International Conference on Rewriting Techniques and Applications 2010 (Edinburgh), pp. 385-400
http://rewriting.loria.fr/rta/

MODULAR COMPLEXITY ANALYSIS VIA RELATIVE COMPLEXITY

HARALD ZANKL AND MARTIN KORP

Institute of Computer Science, University of Innsbruck, Austria
E-mail address: {harald.zankl,martin.korp}@uibk.ac.at

Abstract. In this paper we introduce a modular framework which allows to infer (feasible) upper bounds on the (derivational) complexity of term rewrite systems by combining different criteria. All current investigations to analyze the derivational complexity are based on a single termination proof, possibly preceded by transformations. We prove that the modular framework is strictly more powerful than the conventional setting. Furthermore, the results have been implemented and experiments show significant gains in power.

1. Introduction

Term rewriting is a Turing complete model of computation. As an immediate consequence all interesting properties are undecidable. Nevertheless many powerful techniques have been developed for establishing *termination*. The majority of these techniques have been automated successfully. This development has been stimulated by the international competition of termination tools.[1] Most automated analyzers gain their power from a modular treatment of the rewrite system (typically via the dependency pair framework [2,11,22]).

For terminating rewrite systems Hofbauer and Lautemann [14] consider the length of derivations as a measurement for the complexity of rewrite systems. The resulting notion of *derivational complexity* relates the length of a rewrite sequence to the size of its starting term. Thereby it is, e.g., a suitable metric for the complexity of deciding the word problem for a given confluent and terminating rewrite system (since the decision procedure rewrites terms to normal form). If one regards a rewrite system as a program and wants to estimate the maximal number of computation steps needed to evaluate an expression to a result, then the special shape of the starting terms—a function applied to data which is in normal form—can be taken into account. Hirokawa and Moser [12] identified this special form of complexity and named it *runtime complexity*.

To show (feasible) upper complexity bounds currently few techniques are known. Typically termination criteria are restricted such that complexity bounds can be inferred. The early work by Hofbauer and Lautemann [14] considers polynomial interpretations, suitably

1998 ACM Subject Classification: F.2 Analysis of Algorithms and Problem Complexity, F.4 Mathematical Logic and Formal Languages.

Key words and phrases: term rewriting, complexity analysis, relative complexity, derivation length.

This research is supported by FWF (Austrian Science Fund) project P18763.

[1] http://termcomp.uibk.ac.at

Proceedings of the 21st International Conference on Rewriting Techniques and Applications, Edinburgh, July, 2010
Editors: Christopher Lynch
LIPIcs - Leibniz International Proceedings in Informatics. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.RTA.2010.385

restricted, to admit quadratic derivational complexity. Match-bounds [9] and arctic matrix interpretations [16] induce linear derivational complexity and triangular matrix interpretations [19] admit polynomially long derivations (the dimension of the matrices yields the degree of the polynomial). All these methods share the property that until now they have been used directly only, meaning that a single termination technique has to orient all rules in one go. However, using direct criteria exclusively is problematic due to their restricted power.

In [12,13] Hirokawa and Moser lift many aspects of the dependency pair framework from termination analysis into the complexity setting, resulting in the notion of weak dependency pairs. So for the special case of runtime complexity for the first time a modular approach has been introduced. There the modular aspect amounts to using different interpretation based criteria for (parts of the) weak dependency graph and the usable rules. However, still all rewrite rules considered must be oriented strictly in one go and only restrictive criteria may be applied for the usable rules. A further drawback of weak dependency pairs is that they may only be used for bounding runtime complexity while there seems to be no hope to generalize the method to derivational complexity.

In this paper we present a different approach which admits a fully modular treatment. The approach is general enough that it applies to derivational complexity (and hence also to runtime complexity) and basic enough that it allows to combine completely different complexity criteria such as match-bounds and triangular matrix interpretations. By the modular combination of different criteria also gains in power are achieved. These gains come in two flavors. On one hand our approach allows to obtain lower complexity bounds for several rewrite systems where bounds have already been established before and on the other hand we found bounds for systems that could not be dealt with so far automatically.

The remainder of the paper is organized as follows. In Section 2 preliminaries about term rewriting and complexity analysis are fixed. Afterwards, Section 3 familiarizes the reader with the concept of a suitable complexity measurement for relative rewriting. Section 4 formulates a modular framework for complexity analysis based on relative complexity. In Section 5 we show that the modular setting is strictly more powerful than the conventional approach. Our results have been implemented in the complexity prover GaT. The technical details can be inferred from Section 6. Section 7 is devoted to demonstrate the power of the modular treatment by means of an empirical evaluation. Section 8 concludes.

2. Preliminaries

We assume familiarity with (relative) term rewriting [3, 10, 21]. Let $\mathcal{F}$ be a signature and $\mathcal{V}$ be a disjoint set of variables. By $\mathcal{T}(\mathcal{F},\mathcal{V})$ we denote the set of terms over $\mathcal{F}$ and $\mathcal{V}$ and by $\mathcal{T}(\mathcal{F})$ the set of ground terms over $\mathcal{F}$. We write $\mathcal{F}\mathsf{un}(t)$ for the set of function symbols occurring in a term t. The size of a term t is denoted $|t|$ and $\|t\|$ computes the number of occurrences of function symbols in t. Positions are used to address symbol occurrences in terms. Given a term t and a position $p \in \mathcal{P}\mathsf{os}(t)$, we write $t(p)$ for the symbol at position p. We use $\mathcal{F}\mathcal{P}\mathsf{os}(t)$ to denote the subset of positions $p \in \mathcal{P}\mathsf{os}(t)$ such that $t(p) \in \mathcal{F}$.

A *rewrite rule* is a pair of terms (l, r), written $l \to r$ such that l is not a variable and all variables in r are contained in l. A *term rewrite system* (TRS for short) is a set of rewrite rules. For complexity analysis we assume TRSs to be finite. A TRS $\mathcal{R}$ is said to be *duplicating* if there exists a rewrite rule $l \to r \in \mathcal{R}$ and a variable x that occurs more often in r than in l. A TRS is called *linear* if for all rewrite rules $l \to r \in \mathcal{R}$ any variable x

occurs at most once in l and r, respectively. A *rewrite relation* is a binary relation on terms that is closed under contexts and substitutions. For a TRS $\mathcal{R}$ we define $\to_{\mathcal{R}}$ to be the smallest rewrite relation that contains $\mathcal{R}$. As usual $\to^*$ denotes the reflexive and transitive closure of $\to$ and $\to^n$ the n-th iterate of $\to$. A *relative* TRS $\mathcal{R}/\mathcal{S}$ is a pair of TRSs $\mathcal{R}$ and $\mathcal{S}$ with the induced rewrite relation $\to_{\mathcal{R}/\mathcal{S}} = \to_{\mathcal{S}}^* \cdot \to_{\mathcal{R}} \cdot \to_{\mathcal{S}}^*$. In the sequel we will sometimes identify a TRS $\mathcal{R}$ with the relative TRS $\mathcal{R}/\varnothing$.

