(\mathsf{c}(\mathsf{0},x))$ which have no narrowing substitutions either. Since $\mathsf{f}^\sharp(x)$ is not captured, (13) can be transformed into $\mathsf{f}^\sharp(\mathsf{c}(\mathsf{0},x)) \to \textsc{Com}_1(\mathsf{f}^\sharp(x))$.

where $\mathcal{D}_3 = \{(16), (17), (20), (8)\}$ *and* $\mathcal{R}_2$ *are the* gt*-rules. By the polynomial interpretation* $[0] = [\mathsf{true}] = [\mathsf{false}] = [\mathsf{p}^\sharp](x) = 0$, $[\mathsf{s}](x) = x+2$, $[\mathsf{gt}](x, y) = [\mathsf{gt}^\sharp](x, y) = x$, $[\mathsf{m}^\sharp](x, y) = (x+1)^2$, $[\mathsf{if}^\sharp](x, y, z) = y^2$, *all DTs in* $\mathcal{D}_3$ *are strictly decreasing and all rules in* $\mathcal{R}_2$ *are weakly decreasing. So the reduction pair processor yields* $\langle \mathcal{D}_3, \mathcal{D}_3, \varnothing, \mathcal{R}_2 \rangle \stackrel{\mathcal{P}ol_2}{\leadsto} \langle \mathcal{D}_3, \varnothing, \mathcal{D}_3, \mathcal{R}_2 \rangle$*. As this DT problem is solved, we obtain* $\iota_\mathcal{R} \sqsubseteq \mathcal{P}ol_0 \oplus \ldots \oplus \mathcal{P}ol_0 \oplus \mathcal{P}ol_2 = \mathcal{P}ol_2$*, i.e.,* $\mathcal{R}$ *has quadratic complexity.*

6 Evaluation and Conclusion

We presented a new technique for (innermost) runtime complexity analysis by adapting the termination techniques of the DP framework. To this end, we introduced several processors to simplify "DT problems", which gives rise to a flexible and modular framework for automated complexity proofs. Thus, recent advances in termination analysis can now also be used for complexity analysis.

To evaluate our contributions, we implemented them in the termination prover AProVE and compared it with the complexity tools CaT 1.5 [22] and TCT 1.6 [2]. We ran the tools on 1323 TRSs from the *Termination Problem Data Base* used in the *International Termination Competition* 2010.[16] As in the competition, each tool had a timeout of 60 seconds for each example. The left half of the table compares CaT and AProVE. For instance, the first row means that AProVE showed constant complexity for 209 examples. On those examples, CaT proved linear complexity in 182 cases and failed in 27 cases. So in the light gray part of the table, AProVE gave more precise results than CaT. In the medium gray part, both tools obtained equal results. In the dark gray part, CaT was more precise than AProVE. Similarly, the right half of the table compares TCT and AProVE.

		CaT						TCT					
		$\mathcal{P}ol_0$	$\mathcal{P}ol_1$	$\mathcal{P}ol_2$	$\mathcal{P}ol_3$	no result	$\sum$	$\mathcal{P}ol_0$	$\mathcal{P}ol_1$	$\mathcal{P}ol_2$	$\mathcal{P}ol_3$	no result	$\sum$
AProVE	$\mathcal{P}ol_0$	-	182	-	-	27	209	10	157	-	-	42	209
	$\mathcal{P}ol_1$	-	187	7	-	76	270	-	152	1	-	117	270
	$\mathcal{P}ol_2$	-	32	2	-	83	117	-	35	-	-	82	117
	$\mathcal{P}ol_3$	-	6	-	-	16	22	-	5	-	-	17	22
	no result	-	27	3	1	674	705	-	22	3	-	680	705
	$\sum$	0	434	12	1	876	1323	10	371	4	0	938	1323

So AProVE showed polynomial runtime for 618 of the 1323 examples (47 %). (Note that the collection also contains many examples whose complexity is not polynomial.) In contrast, CaT resp. TCT proved polynomial runtime for 447 (33 %) resp. 385 (29 %) examples. Even a "combined tool" of CaT and TCT (which always returns the better result of these two tools) would only show polynomial runtime for 464 examples (35 %). Hence, our contributions represent a significant advance. This also confirms the results of the *International Termination Competition 2010*, where AProVE won the category of innermost runtime complexity analysis. AProVE also succeeds on Ex. 3, 25, and 37, whereas CaT and TCT fail. (Ex. 22 can be analyzed by all three tools.) For details on our experiments and to run our implementation in AProVE via a web interface, we refer

[16] See `http://www.termination-portal.org/wiki/Termination_Competition`.

to http://aprove.informatik.rwth-aachen.de/eval/RuntimeComplexity/.

Acknowledgments. We are grateful to the CaT and the TCT team for their support with the experiments and to G. Moser and H. Zankl for many helpful comments.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
2. M. Avanzini, G. Moser, and A. Schnabl. Automated implicit computational complexity analysis. In *Proc. IJCAR '08*, LNAI 5195, pages 132–138, 2008.
3. M. Avanzini and G. Moser. Dependency pairs and polynomial path orders. In *Proc. RTA '09*, LNCS 5595, pages 48–62, 2009.
4. M. Avanzini and G. Moser. Closing the gap between runtime complexity and polytime computability. In *Proc. RTA '10*, LIPIcs 6, pages 33–48, 2010.
5. M. Avanzini and G. Moser. Complexity analysis by graph rewriting. In *Proc. FLOPS '10*, LNCS 6009, pages 257–271, 2010.
6. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge U. Pr., 1998.
7. G. Bonfante, A. Cichon, J.-Y. Marion, and H. Touzet. Algorithms with polynomial interpretation termination proof. *J. Functional Programming*, 11(1):33–53, 2001.
8. J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *J. Automated Reasoning*, 40(2-3):195–220, 2008.
9. A. Geser, D. Hofbauer, J. Waldmann, and H. Zantema. On tree automata that certify termination of left-linear term rewriting systems. *Information and Computation*, 205(4):512–534, 2007.
10. J. Giesl, R. Thiemann, P. Schneider-Kamp. The DP framework: Combining techniques for automated termination proofs. *LPAR '04*, LNAI 3452, p. 301-331, 2005.
11. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.
12. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1,2):172–199, 2005.
13. N. Hirokawa and G. Moser. Automated complexity analysis based on the dependency pair method. In *Proc. IJCAR '08*, LNAI 5195, pages 364–379, 2008.
14. N. Hirokawa and G. Moser. Complexity, graphs, and the dependency pair method. In *Proc. LPAR '08*, LNAI 5330, pages 652–666, 2008.
15. D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations. In *Proc. RTA '89*, LNCS 355, pages 167–177, 1989.
16. A. Koprowski and J. Waldmann. Max/plus tree automata for termination of term rewriting. *Acta Cybernetica*, 19(2):357–392, 2009.
17. J.-Y. Marion and R. Péchoux. Characterizations of polynomial complexity classes with a better intensionality. In *Proc. PPDP '08*, pages 79–88. ACM Press, 2008.
18. G. Moser, A. Schnabl, and J. Waldmann. Complexity analysis of term rewriting based on matrix and context dependent interpretations. In *Proc. FSTTCS '08*, LIPIcs 2, pages 304–315, 2008.
19. G. Moser. Personal communication, 2010.
20. F. Neurauter, H. Zankl, and A. Middeldorp. Revisiting matrix interpretations for polynomial derivational complexity of term rewriting. In *Proc. LPAR '10*, LNCS 6397, pages 550–564, 2010.
21. J. Waldmann. Polynomially bounded matrix interpretations. In *Proc. RTA '10*, LIPIcs 6, pages 357–372, 2010.
22. H. Zankl and M. Korp. Modular complexity analysis via relative complexity. In *Proc. RTA '10*, LIPIcs 6, pages 385–400, 2010.

A Proofs

We first state a lemma with useful observations on $\oplus$, $\ominus$, and $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$, which will be used throughout the proofs. Lemma 39 (a) and (b) shows that $\oplus$ and $\ominus$ correspond to the addition and subtraction of functions (where for two functions $f, g : \mathbb{N} \to \mathbb{N}$, we have $(f + g)(n) = f(n) + g(n)$ and $(f - g)(n) = \max(f(n) - g(n), 0)$).

Moreover, the lemma shows the connection between $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$ and the operations $\oplus$ and $\ominus$. For instance, for the m-TRS $\mathcal{R}$ from Ex. 3 and $\mathcal{D} = DT(\mathcal{R})$, we have $\iota_{\langle \mathcal{D},\mathcal{D},\mathcal{R}\rangle} = \mathcal{P}ol_2$. In Ex. 9 we also regarded the set $\mathcal{S}$ which contains all DTs except (6) – (8). We have $\mathrm{rc}_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}(n) \in \mathcal{O}(n)$ and thus, $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} = \mathcal{P}ol_1$. On the other hand, if one counts just the $\mathsf{gt}^\sharp$-DTs (6) – (8), then one again obtains $\mathrm{rc}_{\langle \mathcal{D},\mathcal{D}\setminus\mathcal{S},\mathcal{R}\rangle}(n) \in \mathcal{O}(n^2)$ and thus, $\iota_{\langle \mathcal{D},\mathcal{D}\setminus\mathcal{S},\mathcal{R}\rangle} = \mathcal{P}ol_2$. So in particular, we have $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{D},\mathcal{R}\rangle}$ and $\iota_{\langle \mathcal{D},\mathcal{D},\mathcal{R}\rangle} = \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \oplus \iota_{\langle \mathcal{D},\mathcal{D}\setminus\mathcal{S},\mathcal{R}\rangle}$. These observations are generalized in Lemma 39 (g) and (h).

Lemma 39 (Properties of $\oplus$, $\ominus$, and $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$). *Let f and g be functions from $\mathbb{N}$ to $\mathbb{N} \cup \{\infty\}$ and let $c, d, e \in \mathfrak{C}$.*

(a) $\iota(f) \oplus \iota(g) = \iota(f + g)$
(b) $\iota(f) \ominus \iota(g) \sqsubseteq \iota(f - g)$
(c) $\oplus$ *is associative and commutative*
(d) $c \ominus d \sqsubseteq e$ *iff* $c \sqsubseteq d \oplus e$
(e) $c \ominus d \sqsupseteq e$ *does not imply* $c \sqsupseteq d \oplus e$
(f) $c \sqsupseteq d \oplus e$ *does not imply* $c \ominus d \sqsupseteq e$
(g) If $\mathcal{S}_1 \subseteq \mathcal{S}_2$ *then* $\iota_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}$
(h) For any $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{D}$, *we have* $\iota_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle} \oplus \iota_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle} = \iota_{\langle \mathcal{D},\mathcal{S}_1\cup\mathcal{S}_2,\mathcal{R}\rangle}$
(i) For any $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{D}$, *we have* $\iota_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle} \ominus \iota_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{S}_1\setminus\mathcal{S}_2,\mathcal{R}\rangle}$

Proof. For (a), $\iota(g) \sqsubset \iota(f)$ implies $\iota(f + g) = \iota(f)$ and $\iota(f) \sqsubseteq \iota(g)$ implies $\iota(f + g) = \iota(g)$.

For (b), first let $\iota(f) \sqsubseteq \iota(g)$. Then $\iota(f) \ominus \iota(g) = \mathcal{P}ol_0 \sqsubseteq \iota(f - g)$. If $\iota(g) \sqsubset \iota(f)$ then $\iota(f) \ominus \iota(g) = \iota(f) = \iota(f - g)$.

The claim in (c) is obvious, since the "maximum" function on $\mathfrak{C}$ is associative and commutative.

For (d), if $c \sqsubseteq d$, we have both $c \ominus d = \mathcal{P}ol_0 \sqsubseteq e$ and $c \sqsubseteq d \sqsubseteq d \oplus e$. Otherwise, let $d \sqsubset c$. If $d \sqsubseteq e$, we have $c \ominus d = c \sqsubseteq e$ iff $c \sqsubseteq d \oplus e = e$. If $e \sqsubset d$, then $d \sqsubset c$ implies that $c \ominus d = c \sqsubseteq e$ is false. Similarly, then $c \sqsubseteq d \oplus e = d$ is also false.

For (e), let $c = e = \mathcal{P}ol_0$ and $d = \mathcal{P}ol_1$. Then we have $c \ominus d = \mathcal{P}ol_0 \ominus \mathcal{P}ol_1 = \mathcal{P}ol_0 \sqsupseteq \mathcal{P}ol_0 = e$, but $c = \mathcal{P}ol_0 \not\sqsupseteq \mathcal{P}ol_1 = \mathcal{P}ol_1 \oplus \mathcal{P}ol_0 = d \oplus e$.

For (f), let $c = d = e = \mathcal{P}ol_1$. Then we have $c = \mathcal{P}ol_1 \sqsupseteq \mathcal{P}ol_1 \oplus \mathcal{P}ol_1 = d \oplus e$, but $c \ominus d = \mathcal{P}ol_1 \ominus \mathcal{P}ol_1 = \mathcal{P}ol_0 \not\sqsupseteq \mathcal{P}ol_1 = e$.

For (g), $\mathcal{S}_1 \subseteq \mathcal{S}_2$ implies that $\mathcal{C}plx_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle}(t^\sharp) \leq \mathcal{C}plx_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}(t^\sharp)$ for any $t^\sharp \in \mathcal{T}^\sharp$. This implies $\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle}(n) \leq \mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}(n)$ for all $n \in \mathbb{N}$ and thus, $\iota_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle} = \iota(\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle}) \sqsubseteq \iota(\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}) = \iota_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}$.

For (h), consider an arbitrary $t^\sharp \in \mathcal{T}^\sharp$. Let m be the maximal number of nodes from $\mathcal{S}_1 \cup \mathcal{S}_2$ occurring in any $(\mathcal{D}, \mathcal{R})$-chain tree for $t^\sharp$, i.e., $\mathcal{C}plx_{\langle \mathcal{D},\mathcal{S}_1\cup\mathcal{S}_2,\mathcal{R}\rangle}(t^\sharp) =$

m. Similarly, let m_1 and m_2 be the maximal numbers of nodes from $\mathcal{S}_1$ resp. from $\mathcal{S}_2$ occurring in any $(\mathcal{D},\mathcal{R})$-chain trees for $t^\sharp$, i.e., $\mathcal{C}plx_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle}(t^\sharp) = m_1$ and $\mathcal{C}plx_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}(t^\sharp) = m_2$. When extending "$\leq$" and "$+$" to $\mathbb{N} \cup \{\infty\}$, we clearly have $\sup\{m_1, m_2\} \leq m \leq m_1 + m_2$, i.e., $\sup\{\mathcal{C}plx_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle}(t^\sharp), \mathcal{C}plx_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}(t^\sharp)\}$ $\leq \mathcal{C}plx_{\langle \mathcal{D},\mathcal{S}_1\cup\mathcal{S}_2,\mathcal{R}\rangle}(t^\sharp) \leq \mathcal{C}plx_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle}(t^\sharp) + \mathcal{C}plx_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}(t^\sharp)$. So on the one hand, we have $\sup\{\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle}(n), \mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}(n)\} \leq \mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1\cup\mathcal{S}_2,\mathcal{R}\rangle}(n)$ for all $n \in \mathbb{N}$ which means $\iota_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle} \oplus \iota_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle} = \iota(\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle}) \oplus \iota(\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}) \sqsubseteq \iota(\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1\cup\mathcal{S}_2,\mathcal{R}\rangle}) = \iota_{\langle \mathcal{D},\mathcal{S}_1\cup\mathcal{S}_2,\mathcal{R}\rangle}$. On the other hand, we have $\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1\cup\mathcal{S}_2,\mathcal{R}\rangle}(n) \leq \mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle}(n) + \mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}(n)$ for all $n \in \mathbb{N}$ which means $\iota_{\langle \mathcal{D},\mathcal{S}_1\cup\mathcal{S}_2,\mathcal{R}\rangle} = \iota(\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1\cup\mathcal{S}_2,\mathcal{R}\rangle}) \sqsubseteq \iota(\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle} + \mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}) = \iota(\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle}) \oplus \iota(\mathrm{rc}_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}) = \iota_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle} \oplus \iota_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle}$ by (a).

For (i), we have $\iota_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle} \ominus \iota_{\langle \mathcal{D},\mathcal{S}_2\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{S}_1\setminus\mathcal{S}_2,\mathcal{R}\rangle}$ iff $\iota_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{S}_2,\mathcal{R}\rangle} \oplus \iota_{\langle \mathcal{D},\mathcal{S}_1\setminus\mathcal{S}_2,\mathcal{R}\rangle}$ by (d). But by (b), this is equivalent to $\iota_{\langle \mathcal{D},\mathcal{S}_1,\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{S}_2\cup(\mathcal{S}_1\setminus\mathcal{S}_2),\mathcal{R}\rangle}$. As $\mathcal{S}_2 \cup (\mathcal{S}_1 \setminus \mathcal{S}_2) = \mathcal{S}_1 \cup \mathcal{S}_2$, this is true by (g). □

For any term $t \in \mathcal{T}$, let $t\Downarrow$ denote a *maximal argument normal form* of t, i.e., $t\Downarrow$ is an argument normal form such that[17] $t \xrightarrow{\mathrm{i},>\varepsilon}{}^*_{\mathcal{R}} t\Downarrow$ and such that for all argument normal forms v with $t \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} v$, we have $\mathrm{dl}(t\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \geq \mathrm{dl}(v, \xrightarrow{\mathrm{i}}_{\mathcal{R}})$.