The *derivation length* of a term t with respect to a relation $\to$ and a set of terms L is defined as follows: $\mathsf{dl}(t, \to, L) = \mathsf{max}\{m \mid \exists u\, t \to^m u, t \in L\}$. We abbreviate $\mathsf{dl}(t, \to, \mathcal{T}(\mathcal{F}, \mathcal{V}))$ by $\mathsf{dl}(t, \to)$. The *derivational complexity* computes the maximal derivation length of all terms up to a given size, i.e., $\mathsf{dc}(n, \to) = \mathsf{max}\{\mathsf{dl}(t, \to) \mid |t| \leqslant n\}$. Sometimes we say that $\mathcal{R}$ ($\mathcal{R}/\mathcal{S}$) has linear, quadratic, etc. derivational complexity if $\mathsf{dc}(n, \to_{\mathcal{R}})$ ($\mathsf{dc}(n, \to_{\mathcal{R}/\mathcal{S}})$) can be bounded by a linear, quadratic, etc. polynomial in n.

2.1. Triangular Matrix Interpretations

An *$\mathcal{F}$-algebra* $\mathcal{A}$ consists of a non-empty carrier A and a set of interpretations $f_{\mathcal{A}}$ for every $f \in \mathcal{F}$. By $[\alpha](\cdot)_{\mathcal{A}}$ we denote the usual evaluation function of $\mathcal{A}$ according to an assignment α. An $\mathcal{F}$-algebra $\mathcal{A}$ together with two relations $\succ$ and $\succeq$ on A is called a *monotone algebra* if every $f_{\mathcal{A}}$ is monotone with respect to $\succ$ and $\succeq$, $\succ$ is a well-founded order, and $\succ \cdot \succeq \subseteq \succ$ and $\succeq \cdot \succ \subseteq \succ$ holds. Any monotone algebra $(\mathcal{A}, \succ, \succeq)$ induces a well-founded order on terms, i.e., $s \succ_{\mathcal{A}} t$ if for any assignment α the condition $[\alpha](s)_{\mathcal{A}} \succ [\alpha](t)_{\mathcal{A}}$ holds. The order $\succeq_{\mathcal{A}}$ is defined similarly. A relative TRS $\mathcal{R}/\mathcal{S}$ is *compatible* with a monotone algebra $(\mathcal{A}, \succ_{\mathcal{A}}, \succeq_{\mathcal{A}})$ if $\mathcal{R} \subseteq \succ_{\mathcal{A}}$ and $\mathcal{S} \subseteq \succeq_{\mathcal{A}}$. *Matrix interpretations* $(\mathcal{M}, \succ_{\mathcal{M}}, \succeq_{\mathcal{M}})$ (often just denoted $\mathcal{M}$) are a special form of monotone algebras. Here the carrier is $\mathbb{N}^d$ for some fixed dimension $d \in \mathbb{N} \setminus \{0\}$. The order $\succeq_{\mathcal{M}}$ is the point-wise extension of $\geqslant_{\mathbb{N}}$ to vectors and $\vec{u} \succ_{\mathcal{M}} \vec{v}$ if $\vec{u}_1 >_{\mathbb{N}} \vec{v}_1$ and $\vec{u} \succeq_{\mathcal{M}} \vec{v}$. If every $f \in \mathcal{F}$ of arity n is interpreted as $f_{\mathcal{M}}(\vec{x_1}, \ldots, \vec{x_n}) = F_1\vec{x_1} + \cdots + F_n\vec{x_n} + \vec{f}$ where $F_i \in \mathbb{N}^{d \times d}$ and $\vec{f} \in \mathbb{N}^d$ for all $1 \leqslant i \leqslant n$ then monotonicity of $\succ_{\mathcal{M}}$ is achieved by demanding $F_{i(1,1)} \geqslant 1$ for any $1 \leqslant i \leqslant n$. A matrix interpretation where for every $f \in \mathcal{F}$ all F_i ($1 \leqslant i \leqslant \mathsf{arity}(f)$) are upper triangular is called *triangular matrix interpretation* (abbreviated by TMI). A square matrix A of dimension d is of *upper triangular* shape if $A_{(i,i)} \leqslant 1$ and $A_{(i,j)} = 0$ if $i > j$ for all $1 \leqslant i, j \leqslant d$. For historic reasons a TMI based on matrices of dimension one is also called *strongly linear interpretation* (SLI for short). In [19] it is shown that the derivational complexity of a TRS $\mathcal{R}$ is bounded by a polynomial of degree d if there exists a TMI $\mathcal{M}$ of dimension d such that $\mathcal{R}$ is compatible with $\mathcal{M}$.

2.2. Match-Bounds

Let $\mathcal{F}$ be a signature, $\mathcal{R}$ a TRS over $\mathcal{F}$, and $L \subseteq \mathcal{T}(\mathcal{F})$ a set of ground terms. The set $\{t \in \mathcal{T}(\mathcal{F}) \mid s \to_{\mathcal{R}}^* t \text{ for some } s \in L\}$ of reducts of L is denoted by $\to_{\mathcal{R}}^*(L)$. Given a set $N \subseteq \mathbb{N}$ of natural numbers, the signature $\mathcal{F} \times N$ is denoted by $\mathcal{F}_N$. Here function symbols (f, n) with $f \in \mathcal{F}$ and $n \in N$ have the same arity as f and are written as f_n. The mappings $\mathsf{lift}_c \colon \mathcal{F} \to \mathcal{F}_{\mathbb{N}}$, $\mathsf{base} \colon \mathcal{F}_{\mathbb{N}} \to \mathcal{F}$, and $\mathsf{height} \colon \mathcal{F}_{\mathbb{N}} \to \mathbb{N}$ are defined as $\mathsf{lift}_c(f) = f_c$, $\mathsf{base}(f_c) = f$, and $\mathsf{height}(f_c) = c$ for all $f \in \mathcal{F}$ and $c \in \mathbb{N}$. They are extended to terms, sets of terms, and TRSs in the obvious way. The TRS $\mathsf{match}(\mathcal{R})$ over the signature $\mathcal{F}_{\mathbb{N}}$ consists of all rewrite rules $l' \to \mathsf{lift}_c(r)$ for which there exists a rule $l \to r \in \mathcal{R}$ such that $\mathsf{base}(l') = l$ and $c = 1 + \mathsf{min}\{\mathsf{height}(l'(p)) \mid p \in \mathcal{FP}\mathsf{os}(l)\}$. Here $c \in \mathbb{N}$. The restriction of

$\mathsf{match}(\mathcal{R})$ to the signature $\mathcal{F}_{\{0,\ldots,c\}}$ is denoted by $\mathsf{match}_c(\mathcal{R})$. Let L be a set of terms. The TRS $\mathcal{R}$ is called *match-bounded* for L if there exists a $c \in \mathbb{N}$ such that the maximum height of function symbols occurring in terms in $\to^*_{\mathsf{match}(\mathcal{R})}(\mathsf{lift}_0(L))$ is at most c. If we want to make the bound c precise, we say that $\mathcal{R}$ is match-bounded for L *by* c. If we do not specify the set of terms L then it is assumed that $L = \mathcal{T}(\mathcal{F})$. If a linear TRS $\mathcal{R}$ is match-bounded for a language L then $\mathcal{R}$ is terminating on L. Furthermore $\mathsf{dl}(t, \to_{\mathcal{R}}, L)$ is bounded by a linear polynomial for any term $t \in L$ [9].