So for a TRS with the rules $\mathsf{a} \to \mathsf{b}, \mathsf{a} \to \mathsf{c}, \mathsf{f}(\mathsf{c}) \to \mathsf{a}$, the term $\mathsf{f}(\mathsf{a})$ has two argument normal forms $\mathsf{f}(\mathsf{b})$ and $\mathsf{f}(\mathsf{c})$. As the derivation length of $\mathsf{f}(\mathsf{b})$ is 0 and the derivation length of $\mathsf{f}(\mathsf{c})$ is 1, we obtain $\mathsf{f}(\mathsf{a})\Downarrow = \mathsf{f}(\mathsf{c})$.

To prove Thm. 10, we first show that the derivation length of a term is bounded by the sum of the derivation lengths of the maximal argument normal forms of its subterms. So to find an upper bound for the (innermost) derivation length of a term $f(t_1, \ldots, t_n)$, one can find bounds for its arguments $t_1, \ldots, t_n$ first, add them up, and finally also add the derivation length of the reduced term $t\Downarrow$ in argument normal form.

Lemma 40 (Derivation Lengths of Subterms). *Let $t \in \mathcal{T}$ and let $\mathcal{R}$ be a TRS where t has no infinite innermost $\mathcal{R}$-reduction. Then*

$$\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \leq \sum_{\pi \in \mathcal{P}os_d(t)} \mathrm{dl}(t|_\pi\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}).$$

If $\mathcal{R}$ is confluent, we even have $\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = \sum_{\pi \in \mathcal{P}os_d(t)} \mathrm{dl}(t|_\pi\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}})$.

Proof. We use induction on $|t|$. For $|t| = 1$, the lemma is obvious as $t\Downarrow = t$. Now let $|t| > 1$ and let the root symbol of t have arity n. Because of the innermost strategy, a rewrite step at the root is only possible after its arguments have been rewritten to normal forms. Thus, we have

$$\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \leq \mathrm{dl}(t\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) + \textstyle\sum_{1\leq i\leq n} \mathrm{dl}(t|_i, \xrightarrow{\mathrm{i}}_{\mathcal{R}}).$$

For confluent rewrite systems, t has a unique argument normal form and hence we have equality here (and in the next equation). The subterms $t|_i$ have a smaller

[17] Here, "$\xrightarrow{\mathrm{i},>\varepsilon}{}^*_{\mathcal{R}}$" denotes innermost reductions below the root position.

size than t and hence the induction hypothesis can be applied:

$$\begin{aligned}\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) &\leq \mathrm{dl}(t\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) + \textstyle\sum_{1\leq i\leq n} \sum_{\pi \in \mathcal{P}os_d(t|_i)} \mathrm{dl}(t|_{i.\pi}\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \\ &= \textstyle\sum_{\pi \in \mathcal{P}os_d(t)} \mathrm{dl}(t|_\pi\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}).\end{aligned}$$

For the last step above, note that if $\varepsilon \notin \mathcal{P}os_d(t)$, then $t\Downarrow$ is a normal form and thus, $\mathrm{dl}(t\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = 0$. □

Using Lemma 40, we can now prove Thm. 10 which shows how dependency tuples can be used to approximate the derivation lengths of terms.

Theorem 10 (*Cplx* bounds Derivation Length). *Let $\mathcal{R}$ be a TRS. Let $t = f(t_1, \ldots, t_n) \in \mathcal{T}$ be in* argument normal form*, i.e., all t_i are normal forms w.r.t. $\mathcal{R}$. Then we have $\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \leq \mathcal{C}plx_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R}\rangle}(t^\sharp)$. If $\mathcal{R}$ is confluent, we even have $\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = \mathcal{C}plx_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R}\rangle}(t^\sharp)$.*

Proof. If t starts an infinite innermost $\mathcal{R}$-reduction (i.e., $\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = \infty$), then there exists an infinite chain starting with $t^\sharp$. The reason is that as t is in argument normal form, the infinite $\mathcal{R}$-reduction of t must begin on the root position. Hence, there is a rule $\ell_1 \to r_1 \in \mathcal{R}$ such that $t = \ell_1\sigma_1$ and such that $r_1\sigma_1$ also starts an infinite innermost $\mathcal{R}$-reduction. Thus, there exists a minimal subterm of $r_1\sigma_1$ with an infinite innermost $\mathcal{R}$-reduction, but where all proper subterms of $r_1\sigma_1$ are innermost terminating. Since σ_1 instantiates all variables with normal forms, this minimal subterm is at a position $\pi_1 \in \mathcal{P}os_d(r_1)$, i.e., the minimal subterm is $r_1|_{\pi_1}\sigma_1$. In the infinite innermost reduction of $r_1|_{\pi_1}\sigma_1$, again all arguments are normalized first, leading to a term t_1 in argument normal form that starts an infinite innermost $\mathcal{R}$-reduction. So the infinite reduction of t_1 must again begin on the root position with some rule $\ell_2 \to r_2 \in \mathcal{R}$. Continuing in this way, one obtains an infinite chain

$$\big(\ell_1^\sharp \to \textsc{Com}_k(\ldots, r_1|^\sharp_{\pi_1}, \ldots) \mid \sigma_1\big), \quad \big(\ell_2^\sharp \to \textsc{Com}_m(\ldots, r_2|^\sharp_{\pi_2}, \ldots) \mid \sigma_2\big), \quad \ldots$$

So there is an infinite chain tree for $\ell_1^\sharp\sigma_1 = t^\sharp$ and hence, $\mathcal{C}plx_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R}\rangle}(t^\sharp) = \infty$.

Now we regard the case where t does not start an infinite innermost $\mathcal{R}$-reduction. Here, we prove the theorem by induction on $\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}})$. If $\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = 0$, then t is in $\mathcal{R}$-normal form. Thus, $t^\sharp$ is in normal form w.r.t. $DT(\mathcal{R}) \cup \mathcal{R}$ and $\mathcal{C}plx_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R}\rangle}(t^\sharp) = 0$.

Otherwise, as the arguments of t are in normal form, there exists a rule $\ell \to r \in \mathcal{R}$ and a substitution σ such that $t = \ell\sigma \xrightarrow{\mathrm{i}}_{\mathcal{R}} r\sigma = u$ and

$$\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = 1 + \mathrm{dl}(u, \xrightarrow{\mathrm{i}}_{\mathcal{R}}). \tag{21}$$

By Lemma 40 we have

$$\mathrm{dl}(u, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \leq \textstyle\sum_{\pi \in \mathcal{P}os_d(u)} \mathrm{dl}(u|_\pi\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \tag{22}$$

(with equality if $\mathcal{R}$ is confluent). As σ instantiates all variables by normal forms, $u|_\pi = r\sigma|_\pi$ is in normal form for all $\pi \in \mathcal{P}os_d(u) \setminus \mathcal{P}os_d(r)$. For such π, this implies $u|_\pi\Downarrow = u|_\pi$ and $\mathrm{dl}(u|_\pi\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = \mathrm{dl}(u|_\pi, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = 0$. Hence, from (22) we obtain

$$\mathrm{dl}(u, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \leq \textstyle\sum_{\pi \in \mathcal{P}os_d(r)} \mathrm{dl}(u|_\pi\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}). \tag{23}$$

Note that $\mathrm{dl}(u|_\pi\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) < \mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}})$ and $u|_\pi\Downarrow$ is in argument normal form. So the induction hypothesis implies

$$\mathrm{dl}(u|_\pi\Downarrow, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \leq \mathcal{C}plx_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R}\rangle}(u|_\pi\Downarrow^\sharp)$$

for all π. Together with (21) and (23) we obtain

$$\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = 1 + \mathrm{dl}(u, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \leq 1 + \sum_{\pi \in \mathcal{P}os_d(r)} \mathcal{C}plx_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R}\rangle}(u|_\pi\Downarrow^\sharp). \tag{24}$$

Let $\mathcal{P}os_d(r) = \{\pi_1, \ldots, \pi_n\}$. Then there exists a chain tree for $t^\sharp$ where $(\ell^\sharp \to \mathrm{C{\small OM}}_n(r|^\sharp_{\pi_1}, \ldots, r|^\sharp_{\pi_n}) \mid \sigma)$ is the root node and where the children of the root node are chain trees for $u|_{\pi_1}\Downarrow^\sharp, \ldots, u|_{\pi_n}\Downarrow^\sharp$. The reason is that $r|_{\pi_j}\sigma = u|_{\pi_j}$ and hence, $r|^\sharp_{\pi_j}\sigma \xrightarrow{\mathrm{i}}^*_{\mathcal{R}} u|_{\pi_j}\Downarrow^\sharp$ for all $j \in \{1, \ldots, n\}$. For confluent $\mathcal{R}$, this chain tree is also a maximal one. Hence, together with (24) we have

$$\begin{aligned} \mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) &\leq 1 + \textstyle\sum_{\pi \in \mathcal{P}os_d(r)} \mathcal{C}plx_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R}\rangle}(u|_\pi\Downarrow^\sharp) \\ &\leq \mathcal{C}plx_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R}\rangle}(t^\sharp) \end{aligned}$$

with "$=$" instead of "$\leq$" for confluent $\mathcal{R}$. □

Theorem 14 (Upper bound for TRS via Canonical DT Problem). *Let $\mathcal{R}$ be a TRS and let $\langle \mathcal{D}, \mathcal{D}, \mathcal{R}\rangle$ be the corresponding canonical DT problem. Then we have $\iota_{\mathcal{R}} \sqsubseteq \iota_{\langle \mathcal{D}, \mathcal{D}, \mathcal{R}\rangle}$ and if $\mathcal{R}$ is confluent, we even have $\iota_{\mathcal{R}} = \iota_{\langle \mathcal{D}, \mathcal{D}, \mathcal{R}\rangle}$.*

Proof. For any $n \in \mathbb{N}$, we have $\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\, \mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \mid t \in \mathcal{T}_B, |t| \leq n \,\} \leq \sup\{\, \mathcal{C}plx_{\langle \mathcal{D}, \mathcal{D}, \mathcal{R}\rangle}(t^\sharp) \mid t \in \mathcal{T}_B, |t| \leq n \,\} = \mathrm{rc}_{\langle \mathcal{D}, \mathcal{D}, \mathcal{R}\rangle}(n)$ by Thm. 10, with equality if $\mathcal{R}$ is confluent. Thus, $\iota_{\mathcal{R}} = \iota(\mathrm{rc}_{\mathcal{R}}) \sqsubseteq \iota(\mathrm{rc}_{\langle \mathcal{D}, \mathcal{D}, \mathcal{R}\rangle}) = \iota_{\langle \mathcal{D}, \mathcal{D}, \mathcal{R}\rangle}$ and if $\mathcal{R}$ is confluent, we even have $\iota_{\mathcal{R}} = \iota_{\langle \mathcal{D}, \mathcal{D}, \mathcal{R}\rangle}$. □

Theorem 17 (Approximating Complexity by Proof Chain). *Let $P_0 \overset{c_1}{\leadsto} P_1 \overset{c_2}{\leadsto} \ldots \overset{c_k}{\leadsto} P_k$ be a proof chain. Then $\iota_{P_0} \sqsubseteq c_1 \oplus \ldots \oplus c_k$.*

Proof. We prove the theorem by induction on the length k of the proof chain. If $k = 0$, then $P_0 = P_k$ is a solved DT problem and hence we have $\iota_{P_0} = \mathcal{P}ol_0$.

Otherwise by the definition of a proof chain, there exists a sound processor P{\small ROC} such that $\mathrm{P{\small ROC}}(P_0) = (c_1, P_1)$. Moreover, $P_1 \overset{c_2}{\leadsto} \ldots \overset{c_k}{\leadsto} P_k$ is also a proof chain and the induction hypothesis implies $\iota_{P_1} \sqsubseteq c_2 \oplus \ldots \oplus c_k$. As PROC is sound, we have $\iota_{P_0} \sqsubseteq c_1 \oplus \iota_{P_1}$. Hence, we obtain $\iota_{P_0} \sqsubseteq c_1 \oplus \ldots c_k$. □

Corollary 18 (Correctness of Approach). *If P_0 is the canonical DT problem for a TRS $\mathcal{R}$ and $P_0 \overset{c_1}{\leadsto} \ldots \overset{c_k}{\leadsto} P_k$ is a proof chain, then $\iota_{\mathcal{R}} \sqsubseteq c_1 \oplus \ldots \oplus c_k$.*

Proof. We have $\iota_{\mathcal{R}} \sqsubseteq \iota_{P_0}$ by Thm. 14 and $\iota_{P_0} \sqsubseteq c_1 \oplus \ldots \oplus c_k$ by Thm. 17. □

Theorem 20 (Usable Rules Processor). *Let $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ be a DT problem. Then the following processor is sound:* $\textsc{Proc}(\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle) = (\mathcal{P}ol_0, \langle \mathcal{D}, \mathcal{S}, \mathcal{U}_\mathcal{R}(\mathcal{D}) \rangle)$.

Proof. Let $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ be a DT problem. For the soundness of this processor we have to prove that $\iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} \sqsubseteq \mathcal{P}ol_0 \oplus \iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{U}_\mathcal{R}(\mathcal{D}) \rangle}$. This is equivalent to $\iota(\mathrm{rc}_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle}) \sqsubseteq \iota(\mathrm{rc}_{\langle \mathcal{D}, \mathcal{S}, \mathcal{U}_\mathcal{R}(\mathcal{D}) \rangle})$. This holds, since for every $\mathcal{S} \subseteq \mathcal{D}$, we have $\mathrm{rc}_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} = \mathrm{rc}_{\langle \mathcal{D}, \mathcal{S}, \mathcal{U}_\mathcal{R}(\mathcal{D}) \rangle}$. The reason is that in a chain tree, variables are always instantiated with normal forms. So (as in the corresponding proofs for usable rules in termination analysis), the only rules applicable to the right-hand side of an instantiated DT are its usable rules. □

Theorem 23 (Reduction Pair Processor). *Let $P = \langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ be a DT problem and $(\succsim, \succ)$ be a* Com*-monotonic reduction pair. Let $\mathcal{D} \subseteq {\succsim} \cup {\succ}$, $\mathcal{R} \subseteq {\succsim}$, and $c \sqsupseteq \iota(\mathrm{rc}_\succ)$ for the function $\mathrm{rc}_\succ(n) = \sup\{ \mathrm{dl}(t^\sharp, \succ) \mid t \in \mathcal{T}_B, |t| \leq n\}$. Then the following processor is sound:* $\textsc{Proc}(\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle) = (c, \langle \mathcal{D}, \mathcal{S} \setminus \mathcal{D}_\succ, \mathcal{R} \rangle)$.

Proof. To prove soundness, we need to show that $\iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} \sqsubseteq c \oplus \iota_{\langle \mathcal{D}, \mathcal{S} \setminus \mathcal{D}_\succ, \mathcal{R} \rangle}$ holds. This follows from Lemma 39, if we can show $\iota_{\langle \mathcal{D}, \mathcal{D}_\succ, \mathcal{R} \rangle} \sqsubseteq c$:

$$\begin{aligned} \iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} &\sqsubseteq \iota_{\langle \mathcal{D}, \mathcal{S} \cup \mathcal{D}_\succ, \mathcal{R} \rangle} && \text{by Lemma 39(g)} \\ &= \iota_{\langle \mathcal{D}, \mathcal{D}_\succ, \mathcal{R} \rangle} \oplus \iota_{\langle \mathcal{D}, \mathcal{S} \setminus \mathcal{D}_\succ, \mathcal{R} \rangle} && \text{by Lemma 39(h)} \\ &\sqsubseteq c \oplus \iota_{\langle \mathcal{D}, \mathcal{S} \setminus \mathcal{D}_\succ, \mathcal{R} \rangle} \end{aligned}$$

As we have $\iota(\mathrm{rc}_\succ) \sqsubseteq c$, it suffices to show $\iota_{\langle \mathcal{D}, \mathcal{D}_\succ, \mathcal{R} \rangle} \sqsubseteq \iota(\mathrm{rc}_\succ)$. Let $s \in \mathcal{T}_B$ be a basic term and consider an arbitrary innermost $(\mathcal{D} \cup \mathcal{R})$-reduction sequence starting with $s^\sharp$. All terms in such a reduction sequence are of the form $C[t_1^\sharp, \dots, t_n^\sharp]$ for a context C consisting only of compound symbols and where $t_1^\sharp, \dots, t_n^\sharp$ are sharped terms from $\mathcal{T}^\sharp$. As $\succ$ is Com-monotonic, all $\mathcal{D}$-steps in such a reduction sequence take place on monotonic positions.

So if $u \xrightarrow{\mathrm{i}}_{\mathcal{D}_\succ} v$ is a rewrite step in an innermost $(\mathcal{D} \cup \mathcal{R})$-reduction of $s^\sharp$, then $u \succ v$. On the other hand, $\succsim$ is monotonic, too. Hence, $u \to_{\mathcal{D}_\succsim \cup \mathcal{R}} v$ implies $u \succsim v$, where $\mathcal{D}_\succsim$ are those DTs from $\mathcal{D}$ which are weakly decreasing. Now let

$$s^\sharp \;=\; s_0 \;\xrightarrow{\mathrm{i}}_{\nu_0}\; t_0 \;\xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}}\; s_1 \;\xrightarrow{\mathrm{i}}_{\nu_1}\; t_1 \;\xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}}\; s_2 \;\dots$$

be a (finite or infinite) innermost $(\mathcal{D} \cup \mathcal{R})$-reduction, where $\nu_i \in \mathcal{D}$ for all i. Then

$$s^\sharp = s_0 \cdot\!\!\succ_0 t_0 \succsim s_1 \cdot\!\!\succ_1 t_1 \succsim s_2 \;\dots$$

holds. Here "$\cdot\!\!\succ_i$" is "$\succ$" if $\nu_i \in \mathcal{D}_\succ$ and "$\succsim$" else. Let $n_1 < n_2 < \dots$ be the sequence of indexes where $\cdot\!\!\succ_{n_j} = \succ$. For each n_j we have $s_{n_j} \succ t_{n_j}$. As ${\succsim} \circ {\succ} \circ {\succsim} \subseteq {\succ}$, we obtain $s^\sharp \succ t_{n_1} \succ t_{n_2} \succ \dots$ and therefore $\mathrm{dl}(s^\sharp, \succ) > \mathrm{dl}(t_{n_1}^\sharp, \succ) > \mathrm{dl}(t_{n_2}^\sharp, \succ) > \dots$ or $\mathrm{dl}(s^\sharp, \succ) = \infty$.