In order to prove that a TRS $\mathcal{R}$ is match-bounded for some language L, the idea is to construct a tree automaton that is compatible with $\mathsf{match}(\mathcal{R})$ and $\mathsf{lift}_0(L)$. A tree automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ is said to be *compatible* with some TRS $\mathcal{R}$ and some language L if $L \subseteq \mathcal{L}(\mathcal{A})$ and for each rewrite rule $l \to r \in \mathcal{R}$ and state substitution $\sigma \colon \mathcal{V}\mathsf{ar}(l) \to Q$ such that $l\sigma \to^*_\Delta q$ it holds that $r\sigma \to^*_\Delta q$.

3. Relative Complexity

In this section we introduce complexity analysis for relative rewriting, i.e., given $\mathcal{R}/\mathcal{S}$ only the $\mathcal{R}$-steps contribute to the complexity. To estimate the derivational complexity of a relative TRS $\mathcal{R}/\mathcal{S}$, a pair of orderings $(\succ, \succeq)$ will be used such that $\mathcal{R} \subseteq \succ$ and $\mathcal{S} \subseteq \succeq$. The necessary properties of these orderings are given in the next definition.

Definition 3.1. A *complexity pair* $(\succ, \succeq)$ consists of two finitely branching rewrite relations $\succ$ and $\succeq$ that are *compatible*, i.e., $\succeq \cdot \succ \subseteq \succ$ and $\succ \cdot \succeq \subseteq \succ$. We call a relative TRS $\mathcal{R}/\mathcal{S}$ *compatible* with a complexity pair $(\succ, \succeq)$ if $\mathcal{R} \subseteq \succ$ and $\mathcal{S} \subseteq \succeq$.

The next lemma states that given a complexity pair $(\succ, \succeq)$ and a compatible relative TRS $\mathcal{R}/\mathcal{S}$, the $\succ$ ordering is crucial for estimating the derivational complexity of $\mathcal{R}/\mathcal{S}$. Intuitively the result states that every $\mathcal{R}/\mathcal{S}$-step gives rise to at least one $\succ$-step.

Lemma 3.2. *Let $(\succ, \succeq)$ be a complexity pair and $\mathcal{R}/\mathcal{S}$ be a compatible relative TRS. Then for any term t terminating on $\mathcal{R}/\mathcal{S}$ we have $\mathsf{dl}(t, \succ) \geqslant \mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}})$.*

Proof. By assumption $\mathcal{R}/\mathcal{S}$ is compatible with $(\succ, \succeq)$. Since $\succ$ and $\succeq$ are rewrite relations $\to_{\mathcal{R}} \subseteq \succ$ and $\to_{\mathcal{S}} \subseteq \succeq$ holds. From the compatibility of $\succ$ and $\succeq$ we obtain $\to_{\mathcal{R}/\mathcal{S}} \subseteq \succ$. Hence for any sequence

$$t \to_{\mathcal{R}/\mathcal{S}} t_1 \to_{\mathcal{R}/\mathcal{S}} t_2 \to_{\mathcal{R}/\mathcal{S}} \cdots$$

also

$$t \succ t_1 \succ t_2 \succ \cdots$$

holds. The result follows immediately from this. ∎

Obviously $\succ$ must be at least well-founded if *finite* complexities should be estimated. Because we are especially interested in feasible upper bounds the following corollary is specialized to polynomials.

Corollary 3.3. *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS compatible with a complexity pair $(\succ, \succeq)$. If the derivational complexity of $\succ$ is linear, quadratic, etc. then the derivational complexity of $\mathcal{R}/\mathcal{S}$ is linear, quadratic, etc.*

Proof. By Lemma 3.2. ∎

This corollary allows to investigate the derivational complexity of (compatible) complexity pairs instead of the derivational complexity of the underlying relative TRS. Complexity pairs can, e.g., be obtained by TMIs.

Theorem 3.4. *Let $(\mathcal{M}, \succ_{\mathcal{M}}, \succeq_{\mathcal{M}})$ be a TMI of dimension d. Then $(\succ_{\mathcal{M}}, \succeq_{\mathcal{M}})$ is a complexity pair. Furthermore $\mathsf{dl}(t, \succ_{\mathcal{M}})$ is bounded by a polynomial of degree d for any term t.*

Proof. Straightforward from [19, Theorem 6]. ∎

The following example familiarizes the reader with relative derivational complexity analysis.

Example 3.5. Consider the relative TRS $\mathcal{R}/\mathcal{S}$ where $\mathcal{R} = \{\mathsf{f}(x) \to x\}$ and $\mathcal{S} = \{\mathsf{a} \to \mathsf{a}\}$. Then the SLI $\mathcal{M}$ satisfying $f_{\mathcal{M}}(x) = x + 1$ and $\mathsf{a}_{\mathcal{M}} = 0$ induces the complexity pair $(\succ_{\mathcal{M}}, \succeq_{\mathcal{M}})$. Furthermore $\mathcal{R}/\mathcal{S}$ is compatible with $(\succ_{\mathcal{M}}, \succeq_{\mathcal{M}})$. Theorem 3.4 gives a linear bound on $\succ_{\mathcal{M}}$. Hence $\mathcal{R}/\mathcal{S}$ admits (at most) linear derivational complexity by Corollary 3.3. It is easy to see that this bound is tight.

4. Modular Complexity Analysis

Although Theorem 3.4 in combination with Corollary 3.3 is already powerful, a severe drawback is that given a relative TRS $\mathcal{R}/\mathcal{S}$ all rules in $\mathcal{R}$ must be strictly oriented by $\succ_{\mathcal{M}}$. The next (abstract) example demonstrates the basic idea of an approach that allows to weaken this burden.

Example 4.1. Consider the relative TRS $\mathcal{R}/\mathcal{S}$ where $\mathcal{R} = \{r, r'\}$. Instead of estimating $\mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}})$ directly, we want to bound it by $\mathsf{dl}(t, \to_{\{r\}/(\{r'\} \cup \mathcal{S})}) + \mathsf{dl}(t, \to_{\{r'\}/(\{r\} \cup \mathcal{S})})$. To proceed we separate the r from the r'-steps. I.e., any reduction sequence in $\mathcal{R}/\mathcal{S}$ can be written as

$$t \quad \to_{\{r\}/\mathcal{S}} \quad t_1 \quad \to_{\{r\}/\mathcal{S}} \quad t_2 \quad \to_{\{r'\}/\mathcal{S}} \quad t_3 \quad \to_{\{r\}/\mathcal{S}} \quad \cdots$$

which immediately gives rise to a sequence

$$t \to_{\{r\}/(\{r'\} \cup \mathcal{S})} t_1 \to_{\{r\}/(\{r'\} \cup \mathcal{S})} t_2 \to_{\{r'\}/(\{r\} \cup \mathcal{S})} t_3 \to_{\{r\}/(\{r'\} \cup \mathcal{S})} \cdots$$

Obviously $\mathsf{dl}(t, \to_{\{r\}/(\{r'\} \cup \mathcal{S})}) + \mathsf{dl}(t, \to_{\{r'\}/(\{r\} \cup \mathcal{S})}) \geqslant \mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}})$.

Next we formalize the main observation needed for modular complexity analysis of a relative TRS $\mathcal{R}/\mathcal{S}$. As already indicated in the example above the key idea is that every rule from $\mathcal{R}$ can be investigated relative to the other rules in $\mathcal{R}$ and $\mathcal{S}$. The next theorem states the main result in this direction. In the sequel we assume that $\mathcal{R}_i \subseteq \mathcal{R}$ for any i. Furthermore, sometimes $\mathcal{R}_i$ refers to $\{r_i\}$ if $\mathcal{R} = \{r_1, \ldots, r_n\}$ and $1 \leqslant i \leqslant n$. The context clarifies if $\mathcal{R}_i$ is an arbitrary subset of $\mathcal{R}$ or its i-th rule. In addition we denote by $\mathcal{S}_i$ the TRS $(\mathcal{S} \cup \mathcal{R}) \setminus \mathcal{R}_i$. If $\mathcal{S}$ is not explicitly mentioned then $\mathcal{S} = \varnothing$.

Theorem 4.2. *Let $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ be a relative TRS for which the term t terminates. Then $\mathsf{dl}(t, \to_{\mathcal{R}_1/\mathcal{S}_1}) + \mathsf{dl}(t, \to_{\mathcal{R}_2/\mathcal{S}_2}) \geqslant \mathsf{dl}(t, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}})$.*

Proof. We abbreviate $\mathcal{R}_1 \cup \mathcal{R}_2$ by $\mathcal{R}$. Assume that $\mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}}) = m$. Then there exists a rewrite sequence

$$t \to_{\mathcal{R}/\mathcal{S}} t_1 \to_{\mathcal{R}/\mathcal{S}} t_2 \to_{\mathcal{R}/\mathcal{S}} \cdots \to_{\mathcal{R}/\mathcal{S}} t_{m-1} \to_{\mathcal{R}/\mathcal{S}} t_m \tag{1}$$

of length m. Next we investigate this sequence for every relative TRS $\mathcal{R}_i/\mathcal{S}_i$ ($1 \leqslant i \leqslant 2$) where m_i overestimates how often rules from $\mathcal{R}_i$ have been applied in the original sequence. Fix i. If the sequence (1) does not contain an $\mathcal{R}_i$ step then $t \to_{\mathcal{S}_i}^{m} t_m$ and $m_i = 0$. In the other case there exists a maximal (with respect to m_i) sequence

$$t \to_{\mathcal{R}_i/\mathcal{S}_i} t_{i_1} \to_{\mathcal{R}_i/\mathcal{S}_i} t_{i_2} \to_{\mathcal{R}_i/\mathcal{S}_i} \cdots \to_{\mathcal{R}_i/\mathcal{S}_i} t_{i_{m_i-1}} \to_{\mathcal{R}_i/\mathcal{S}_i} t_m \tag{2}$$

where $i_1 < i_2 < \cdots < i_{m_i-1} < m$. Together with the fact that every rewrite rule in $\mathcal{R}$ is either contained in $\mathcal{R}_1$ or $\mathcal{R}_2$ we have $m_1 + m_2 \geqslant m$. If $m_i = 0$ we obviously have $\mathsf{dl}(t, \to_{\mathcal{R}_i/\mathcal{S}_i}) \geqslant m_i$ and if $t \to_{\mathcal{R}_i/\mathcal{S}_i}^{m_i} t_m$ with $m_i > 0$ we know that $\mathsf{dl}(t, \to_{\mathcal{R}_i/\mathcal{S}_i}) \geqslant m_i$ by the choice of sequence (2). (Note that in both cases it can happen that $\mathsf{dl}(t, \to_{\mathcal{R}_i/\mathcal{S}_i}) > m_i$ because sequence (1) need not be maximal with respect to $\to_{\mathcal{R}_i/\mathcal{S}_i}$.) Putting things together yields

$$\mathsf{dl}(t, \to_{\mathcal{R}_1/\mathcal{S}_1}) + \mathsf{dl}(t, \to_{\mathcal{R}_2/\mathcal{S}_2}) \geqslant m_1 + m_2 \geqslant m = \mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}})$$

which concludes the proof. ∎

As already indicated, the reverse direction of the above theorem does not hold. This is illustrated by the following example.

Example 4.3. Consider the relative TRS $\mathcal{R}/\mathcal{S}$ with $\mathcal{R} = \{\mathsf{a} \to \mathsf{b}, \mathsf{a} \to \mathsf{c}\}$ and $\mathcal{S} = \varnothing$. We have $\mathsf{a} \to_{\mathcal{R}/\mathcal{S}} \mathsf{b}$ or $\mathsf{a} \to_{\mathcal{R}/\mathcal{S}} \mathsf{c}$. Hence $\mathsf{dl}(\mathsf{a}, \to_{\mathcal{R}/\mathcal{S}}) = 1$. However, the sum of the derivation lengths $\mathsf{dl}(\mathsf{a}, \to_{\mathcal{R}_1/\mathcal{S}_1})$ and $\mathsf{dl}(\mathsf{a}, \to_{\mathcal{R}_2/\mathcal{S}_2})$ is 2.

Although for Theorem 4.2 equality cannot be established the next result states that for complexity analysis this does not matter.

Theorem 4.4. *Let $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ be a relative TRS for which the term t terminates. Then* $\mathsf{max}\{\mathcal{O}(\mathsf{dl}(t, \to_{\mathcal{R}_1/\mathcal{S}_1})), \mathcal{O}(\mathsf{dl}(t, \to_{\mathcal{R}_2/\mathcal{S}_2}))\} = \mathcal{O}(\mathsf{dl}(t, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}))$.