Hence $\mathrm{rc}_\succ(|s|)$ is an upper bound for the number of $\mathcal{D}_\succ$-steps in any innermost $(\mathcal{D} \cup \mathcal{R})$-reduction of $s^\sharp$. Moreover, $\mathcal{C}plx_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle}(s^\sharp)$ is the maximal number of $\mathcal{S}$-steps in any innermost $(\mathcal{D} \cup \mathcal{R})$-reduction of $s^\sharp$. Hence, $\mathcal{C}plx_{\langle \mathcal{D}, \mathcal{D}_\succ, \mathcal{R} \rangle}(s^\sharp) \leq \mathrm{rc}_\succ(|s|)$ for all $s \in \mathcal{T}_B$. This implies $\mathrm{rc}_{\langle \mathcal{D}, \mathcal{D}_\succ, \mathcal{R} \rangle}(n) \leq \mathrm{rc}_\succ(n)$ for all n and hence, $\iota_{\langle \mathcal{D}, \mathcal{D}_\succ, \mathcal{R} \rangle} = \iota(\mathrm{rc}_{\langle \mathcal{D}, \mathcal{D}_\succ, \mathcal{R} \rangle}) \sqsubseteq \iota(\mathrm{rc}_\succ)$. □

Theorem 24 (Reduction Pair Processor with Polynomial Interpretations). *Let $P = \langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ be a DT problem and let $\succsim$ and $\succ$ be induced by a complexity polynomial interpretation $[\cdot]$. Let $m \in \mathbb{N}$ be the maximal degree of all polynomials $[f^\sharp]$, for all $f^\sharp$ with $f \in \Sigma_d$. Let $\mathcal{D} \subseteq {\succsim} \cup {\succ}$ and $\mathcal{R} \subseteq {\succsim}$. Then the following processor is sound:* $\textsc{Proc}(\, \langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle \,) = (\mathcal{P}ol_m,\ \langle \mathcal{D},\, \mathcal{S} \setminus \mathcal{D}_\succ,\, \mathcal{R} \rangle)$.

Proof. Complexity polynomial interpretations are obviously $\textsc{Com}$-monotonic. Hence, it remains to prove that $\mathcal{P}ol_m \sqsupseteq \iota(\mathrm{rc}_\succ)$ holds. Recall that $\mathrm{rc}_\succ(n) = \sup\{\mathrm{dl}(t^\sharp, \succ) \mid t \in \mathcal{T}_B \text{ and } |t| \le n\}$. Let $[\cdot]_0$ be a variant of the polynomial interpretation which maps every variable to 0. Then we have $\mathrm{dl}(t, \succ) \le [t]_0$ for all terms t. Thus,

$$\mathrm{rc}_\succ(n) \le \sup\{[t^\sharp]_0 \mid t \in \mathcal{T}_B \text{ and } |t| \le n\}. \tag{25}$$

Let b_{max} be the maximum of all $[f](0, \ldots, 0)$, for all constructors $f \in \Sigma \setminus \Sigma_d$. Then for every term s containing only constructors and variables, we obtain $[s]_0 \le b_{max} \cdot |s|$, where $|s|$ is again the size of s. Hence, there exists a number $k \in \mathbb{N}$ such that for all $t \in \mathcal{T}_B$ we have

$$[t^\sharp]_0 \le k \cdot [f^\sharp](|t|, \ldots, |t|), \quad \text{where } f^\sharp = \mathrm{root}(t^\sharp).$$

To see this, note that for $t = f(t_1, \ldots, t_n) \in \mathcal{T}_B$ we have

$$\begin{aligned}
[t^\sharp]_0 &= [f^\sharp]([t_1]_0, \ldots, [t_n]_0) \\
&\le [f^\sharp](b_{max} \cdot |t_1|, \ldots, b_{max} \cdot |t_n|) \\
&\le [f^\sharp](b_{max} \cdot |t|, \ldots, b_{max} \cdot |t|) \\
&\le b_{max}^m \cdot [f^\sharp](|t|, \ldots, |t|), \quad \text{where } m \text{ is the degree of } [f^\sharp] \\
&\le k \cdot [f^\sharp](|t|, \ldots, |t|), \quad \text{where } k = b_{max}^d \text{ and } d \text{ is the maximum degree of all } [g^\sharp], \text{ for all sharped symbols } g^\sharp
\end{aligned} \tag{26}$$

Hence,

$$\begin{aligned}
\mathrm{rc}_\succ(n) &\le \sup\{[t^\sharp]_0 \mid t \in \mathcal{T}_B \text{ and } |t| \le n\} && \text{by (25)} \\
&\le k \cdot [f^\sharp](n, \ldots, n) && \text{by (26).}
\end{aligned}$$

Since the polynomials $[f^\sharp]$ have at most degree m, we have $\iota(\mathrm{rc}_\succ) \sqsubseteq \mathcal{P}ol_m$. □

Theorem 28 (Leaf Removal Processor). *Let $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ be a DT problem and let $s \to t \in \mathcal{D}$ be a leaf in the $(\mathcal{D}, \mathcal{R})$-dependency graph. Then the following processor is sound:* $\textsc{Proc}(\, \langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle \,) = (\mathcal{P}ol_0, \langle \mathcal{D} \setminus \{s \to t\}, \mathcal{S} \setminus \{s \to t\}, \mathcal{R} \rangle)$.

Proof. Let k be the maximal index of compound symbols $\textsc{Com}_k$ occurring in $\mathcal{D}$. Hence, a chain tree with m inner (i.e., non-leaf) nodes can have at most $1 + k \cdot m$ leaves. So for any term t, $\mathcal{C}plx_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle}(t^\sharp) \le 1 + k \cdot \mathcal{C}plx_{\langle \mathcal{D} \setminus \{s \to t\}, \mathcal{S} \setminus \{s \to t\}, \mathcal{R} \rangle}(t^\sharp)$ and thus $\mathrm{rc}_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle}(n) \le 1 + k \cdot \mathrm{rc}_{\langle \mathcal{D} \setminus \{s \to t\}, \mathcal{S} \setminus \{s \to t\}, \mathcal{R} \rangle}$. This implies that the complexity does not change when removing the leaves from chain trees, i.e., $\iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} = \iota_{\langle \mathcal{D} \setminus \{s \to t\}, \mathcal{S} \setminus \{s \to t\}, \mathcal{R} \rangle} = \mathcal{P}ol_0 \oplus \iota_{\langle \mathcal{D} \setminus \{s \to t\}, \mathcal{S} \setminus \{s \to t\}, \mathcal{R} \rangle}$, which implies the soundness of the leaf removal processor. □

In the following, for any set of DTs $\mathcal{M}$, let $|T|_\mathcal{M}$ be the number of nodes in a chain tree T which are marked with DTs from $\mathcal{M}$.

Lemma 29 (Complexity Bounded by Predecessors). *Let* $\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle$ *be a DT problem and* $s \to t \in \mathcal{D}$*. Let* $Pre(s \to t) \subseteq \mathcal{D}$ *be the predecessors of* $s \to t$*, i.e.,* $Pre(s \to t)$ *contains all DTs* $u \to v$ *where there is an edge from* $u \to v$ *to* $s \to t$ *in the* $(\mathcal{D}, \mathcal{R})$*-dependency graph. Then* $\iota_{\langle \mathcal{D}, \{s \to t\}, \mathcal{R} \rangle} \sqsubseteq \iota_{\langle \mathcal{D}, Pre(s \to t), \mathcal{R} \rangle}$.

Proof. Let k be the maximal index of the compound symbols $\textsc{Com}_k$ occurring in $Pre(s \to t)$ and let T be a $(\mathcal{D}, \mathcal{R})$-chain tree. We show that $|T|_{\{s \to t\}} \leq 1 + k \cdot |T|_{Pre(s \to t)}$.

Any node of T labeled with $s \to t$ is either the root node or a child of a node labeled with a DT from $Pre(s \to t)$. As every node labeled with a DT from $Pre(s \to t)$ has at most k children (since every chain corresponds to a path in the dependency graph), we obtain $|T|_{\{s \to t\}} \leq 1 + k \cdot |T|_{Pre(s \to t)}$.