Proof. We abbreviate the union $\mathcal{R}_1 \cup \mathcal{R}_2$ by $\mathcal{R}$. The $\geqslant$-direction follows immediately from Theorem 4.2 and some basic properties of the $\mathcal{O}$-notation like $\mathsf{max}_{1 \leqslant i \leqslant 2}\{\mathcal{O}(\mathsf{dl}(t, \to_{\mathcal{R}_i/\mathcal{S}_i}))\} = \mathcal{O}(\sum_{1 \leqslant i \leqslant 2}\{\mathsf{dl}(t, \to_{\mathcal{R}_i/\mathcal{S}_i})\})$. For the $\leqslant$-direction we select a $j \in \{1, 2\}$ such that $\mathcal{R}_j/\mathcal{S}_j$ determines the complexity of $\mathcal{R}/\mathcal{S}$, i.e., $\mathcal{O}(\mathsf{dl}(t, \to_{\mathcal{R}_j/\mathcal{S}_j})) = \mathsf{max}_{1 \leqslant i \leqslant 2}\{\mathcal{O}(\mathsf{dl}(t, \to_{\mathcal{R}_i/\mathcal{S}_i}))\}$. By construction of $\mathcal{R}_j/\mathcal{S}_j$ any derivation of the form

$$t \to_{\mathcal{R}_j/\mathcal{S}_j} t_1 \to_{\mathcal{R}_j/\mathcal{S}_j} \cdots \to_{\mathcal{R}_j/\mathcal{S}_j} t_m$$

induces a derivation

$$t \to_{\mathcal{R}/\mathcal{S}}^{+} t_1 \to_{\mathcal{R}/\mathcal{S}}^{+} \cdots \to_{\mathcal{R}/\mathcal{S}}^{+} t_m$$

of at least the same length. Putting things together finishes the proof. ∎

Theorems 4.2 and 4.4 allow to split a relative TRS $\mathcal{R}/\mathcal{S}$ into *smaller* components $\mathcal{R}_1/\mathcal{S}_1$ and $\mathcal{R}_2/\mathcal{S}_2$—such that $\mathcal{R}_1 \cup \mathcal{R}_2 = \mathcal{R}$—and evaluate the complexities of these components (e.g., by different complexity pairs) independently. Note that this approach is not restricted to relative rewriting. To estimate the complexity of a (non-relative) TRS $\mathcal{R}$ just consider the relative TRS $\mathcal{R}/\varnothing$.

In Theorem 3.4 we have already seen one method to obtain complexity pairs. In the next subsection we study how the match-bounds technique can be suited for relative complexity analysis. Afterwards, in Section 4.2, we show that under certain circumstances the derivational complexity of a relative TRS $\mathcal{R}/\mathcal{S}$ can be estimated by considering $\mathcal{R}_1/\mathcal{S}_1$ and $\mathcal{R}_2/\mathcal{S}$. Note that Theorem 4.2 requires $\mathcal{R}_2/\mathcal{S}_2$ with $\mathcal{S}_2 = \mathcal{R}_1 \cup \mathcal{S}$ instead of $\mathcal{R}_2/\mathcal{S}$.

4.1. Complexity Pairs via Match-Bounds

It is easy to use the match-bound technique for estimating the derivational complexity of a relative TRS $\mathcal{R}/\mathcal{S}$; just check for match-boundedness of $\mathcal{R} \cup \mathcal{S}$. This process either succeeds by proving that the combined TRS is match-bounded, or, when $\mathcal{R} \cup \mathcal{S}$ cannot be proved to be match-bounded, it fails. Since the construction of a compatible tree automaton does not terminate for TRSs that are not match-bounded, the latter situation typically does not happen. This behavior causes two serious problems. On the one hand we cannot benefit from Theorem 4.4 because whenever we split a relative TRS $\mathcal{R}/\mathcal{S}$ into smaller components $\mathcal{R}_1/\mathcal{S}_1$ and $\mathcal{R}_2/\mathcal{S}_2$ then $\mathcal{R}_1 \cup \mathcal{S}_1$ is match-bounded if and only if $\mathcal{R}_2 \cup \mathcal{S}_2$ is match-bounded since both TRSs coincide. On the other hand the match-bound technique cannot cooperate with other techniques since either linear derivational complexity of all or none of the rules in $\mathcal{R}$ is shown. In [23] these problems have been addressed by checking if there is a $c \in \mathbb{N}$ such that the maximum height of function symbols occurring in terms in derivations caused by the TRS $\mathsf{match}_{c+1}(\mathcal{R}) \cup \mathsf{match}_c(\mathcal{S}) \cup \mathsf{lift}_c(\mathcal{S})$ is at most c. Below we follow a different approach which seems to be more suitable for our setting. That is, given some relative TRS $\mathcal{R}/\mathcal{S}$ we adapt the match-bound technique in such a way that it can orient the rewrite rules in $\mathcal{R}$ strictly and the ones in $\mathcal{S}$ weakly. To this end we introduce a new enrichment $\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})$. The key idea behind the new enrichment is to keep heights until a labeled version of a rule in $\mathcal{R}$ is applied.

To simplify the presentation we consider linear TRSs only. The extension to non-duplicating TRSs is straightforward and follows [17].

Definition 4.5. Let $\mathcal{S}$ be a TRS over a signature $\mathcal{F}$. The TRS $\mathsf{match\text{-}RT}(\mathcal{S})$ over the signature $\mathcal{F}_{\mathbb{N}}$ consists of all rules $l' \to \mathsf{lift}_c(r)$ such that $\mathsf{base}(l') \to r \in \mathcal{S}$. Here $c = \mathsf{height}(l'(\epsilon))$ if $\|\mathsf{base}(l')\| \geqslant \|r\|$ and $\mathsf{lift}_c(\mathsf{base}(l')) = l'$, and $c = \mathsf{min}\,\{1 + \mathsf{height}(l'(p)) \mid p \in \mathcal{FP}\mathsf{os}(l')\}$ otherwise. Given a relative TRS $\mathcal{R}/\mathcal{S}$, the relative TRS $\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})$ is defined as $\mathsf{match}(\mathcal{R})/\mathsf{match\text{-}RT}(\mathcal{S})$.

To satisfy the requirement that the new enrichment $\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})$ counts only $\mathcal{R}$-steps we try to keep the labels of the function symbols in a contracted redex if a size-preserving or size-decreasing rewrite rule in $\mathcal{S}$ is applied. We need this requirement on the size that the new enrichment is compatible with multi-set orderings $\succ_{\mathsf{mul}}$ and $\succeq_{\mathsf{mul}}$ which induce a linear complexity.

Example 4.6. Consider the TRS $\mathcal{R} = \{\mathsf{rev}(x) \to \mathsf{rev}'(x, \mathsf{nil}), \mathsf{rev}'(\mathsf{nil}, y) \to y\}$ and the TRS $\mathcal{S} = \{\mathsf{rev}'(\mathsf{cons}(x, y), z) \to \mathsf{rev}'(y, \mathsf{cons}(x, z))\}$. Then $\mathsf{match}(\mathcal{R})$ contains the rules

$$\begin{array}{ll} \mathsf{rev}_0(x) \to \mathsf{rev}'_1(x, \mathsf{nil}_1) & \mathsf{rev}'_0(\mathsf{nil}_0, y) \to y \\ \mathsf{rev}_1(x) \to \mathsf{rev}'_2(x, \mathsf{nil}_2) & \mathsf{rev}'_0(\mathsf{nil}_1, y) \to y \\ \quad \cdots & \cdots \end{array}$$

and $\mathsf{match\text{-}RT}(\mathcal{S})$ contains

$$\begin{array}{c} \mathsf{rev}'_0(\mathsf{cons}_0(x, y), z) \to \mathsf{rev}'_0(y, \mathsf{cons}_0(x, z)) \\ \mathsf{rev}'_0(\mathsf{cons}_1(x, y), z) \to \mathsf{rev}'_1(y, \mathsf{cons}_1(x, z)) \\ \cdots \end{array}$$

Both infinite TRSs together constitute $\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})$.

In order to prove that a relative TRS $\mathcal{R}/\mathcal{S}$ admits at most linear derivational complexity using the enrichment $\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})$ we need the property defined below.

Definition 4.7. Let $\mathcal{R}/\mathcal{S}$ be a relative TRS. We call $\mathcal{R}/\mathcal{S}$ *match-RT-bounded* for a set of terms L if there exists a number $c \in \mathbb{N}$ such that the height of function symbols occurring in terms in $\to^*_{\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})}(\mathsf{lift}_0(L))$ is at most c.

Let $\mathcal{MF}\mathsf{un}(t) = \{\mathsf{height}(f) \mid f \in \mathcal{F}\mathsf{un}(t)\}$ denote the multiset of the heights of the function symbols that occur in the term t. The precedence $\succ$ on $\mathbb{N}$ is defined as $i \succ j$ if and only if $j >_{\mathbb{N}} i$. Let $\mathcal{M}\mathsf{ul}(\mathbb{N})$ denote all multisets over $\mathbb{N}$. We write $s \succ_{\mathsf{mul}} t$ for terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ if $\mathcal{MF}\mathsf{un}(s)$ is greater than $\mathcal{MF}\mathsf{un}(t)$ in the multiset extension of $\succ$ on $\mathbb{N}$. Similarly, $s \succeq_{\mathsf{mul}} t$ if $s \succ_{\mathsf{mul}} t$ or $\mathcal{MF}\mathsf{un}(s) = \mathcal{MF}\mathsf{un}(t)$.

A key property of the relative TRS $\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})$ is that the rewrite rules which originate from rules in $\mathcal{R}$ can be oriented by $\succ_{\mathsf{mul}}$ and all other rules can be oriented by $\succeq_{\mathsf{mul}}$. Hence the derivation length induced by the relative TRS $\mathcal{R}/\mathcal{S}$ on some language L is bounded by a linear polynomial whenever $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for L.

Lemma 4.8. *Let $\mathcal{R}$ and $\mathcal{S}$ be two non-duplicating TRSs. We have $\to_{\mathsf{match}(\mathcal{R})} \subseteq \succ_{\mathsf{mul}}$ and $\to_{\mathsf{match\text{-}RT}(\mathcal{S})} \subseteq \succeq_{\mathsf{mul}}$.*

Proof. From the proof of [9, Lemma 17] we know that for a non-duplicating TRS $\mathcal{R}$ and two terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, $s \succ_{\mathsf{mul}} t$ whenever $s \to_{\mathsf{match}(\mathcal{R})} t$. If $s \to_{\mathsf{match\text{-}RT}(\mathcal{S})} t$ then either $\mathcal{MF}\mathsf{un}(s) \supseteq \mathcal{MF}\mathsf{un}(t)$ or $s \to_{\mathsf{match}(\mathcal{S})} t$ by Definition 4.5. In the former case we have $s \succeq_{\mathsf{mul}} t$ and in the latter one $s \succ_{\mathsf{mul}} t$ and hence $s \succeq_{\mathsf{mul}} t$ by the definition of $\succeq_{\mathsf{mul}}$. This concludes the proof of the lemma. ∎

Theorem 4.9. *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS such that $\mathcal{R}$ and $\mathcal{S}$ are linear and let L be a set of terms. If $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for L then $\mathcal{R}/\mathcal{S}$ is terminating on L. Furthermore $\mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}}, L)$ is bounded by a linear polynomial for any term $t \in L$.*

Proof. At first we show that $\mathcal{R}/\mathcal{S}$ is terminating on L. Assume to the contrary that there is an infinite rewrite sequence of the form

$$t_1 \to_{\mathcal{R}/\mathcal{S}} t_2 \to_{\mathcal{R}/\mathcal{S}} t_3 \to_{\mathcal{R}/\mathcal{S}} \cdots$$

with $t_1 \in L$. Because $\mathcal{R}$ and $\mathcal{S}$ are left-linear, this derivation can be lifted to an infinite $\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})$ rewrite sequence

$$t'_1 \to_{\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})} t'_2 \to_{\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})} t'_3 \to_{\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})} \cdots$$

starting from $t'_1 = \mathsf{lift}_0(t_1)$. Since the original sequence contains infinitely many $\mathcal{R}$-steps the lifted sequence contains infinitely many $\mathsf{match}(\mathcal{R})$-steps. Moreover, because $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for L, there is a $c \geqslant 0$ such that the height of every function symbol occurring in a term in the lifted sequence is at most c. Lemma 4.8 as well as compatibility of $\succ_{\mathsf{mul}}$ and $\succeq_{\mathsf{mul}}$ yields $t'_i \succ_{\mathsf{mul}} t'_{i+1}$ for all $i \geqslant 1$. However, this is excluded because $\succ$ is well-founded on $\{0, \ldots, c\}$ and hence $\succ_{\mathsf{mul}}$ is well-founded on $\mathcal{T}(\mathcal{F}_{\{0,\ldots,c\}}, \mathcal{V})$ [3]. To prove the second part of the theorem consider an arbitrary (finite) rewrite sequence

$$t \to_{\mathcal{R}/\mathcal{S}} t_1 \to_{\mathcal{R}/\mathcal{S}} \cdots \to_{\mathcal{R}/\mathcal{S}} t_m$$

with $t \in L$. Similar as before this sequence can be lifted to a $\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})$ sequence

$$t' \to_{\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})} t'_1 \to_{\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})} \cdots \to_{\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})} t'_m$$

of the same length starting from $t' = \mathsf{lift}_0(t)$ such that $t'_i \succ_{\mathsf{mul}} t'_{i+1}$ for all $i \geqslant 0$. Here $t'_0 = t'$. Because $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for L, we know that there is a $c \geqslant 0$ such that the height of every function symbol occurring in a term in the lifted sequence is at most c. Let k be the maximal number of function symbols occurring in some right-hand side in $\mathcal{R} \cup \mathcal{S}$. Due to a remark in [4] we know that the length of the $\succ_{\mathsf{mul}}$ chain from t' to t'_m is bounded by $\|t'\| \cdot (k+1)^c$. Since $\|t'\| = \|t\|$ and both, the lifted as well as the original sequence are as long as the $\succ_{\mathsf{mul}}$ chain starting at t', we conclude that the length of the $\mathcal{R}/\mathcal{S}$ rewrite sequence starting at the term t is bounded by $\|t\| \cdot (k+1)^c$. ∎

Corollary 4.10. *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS such that $\mathcal{R}$ and $\mathcal{S}$ are linear and let L be a set of terms. If $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for L then $(\to_{\mathcal{R}/\mathcal{S}}, \to_{\mathcal{S}})$ is a complexity pair. Furthermore $\mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}}, L)$ is bounded by a linear polynomial for any term $t \in L$.*

Proof. Follows immediately from Theorem 4.9. ∎

To prove that a relative TRS $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for a set of terms L we construct a tree automaton $\mathcal{A}$ that is compatible with the rewrite rules of $\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})$ and $\mathsf{lift}_0(L)$. Since the set $\to^*_{\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})}(\mathsf{lift}_0(L))$ need not be regular, even for left-linear $\mathcal{R}$ and $\mathcal{S}$ and regular L (see [9]) we cannot hope to give an exact automaton construction. The general idea [8, 9] is to look for violations of the compatibility requirement: $l\sigma \to^*_\Delta q$ and $r\sigma \not\to^*_\Delta q$ for some rewrite rule $l \to r$, state substitution σ, and state q. Then we add new states and transitions to the current automaton to ensure $r\sigma \to^*_\Delta q$. After $r\sigma \to^*_\Delta q$ has been established, we repeat this process until a compatible automaton is obtained. Note that this may never happen if new states are repeatedly added.