Note that this holds for *any* $(\mathcal{D}, \mathcal{R})$-chain tree T. This implies

$$\mathcal{C}plx_{\langle \mathcal{D}, \{s \to t\}, \mathcal{R} \rangle}(t^\sharp) \leq 1 + k \cdot \mathcal{C}plx_{\langle \mathcal{D}, Pre(s \to t), \mathcal{R} \rangle}(t^\sharp)$$

for any term $t^\sharp \in \mathcal{T}^\sharp$. Thus, $\mathrm{rc}_{\langle \mathcal{D}, \{s \to t\}, \mathcal{R} \rangle}(n) \leq 1 + k \cdot \mathrm{rc}_{\langle \mathcal{D}, Pre(s \to t), \mathcal{R} \rangle}(n)$ for all n and hence $\iota_{\langle \mathcal{D}, \{s \to t\}, \mathcal{R} \rangle} \sqsubseteq \iota_{\langle \mathcal{D}, Pre(s \to t), \mathcal{R} \rangle}$. ☐

Corollary 32 (Correctness). *If* P_0 *is the canonical extended DT problem for a TRS* $\mathcal{R}$ *and* $P_0 \overset{c_1}{\rightsquigarrow} \ldots \overset{c_k}{\rightsquigarrow} P_k$ *is a proof chain, then* $\iota_{\mathcal{R}} = \gamma_{P_0} \sqsubseteq c_1 \oplus \ldots \oplus c_k$.

Proof. We have $\iota_{\mathcal{R}} = \iota_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R} \rangle}$ by Thm. 14. Moreover, $\iota_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \mathcal{R} \rangle} = \gamma_{\langle DT(\mathcal{R}), DT(\mathcal{R}), \varnothing, \mathcal{R} \rangle} = \gamma_{P_0}$. The proof for $\gamma_{P_0} \sqsubseteq c_1 \oplus \ldots \oplus c_k$ is completely analogous to the proof of Thm. 17. ☐

Theorem 33 (Knowledge Propagation Processor). *Let* $\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle$ *be an extended DT problem,* $s \to t \in \mathcal{S}$*, and* $Pre(s \to t) \subseteq \mathcal{K}$*. Then the following processor is sound:* $\textsc{Proc}(\, \langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle \,) = (\mathcal{P}ol_0,\, \langle \mathcal{D},\, \mathcal{S} \setminus \{s \to t\},\, \mathcal{K} \cup \{s \to t\},\, \mathcal{R} \rangle\,)$.

Proof. We have to show that $\gamma_{\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle} \sqsubseteq \mathcal{P}ol_0 \oplus \gamma_{\langle \mathcal{D}, \mathcal{S} \setminus \{s \to t\}, \mathcal{K} \cup \{s \to t\}, \mathcal{R} \rangle}$, i.e., $\gamma_{\langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle} \sqsubseteq \gamma_{\langle \mathcal{D}, \mathcal{S} \setminus \{s \to t\}, \mathcal{K} \cup \{s \to t\}, \mathcal{R} \rangle}$. By the definition of γ, this is equivalent to

$$\iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} \ominus \iota_{\langle \mathcal{D}, \mathcal{K}, \mathcal{R} \rangle} \sqsubseteq \iota_{\langle \mathcal{D}, \mathcal{S} \setminus \{s \to t\}, \mathcal{R} \rangle} \ominus \iota_{\langle \mathcal{D}, \mathcal{K} \cup \{s \to t\}, \mathcal{R} \rangle}. \tag{27}$$

From Lemma 29 and Lemma 39(g), we have $\iota_{\langle \mathcal{D}, \{s \to t\}, \mathcal{R} \rangle} \sqsubseteq \iota_{\langle \mathcal{D}, Pre(\{s \to t\}), \mathcal{R} \rangle} \sqsubseteq \iota_{\langle \mathcal{D}, \mathcal{K}, \mathcal{R} \rangle}$. Hence, Lemma 39(h) implies $\iota_{\langle \mathcal{D}, \mathcal{K} \cup \{s \to t\}, \mathcal{R} \rangle} = \iota_{\langle \mathcal{D}, \mathcal{K}, \mathcal{R} \rangle} \oplus \iota_{\langle \mathcal{D}, \{s \to t\}, \mathcal{R} \rangle} = \iota_{\langle \mathcal{D}, \mathcal{K}, \mathcal{R} \rangle}$. Thus for (27), it suffices to show

$$\iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} \ominus \iota_{\langle \mathcal{D}, \mathcal{K} \cup \{s \to t\}, \mathcal{R} \rangle} \sqsubseteq \iota_{\langle \mathcal{D}, \mathcal{S} \setminus \{s \to t\}, \mathcal{R} \rangle} \ominus \iota_{\langle \mathcal{D}, \mathcal{K} \cup \{s \to t\}, \mathcal{R} \rangle}. \tag{28}$$

To this end, we consider two cases: If $\iota_{\langle \mathcal{D}, \{s \to t\}, \mathcal{R} \rangle} \sqsubset \iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle}$ holds, we have $\iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} = \iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} \ominus \iota_{\langle \mathcal{D}, \{s \to t\}, \mathcal{R} \rangle} \sqsubseteq \iota_{\langle \mathcal{D}, \mathcal{S} \setminus \{s \to t\}, \mathcal{R} \rangle}$ by Lemma 39(i). Otherwise, we obtain $\iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} \sqsubseteq \iota_{\langle \mathcal{D}, \{s \to t\}, \mathcal{R} \rangle} \sqsubseteq \iota_{\langle \mathcal{D}, \mathcal{K} \cup \{s \to t\}, \mathcal{R} \rangle}$ by Lemma 39(g) and thus $\iota_{\langle \mathcal{D}, \mathcal{S}, \mathcal{R} \rangle} \ominus \iota_{\langle \mathcal{D}, \mathcal{K} \cup \{s \to t\}, \mathcal{R} \rangle} = \mathcal{P}ol_0$. In both cases, the required inequality (28) follows. ☐

Theorem 34 (Processors for Extended DT Problems). *Let* $P = \langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R} \rangle$ *be an extended DT problem. Then the following processors are sound.*

- *The usable rules processor:* $\text{PROC}(P) = (\mathcal{P}ol_0, \langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{U}_\mathcal{R}(\mathcal{D})\rangle)$.
- *The leaf removal processor* $\text{PROC}(P) = (\mathcal{P}ol_0, \langle \mathcal{D} \setminus \{s \to t\}, \mathcal{S} \setminus \{s \to t\}, \mathcal{K} \setminus \{s \to t\}, \mathcal{R}\rangle)$*, if* $s \to t$ *is a leaf in the* $(\mathcal{D}, \mathcal{R})$*-dependency graph.*
- *The reduction pair processor:* $\text{PROC}(P) = (c,\ \langle \mathcal{D}, \mathcal{S} \setminus \mathcal{D}_\succ, \mathcal{K} \cup \mathcal{D}_\succ, \mathcal{R}\rangle)$*, if* $(\succsim, \succ)$ *is a* COM*-monotonic reduction pair,* $\mathcal{D} \subseteq {\succ} \cup {\succsim}$*,* $\mathcal{R} \subseteq {\succsim}$*, and* $c \sqsupseteq \iota(\text{rc}_\succ)$ *for the function* $\text{rc}_\succ(n) = \sup\{\text{dl}(t^\sharp, \succ) \mid t \in \mathcal{T}_B, |t| \leq n\}$.

Proof. The soundness of the usable rules processor follows since $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} = \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{U}_\mathcal{R}(\mathcal{D})\rangle}$ and $\iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle} = \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{U}_\mathcal{R}(\mathcal{D})\rangle}$, as in Thm. 20. Thus, $\gamma_{\langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{R}\rangle} = \gamma_{\langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{U}_\mathcal{R}(\mathcal{D})\rangle}$.

Similarly, the soundness of the leaf removal processor holds since $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} = \iota_{\langle \mathcal{D}\setminus\{s\to t\},\mathcal{S}\setminus\{s\to t\},\mathcal{R}\rangle}$ and $\iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle} = \iota_{\langle \mathcal{D}\setminus\{s\to t\},\mathcal{K}\setminus\{s\to t\},\mathcal{R}\rangle}$, as in Thm. 28. Hence, $\gamma_{\langle \mathcal{D},\mathcal{S},\mathcal{K},\mathcal{R}\rangle} = \gamma_{\langle \mathcal{D}\setminus\{s\to t\},\mathcal{S}\setminus\{s\to t\},\mathcal{K}\setminus\{s\to t\},\mathcal{R}\rangle}$.