Example 4.11. We show that the relative TRS $\mathcal{R}/\mathcal{S}$ of Example 4.6 is match-RT-bounded by constructing a compatible tree automaton. As starting point we consider the automaton

$$\mathsf{nil}_0 \to 1 \qquad \mathsf{rev}_0(1) \to 1 \qquad \mathsf{cons}_0(1,1) \to 1 \qquad \mathsf{rev}'_0(1,1) \to 1$$

which accepts the set of all ground terms over the signature $\{\mathsf{nil}, \mathsf{rev}, \mathsf{cons}, \mathsf{rev}'\}$. Since $\mathsf{rev}_0(x) \to_{\mathsf{match}(\mathcal{R})} \mathsf{rev}'_1(x, \mathsf{nil}_1)$ and $\mathsf{rev}_0(1) \to 1$, we add the transitions $\mathsf{nil}_1 \to 2$ and $\mathsf{rev}'_1(1,2) \to 1$. At next consider the rule $\mathsf{rev}'_1(\mathsf{nil}_0, y) \to_{\mathsf{match}(\mathcal{R})} y$ with $\mathsf{rev}'_1(\mathsf{nil}_0, 2) \to^* 1$. In order to solve this compatibility violation we add the transition $2 \to 1$. After that we consider the compatibility violation $\mathsf{rev}'_1(\mathsf{cons}_0(1,1), 2) \to^* 1$ but $\mathsf{rev}'_1(1, \mathsf{cons}_1(1,2)) \not\to^* 1$ caused by the rule $\mathsf{rev}'_1(\mathsf{cons}_0(x,y), z) \to_{\mathsf{match\text{-}RT}(\mathcal{S})} \mathsf{rev}'_1(y, \mathsf{cons}_1(x,z))$. In order to ensure $\mathsf{rev}'_1(1, \mathsf{cons}_1(1,2)) \to^* 1$ we reuse the transition $\mathsf{rev}'_1(1,2) \to 1$ and add the new transition $\mathsf{cons}_1(1,2) \to 2$. Finally $\mathsf{rev}'_0(\mathsf{cons}_1(x,y), z) \to_{\mathsf{match\text{-}RT}(\mathcal{S})} \mathsf{rev}'_1(y, \mathsf{cons}_1(x,z))$ and the derivation $\mathsf{rev}'_0(\mathsf{cons}_1(1,2), 1) \to^* 1$ give rise to the transition $\mathsf{cons}_1(1,1) \to 2$. After this step, the obtained tree automaton is compatible with $\mathsf{match\text{-}RT}(\mathcal{R}/\mathcal{S})$. Hence $\mathcal{R}/\mathcal{S}$ is match-RT-bounded by 1. Due to Corollary 4.10 we can conclude linear derivational complexity of $\mathcal{R}/\mathcal{S}$. We remark that the ordinary match-bound technique fails on $\mathcal{R}/\mathcal{S}$ because $\mathcal{R} \cup \mathcal{S}$ induces quadratic derivational complexity (consider the term $\mathsf{rev}^n(x)\sigma^m$ for $\sigma = \{x \mapsto \mathsf{cons}(y,x)\}$).

4.2. Beyond Complexity Pairs

An obvious question is whether it suffices to estimate the complexity of $\mathcal{R}_1/\mathcal{S}_1$ and $\mathcal{R}_2/\mathcal{S}$ (in contrast to $\mathcal{R}_2/\mathcal{S}_2$) to conclude some complexity bound for $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$. The following example by Hofbauer [15] shows that in general the complexity of $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$

might be much larger than the sum of the components; even for systems where both parts have linear derivational complexity.

Example 4.12. Consider the TRS $\mathcal{R}_1$ consisting of the single rule $\mathsf{c}(\mathsf{L}(x)) \to \mathsf{R}(x)$ and the TRS $\mathcal{R}_2$ consisting of the rules

$$\mathsf{R}(\mathsf{a}(x)) \to \mathsf{b}(\mathsf{b}(\mathsf{R}(x))) \qquad \mathsf{R}(x) \to \mathsf{L}(x) \qquad \mathsf{b}(\mathsf{L}(x)) \to \mathsf{L}(\mathsf{a}(x))$$

Here $\mathcal{S} = \varnothing$. The derivational complexity of the relative TRS $\mathcal{R}_1/\mathcal{S}_1$ is linear, due to the SLI that just counts the c's. The derivational complexity of $\mathcal{R}_2/\mathcal{S}$ is linear as well since the system is match-bounded by 2. However, the relative TRS $\mathcal{R}/\mathcal{S}$ with $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ admits exponentially long $\mathcal{R}$-derivations in the size of the starting term:

$$\begin{aligned}\mathsf{c}^n(\mathsf{L}(\mathsf{a}(x))) \to \mathsf{c}^{n-1}(\mathsf{R}(\mathsf{a}(x))) \quad &\to \mathsf{c}^{n-1}(\mathsf{b}(\mathsf{b}(\mathsf{R}(x)))) \to \ \mathsf{c}^{n-1}(\mathsf{b}(\mathsf{b}(\mathsf{L}(x))))\\ \to \mathsf{c}^{n-1}(\mathsf{b}(\mathsf{L}(\mathsf{a}(x)))) &\to \mathsf{c}^{n-1}(\mathsf{L}(\mathsf{a}(\mathsf{a}(x)))) \to^* \mathsf{L}(\mathsf{a}^{2^n}(x))\end{aligned}$$

Under certain circumstances the problem from the preceding example does not occur. Although developed for a different setting (to estimate weak dependency pair steps relative to usable rule steps) the weight gap principle of Hirokawa and Moser [12] gives a sound criterion when estimating complexity bounds as shown above. Among non-duplication of $\mathcal{R}_1$ the theorem requires a *special* linear bound on $\mathcal{R}_2$. Then the derivational complexity of $\mathcal{R}_1/\mathcal{R}_2$ determines the derivational complexity of $\mathcal{R}_1 \cup \mathcal{R}_2$. However, in contrast to runtime complexity, for derivational complexity non-duplication is no (real) restriction since any duplicating TRSs immediately admits exponentially long derivations. Because we focus on feasible upper bounds all systems considered are non-duplicating.