For the soundness of the reduction pair processor, we have to show $\gamma_P \sqsubseteq c \oplus \gamma_{\langle \mathcal{D},\mathcal{S}\setminus\mathcal{D}_\succ,\mathcal{K}\cup\mathcal{D}_\succ,\mathcal{R}\rangle}$. If we have $\gamma_P \sqsubseteq c$, then this is obviously true. Hence, we consider $c \sqsubset \gamma_P$. Now we have to show $\gamma_P \sqsubseteq \gamma_{\langle \mathcal{D},\mathcal{S}\setminus\mathcal{D}_\succ,\mathcal{K}\cup\mathcal{D}_\succ,\mathcal{R}\rangle}$. By the definition of γ, this means

$$\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \ominus \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{S}\setminus\mathcal{D}_\succ,\mathcal{R}\rangle} \ominus \iota_{\langle \mathcal{D},\mathcal{K}\cup\mathcal{D}_\succ,\mathcal{R}\rangle}. \tag{29}$$

To show (29), we prove (i) $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{S}\setminus\mathcal{D}_\succ,\mathcal{R}\rangle}$ and (ii) $\iota_{\langle \mathcal{D},\mathcal{K}\cup\mathcal{D}_\succ,\mathcal{R}\rangle} \sqsubset \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$. Then (29) follows by the definition of $\ominus$.

We first show (i). As $c \sqsubset \gamma_P$ implies $\gamma_P \neq \mathcal{P}ol_0$, we have $\gamma_P = \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$ and therefore $c \sqsubset \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$. Moreover, from the proof of Thm. 23 we have $\iota_{\langle \mathcal{D},\mathcal{D}_\succ,\mathcal{R}\rangle} \sqsubseteq \iota(\text{rc}_\succ) \sqsubseteq c$. Hence (i) holds, using Lemma 39(i) for the last inequality:

$$\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} = \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \ominus c \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \ominus \iota_{\langle \mathcal{D},\mathcal{D}_\succ,\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{S}\setminus\mathcal{D}_\succ,\mathcal{R}\rangle}$$

Now we show (ii). From Lemma 39(h) and $\iota_{\langle \mathcal{D},\mathcal{D}_\succ,\mathcal{R}\rangle} \sqsubseteq c$ we have

$$\iota_{\langle \mathcal{D},\mathcal{K}\cup\mathcal{D}_\succ,\mathcal{R}\rangle} = \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle} \oplus \iota_{\langle \mathcal{D},\mathcal{D}_\succ,\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle} \oplus c. \tag{30}$$

Note that $\gamma_P \neq \mathcal{P}ol_0$ implies $\iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle} \sqsubset \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$. Together with $c \sqsubset \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$ this implies $\iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle} \oplus c \sqsubset \iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}$ and hence (ii) follows with (30). ◻

Theorem 36 (Narrowing Processor). *Let* $P = \langle \mathcal{D}, \mathcal{S}, \mathcal{K}, \mathcal{R}\rangle$ *be an extended DT problem and let* $s \to t \in \mathcal{D}$ *with* $t = \text{COM}_n(t_1, \ldots, t_i, \ldots, t_n)$*. Let* $\mu_1, \ldots, \mu_d$ *be the narrowing substitutions of* t_i *with the corresponding narrowing results* $w_1, \ldots, w_d$*, where* $d \geq 0$*. Let* $t_{k_1}, \ldots, t_{k_m}$ *be the terms from* $t_1, \ldots, t_n$ *that are not captured by* $\mu_1, \ldots, \mu_d$*, where* $k_1, \ldots, k_m$ *are pairwise different. We define*

$$\begin{aligned} \mathcal{M} = \ & \{s\mu_j \to \text{COM}_n(t_1\mu_j,\ \ldots,\ t_{i-1}\mu_j,\ w_j,\ t_{i+1}\mu_j,\ \ldots,\ t_n\mu_j) \mid 1 \leq j \leq d\} \\ & \cup \{s \quad \to \text{COM}_m(t_{k_1}, \ldots, t_{k_m})\}. \end{aligned}$$

Then the following processor is sound: $\text{PROC}(P) = (\mathcal{P}ol_0, \langle \mathcal{D}', \mathcal{S}', \mathcal{K}', \mathcal{R}\rangle)$*, where* $\mathcal{D}' = \mathcal{D}[s \to t \ /\ \mathcal{M}]$ *and* $\mathcal{S}' = \mathcal{S}[s \to t\ /\ \mathcal{M}]$*.* $\mathcal{K}'$ *results from* $\mathcal{K}$ *by removing* $s \to t$ *and all DTs that are reachable from* $s \to t$ *in the* $(\mathcal{D}, \mathcal{R})$*-dependency graph.*

Proof. W.l.o.g. let $\mathcal{M}$ and $\mathcal{D}$ be disjoint (otherwise, we apply a variable renaming on one of them). Given a $(\mathcal{D},\mathcal{R})$-chain tree T, we construct a $(\mathcal{D}',\mathcal{R})$-chain tree T' by repeatedly replacing every node of the form $(s \to t \mid \sigma)$ by a new node of the form $(s\mu \to t' \mid \sigma')$ with $s\mu \to t' \in \mathcal{M}$. This implies $|T|_{\{s\to t\}} = |T'|_{\mathcal{M}}$ and for any DT $u \to v \notin \{s \to t\} \cup \mathcal{M}$, we have $|T|_{\{u\to v\}} = |T'|_{\{u\to v\}}$. However, we have to show the following two statements in order to ensure that we still obtain a chain tree:

(A) *Relation to predecessor:* If $(s \to t \mid \sigma)$ was the root node of the chain tree for $s\sigma$, then the new node should also be the root note of a chain tree for $s\sigma$, i.e., we need $s\sigma = s\mu\sigma'$.
Otherwise, if $(s \to t \mid \sigma)$ had a predecessor $(p \to \textsc{Com}_k(q_1,\ldots,q_k) \mid \rho)$ with $q_j\rho \xrightarrow{\mathsf{i}}{}^*_{\mathcal{R}} s\sigma$, then the same relation should also hold for the new node $(s\mu \to t' \mid \sigma')$, i.e., we need $q_j\rho \xrightarrow{\mathsf{i}}{}^*_{\mathcal{R}} s\mu\sigma'$. Note that this is obviously fulfilled if $s\sigma = s\mu\sigma'$.

(B) *Relation to successors:* Let $(s \to t \mid \sigma)$ have the children labeled with $(u_1 \to v_1|\tau_1),\ldots,(u_e \to v_e|\tau_e)$ for $e \geq 0$. Hence, there exist pairwise disjoint $i_1,\ldots,i_e \in \{1,\ldots,n\}$ such that $t_{i_j}\sigma \xrightarrow{\mathsf{i}}{}^*_{\mathcal{R}} u_j\tau_j$ for all $1 \leq j \leq e$. When replacing $(s \to t \mid \sigma)$ by a new node $(s\mu \to t' \mid \sigma')$ with $s\mu \to t' \in \mathcal{M}$, we have to show that there exist pairwise different indexes $i'_1,\ldots,i'_e$ such that $t'|_{i'_j}\sigma' \xrightarrow{\mathsf{i}}{}^*_{\mathcal{R}} u_j\tau_j$ for all $1 \leq j \leq e$. Note that this is obviously fulfilled if for all j we have $t'|_{i'_j}\sigma' = t_{i_j}\sigma$.

We now distinguish three cases. For each of them, we show how to choose the new node $(s\mu \to t' \mid \sigma')$ such that the relations to the predecessor and to the successors in (A) and (B) still hold.

- Case 1: none of the terms $t_{i_1},\ldots,t_{i_e}$ is captured by $\mu_1,\ldots,\mu_d$.
  Hence, $\{i_1,\ldots,i_e\} \subseteq \{k_1,\ldots,k_m\}$. We choose $s\mu \to t'$ to be $s \to \textsc{Com}_m(t_{k_1},\ldots,t_{k_m})$ (i.e., μ is the identity) and we choose $\sigma' = \sigma$. This implies $s\sigma = s\mu\sigma'$ and thus, (A) holds. Moreover for every i_j, there exists an i'_j with $\textsc{Com}_m(t_{k_1},\ldots,t_{k_m})|_{i'_j} = t_{i_j}$, since $i_j \in \{k_1,\ldots,k_m\}$. Thus, $t'|_{i'_j}\sigma' = \textsc{Com}_m(t_{k_1},\ldots,t_{k_m})|_{i'_j}\sigma = t_{i_j}\sigma$, which proves (B).

- Case 2: $i \in \{i_1,\ldots,i_e\}$.
  Thus, there is a $1 \leq j_0 \leq e$ with $i = i_{j_0}$. Hence, $t_i\sigma = t_{i_{j_0}}\sigma \xrightarrow{\mathsf{i}}{}^*_{\mathcal{R}} u_{j_0}\tau_{j_0}$.
  First regard the case where this reduction works in zero steps, i.e., $t_i\sigma = u_{j_0}\tau_{j_0}$. W.l.o.g., we can assume that u_{j_0} is variable-disjoint from t_i. Then t_i unifies with u_{j_0} using some mgu μ where $\sigma = \mu\sigma'$ and $\tau_{j_0} = \mu\tau'_{j_0}$ for some substitutions σ' and τ'_{j_0}. Since $(s \to t \mid \sigma)$ and $(u_{j_0} \to v_{j_0} \mid \tau_{j_0})$ are nodes in a chain tree, both $s\sigma$ and $u_{j_0}\tau_{j_0}$ are in $\mathcal{R}$-normal form. This implies that $s\mu$ and $u_{j_0}\mu$ are also in $\mathcal{R}$-normal form. Hence, t_i has the narrowing substitution μ with corresponding result $t_i\mu$. Thus, $s\mu \to t\mu \in \mathcal{M}$ and we can replace the node $(s \to t \mid \sigma)$ by $(s\mu \to t\mu \mid \sigma')$. For (A), we have $s\mu\sigma' = s\sigma$. For (B), we let $i'_j = i_j$ for all $1 \leq j \leq e$. Then we obtain $t'|_{i'_j}\sigma' = t'|_{i_j}\sigma' = t|_{i_j}\mu\sigma' = t|_{i_j}\sigma$, which implies (B).

Otherwise, the reduction $t_i\sigma \xrightarrow{\mathrm{i}}^*_{\mathcal{R}} u_{j_0}\tau_{j_0}$ takes at least one step. Let π be the position of $t_i\sigma$ where the first reduction step takes place. We have $\pi \in \mathcal{P}os(t_i)$ and $t_i|_\pi \notin \mathcal{V}$, since the reduction cannot be "in σ". The reason is that otherwise, $s\sigma$ would not be an $\mathcal{R}$-normal form, due to $\mathcal{V}(t_i) \subseteq \mathcal{V}(s)$. Thus, there exists a rule $\ell \to r \in \mathcal{R}$ which matches $t_i|_\pi\sigma$. W.l.o.g., we can assume that ℓ is variable-disjoint to t_i. Then we can extend σ to the variables of ℓ such that $t_i|_\pi\sigma = \ell\sigma$ and

$$t_i\sigma \;=\; t_i[\ell]_\pi\sigma \;\xrightarrow{\mathrm{i}}_{\mathcal{R}}\; t_i[r]_\pi\sigma \xrightarrow{\mathrm{i}}^*_{\mathcal{R}} u_{j_0}\tau_{j_0}. \tag{31}$$

Since σ is a unifier of $t_i|_\pi$ and ℓ, they also have an mgu μ with $\sigma = \mu\sigma'$ for some substitution σ'. Moreover, since $s\sigma$ is in $\mathcal{R}$-normal form, $s\mu$ is in $\mathcal{R}$-normal form as well. Hence, μ is a narrowing substitution of t_i and the corresponding narrowing result is $t_i[r]_\pi\mu$.
Let $t' = \textsc{Com}_n(t_1, \ldots, t_{i-1}, t_i[r]_\pi, t_{i+1}, \ldots, t_n)\mu$. Then $s\mu \to t' \in \mathcal{M}$ and we replace the node $(s \to t \mid \sigma)$ by $(s\mu \to t' \mid \sigma')$. It remains to show that (A) and (B) hold.
(A) is satisfied since $\sigma = \mu\sigma'$ and hence, $s\sigma = s\mu\sigma'$. For (B), we let $i'_j = i_j$ for all $1 \le j \le e$. For j_0, we now obtain

$$t'|_{i'_{j_0}}\sigma' \;=\; t'|_{i_{j_0}}\sigma' \;=\; t_i[r]_\pi\mu\sigma' \;=\; t_i[r]_\pi\sigma \xrightarrow{\mathrm{i}}^*_{\mathcal{R}} u_{j_0}\tau_{j_0}$$

by (31). For $j \neq j_0$, we have $t'|_{i'_j}\sigma' = t'|_{i_j}\sigma' = t|_{i_j}\mu\sigma' = t|_{i_j}\sigma$, which implies (B).

- Case 3: $i \notin \{i_1, \ldots, i_e\}$ and a term from $t_{i_1}, ..., t_{i_e}$ is captured by $\mu_1, ..., \mu_d$.
  Let $1 \le j_0 \le e$ such that $t_{i_{j_0}}$ is captured by $\mu_1, \ldots, \mu_d$. Hence, $t_{i_{j_0}}\sigma \xrightarrow{\mathrm{i}}^*_{\mathcal{R}} u_{j_0}\tau_{j_0}$. As in Case 2, this implies that there exists a narrowing substitution μ of $t_{i_{j_0}}$ with $\sigma = \mu\tilde{\sigma}$ for some substitution $\tilde{\sigma}$. Since $t_{i_{j_0}}$ is captured by $\mu_1, \ldots, \mu_d$, there is a $1 \le j_1 \le d$ where μ_{j_1} is more general than μ, i.e., $\mu = \mu_{j_1}\overline{\sigma}$ for some substitution $\overline{\sigma}$. We define $\sigma' = \overline{\sigma}\tilde{\sigma}$ which implies $\sigma = \mu_{j_1}\sigma'$. Now we replace $(s \to t \mid \sigma)$ by $(s\mu_{j_1} \to t' \mid \sigma')$ where $t' = \textsc{Com}_n(t_1\mu_{j_1}, \ldots, t_{i-1}\mu_{j_1}, w_{j_1}, t_{i+1}\mu_{j_1}, \ldots, t_n\mu_{j_1})$. Then (A) holds, since $s\sigma = s\mu_{j_1}\sigma'$. For (B), we let $i'_j = i_j$ for all $1 \le j \le e$. Since $i \notin \{i_1, \ldots, i_e\}$, we obtain $t'|_{i'_j}\sigma' = t'|_{i_j}\sigma' = t|_{i_j}\mu_{j_1}\sigma' = t|_{i_j}\sigma$, which implies (B).

Thus, for any $(\mathcal{D}, \mathcal{R})$-chain tree T for a sharped term $w^\sharp$ there exists a $(\mathcal{D}', \mathcal{R})$-chain tree T' for the same term $w^\sharp$ where $|T|_{\{s\to t\}} = |T'|_{\mathcal{M}}$ and for any DT $u \to v \notin \{s \to t\} \cup \mathcal{M}$, we have $|T|_{\{u\to v\}} = |T'|_{\{u\to v\}}$. Hence, for any sharped term $w^\sharp$ and any $\mathcal{S} \subseteq \mathcal{D}$ with $\mathcal{S}' = \mathcal{S}[s \to t \,/\, \mathcal{M}]$, we have $\mathit{Cplx}_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle}(w^\sharp) \le \mathit{Cplx}_{\langle \mathcal{D}',\mathcal{S}',\mathcal{R}\rangle}(w^\sharp)$. This implies $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D}',\mathcal{S}',\mathcal{R}\rangle}$.

Moreover, if $\mathcal{K} \subseteq \mathcal{D}$ and $\mathcal{K}'$ results from $\mathcal{K}$ by removing $s \to t$ and all DTs that are reachable from $s \to t$ in the $(\mathcal{D}, \mathcal{R})$-dependency graph, then $\mathcal{K}'$ also contains no DT that is contained in $\mathcal{M}$ or reachable from $\mathcal{M}$ in the $(\mathcal{D}', \mathcal{R})$-dependency graph. Hence, for $\mathit{Cplx}_{\langle \mathcal{D},\mathcal{K}',\mathcal{R}\rangle}(w^\sharp)$ or $\mathit{Cplx}_{\langle \mathcal{D}',\mathcal{K}',\mathcal{R}\rangle}(w^\sharp)$ it suffices to consider chain trees not containing $s \to t$ or DTs from $\mathcal{M}$. Such chain trees are both $(\mathcal{D},\mathcal{R})$- and $(\mathcal{D}',\mathcal{R})$-chain trees. Hence, we obtain $\mathcal{C}plx_{\langle \mathcal{D},\mathcal{K}',\mathcal{R}\rangle}(w^\sharp) = \mathcal{C}plx_{\langle \mathcal{D}',\mathcal{K}',\mathcal{R}\rangle}(w^\sharp)$ for all $w^\sharp$ and thus, $\iota_{\langle \mathcal{D},\mathcal{K}',\mathcal{R}\rangle} = \iota_{\langle \mathcal{D}',\mathcal{K}',\mathcal{R}\rangle}$. As $\mathcal{K}' \subseteq \mathcal{K}$, we have $\iota_{\langle \mathcal{D},\mathcal{K}',\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle}$ by Lemma 39(g) and hence $\iota_{\langle \mathcal{D}',\mathcal{K}',\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle}$.

From $\iota_{\langle \mathcal{D},\mathcal{S},\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D}',\mathcal{S}',\mathcal{R}\rangle}$ and $\iota_{\langle \mathcal{D}',\mathcal{K}',\mathcal{R}\rangle} \sqsubseteq \iota_{\langle \mathcal{D},\mathcal{K},\mathcal{R}\rangle}$, we obtain that $\gamma_P \sqsubseteq \gamma_{\langle \mathcal{D}',\mathcal{S}',\mathcal{K}',\mathcal{R}\rangle}$, i.e., the narrowing processor is sound. □