Theorem 4.13 (Hirokawa and Moser [12])**.** *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be TRSs, $\mathcal{R}_1$ be non-duplicating, and let $\mathcal{M}$ be an SLI such that $\mathcal{R}_2 \subseteq \succ_{\mathcal{M}}$. Then for all $\mathcal{R}_1$ and $\mathcal{M}$ there exist constants K and L such that*

$$K \cdot \mathsf{dl}(t, \to_{\mathcal{R}_1/\mathcal{R}_2}) + L \cdot |t| \geqslant \mathsf{dl}(t, \to_{\mathcal{R}_1 \cup \mathcal{R}_2})$$

whenever t is terminating on $\mathcal{R}_1 \cup \mathcal{R}_2$. ∎

The next corollary generalizes this result to the relative rewriting setting.

Corollary 4.14. *Let $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ be a relative TRS, $\mathcal{R}_1$ be non-duplicating, and let $\mathcal{M}$ be an SLI such that $\mathcal{R}_2 \subseteq \succ_{\mathcal{M}}$ and $\mathcal{S} \subseteq \succeq_{\mathcal{M}}$. Then for all $\mathcal{R}_1$ and $\mathcal{M}$ there exist constants K and L such that*

$$K \cdot \mathsf{dl}(t, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + L \cdot |t| \geqslant \mathsf{dl}(t, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}})$$

whenever t is terminating on $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$.

Proof. Follows from Theorem 4.13. ∎

An immediate consequence of the above corollary is that for any relative TRS $\mathcal{R}/\mathcal{S}$ we can shift rewrite rules in $\mathcal{R}$ that are strictly oriented by an SLI $\mathcal{M}$ into the $\mathcal{S}$ component, provided that $\mathcal{R}$ is non-duplicating and all rules in $\mathcal{S}$ behave nicely with respect to $\succeq_{\mathcal{M}}$. Note that the above corollary does not require that all rules from $\mathcal{R}$ are (strictly) oriented. This causes some kind of non-determinism which is demonstrated in the next example.

Example 4.15. Consider the TRSs $\mathcal{R} = \{\mathsf{a}(\mathsf{b}(x)) \to \mathsf{a}(\mathsf{c}(x)), \mathsf{d}(\mathsf{c}(x)) \to \mathsf{d}(\mathsf{b}(x))\}$ and $\mathcal{S} = \varnothing$. Obviously there are two SLIs that allow to preprocess $\mathcal{R}/\mathcal{S}$. Counting b's results in the relative TRS $\mathcal{R}_2/\mathcal{S}_2$ whereas counting c's yields $\mathcal{R}_1/\mathcal{S}_1$. Note that both results are different and cannot be further processed by Corollary 4.14.

5. Assessment

In the first part of this section we prove that the modular setting based on Theorem 4.4 is strictly more powerful in theory than the direct approach. (For gains in power in practice, cf. Section 7.) To make the reasoning easier we assume full (in contrast to relative) rewriting and that Theorem 4.4 has been applied iteratively, until every component $\mathcal{R}_i$ consists of a single rule. The next lemma states that if only TMIs of dimension one are employed, then in theory there is no difference in power between the two settings.

Lemma 5.1. *Let $\mathcal{R}$ be a TRS. There exists a TMI $\mathcal{M}$ of dimension one compatible with $\mathcal{R}$ if and only if there exists a family of TMIs $\mathcal{M}_i$ of dimension one such that $\mathcal{M}_i$ is compatible with $\mathcal{R}_i/\mathcal{S}_i$ for any i.*

Proof. The implication from left to right obviously holds since $\mathcal{M}$ is a suitable candidate for every $\mathcal{M}_i$. For the reverse direction we construct a TMI $\mathcal{M}$ compatible with $\mathcal{R}$ based on the family of TMIs $\mathcal{M}_i$. It is straightforward to check that defining $f_{\mathcal{M}}(x_1,\dots,x_m) = \sum_i f_{\mathcal{M}_i}(x_1,\dots,x_m)$ for any $f \in \mathcal{F}$ yields a TMI of dimension one compatible with $\mathcal{R}$. ∎

Due to Theorem 4.4 the complexity is not affected when using the modular setting. Hence when using TMIs of dimension one in theory both approaches can prove the same bounds. But experiments in Section 7 show that in practice proofs are easier to find in the modular setting since, e.g., the coefficients of the interpretations can be chosen smaller. For larger dimensions just the only-if direction of Lemma 5.1 holds. To see this we consider the TRS Strategy_removed_AG01/#4.21.[2] For this system a modular complexity proof based on TMIs of dimension two (see web site in Footnote 5 on page 398) yields a quadratic upper bound on the derivational complexity. However, due to the next lemma no such proof is possible in the direct setting. (We remark that there exist TMIs of dimension three that are compatible with this TRS).

Lemma 5.2. *The TRS* Strategy_removed_AG01/#4.21 *does not admit a TMI of dimension two compatible with it.*

Proof. To address all possible interpretations we extracted the set of constraints that represent a TMI of dimension two compatible with the TRS. MiniSmt[3] can detect unsatisfiability of these constraints. Details can be found at the web site in Footnote 5 on page 398. ∎

The next result shows that any direct proof transfers into the modular setting without increasing the bounds on the derivational complexity.

Lemma 5.3. *Let $\succ$ be a finitely branching rewrite relation and let $\mathcal{R}$ be a TRS compatible with $\succ$. Then for any i there exits a complexity pair $(\succ_i, \succeq_i)$ which is compatible with the relative TRS $\mathcal{R}_i/\mathcal{S}_i$. Furthermore for any term t we have $\mathcal{O}(\mathsf{dl}(t,\succ)) = \mathsf{max}_i\{\mathcal{O}(\mathsf{dl}(t,\succ_i))\}$.*

Proof. Fix i. Let $(\succ_i, \succeq_i)$ be $(\succ, =)$. It is easy to see that $(\succ, =)$ is a complexity pair because $\succ$ and $=$ are compatible rewrite relations. It remains to show that $\mathcal{O}(\mathsf{dl}(t,\succ)) = \mathsf{max}_i\{\mathcal{O}(\mathsf{dl}(t,\succ_i))\}$. To this end we observe that $\sum_i(\mathsf{dl}(t,(\succ_i,\succeq_i)) = n \cdot \mathsf{dl}(t,\succ)$ for all terms t. Here n denotes the number of complexity pairs. Basic properties of the $\mathcal{O}$-notation yield the desired result. ∎

[2] Labels in sans-serif font refer to TRSs from the TPDB 7.0.2, see `http://termination-portal.org`.

[3] `http://cl-informatik.uibk.ac.at/software/minismt`

As a corollary we obtain that in general the modular complexity setting is more powerful than the direct one. Even when applying just TMIs of the same dimension modularly.

Corollary 5.4. *The modular complexity setting is strictly more powerful than the direct one.*

Proof. Due to Lemmata 5.2 and 5.3. and the discussion before Lemma 5.2. ∎

As already mentioned earlier, the modular setting does not only allow to combine TMIs (of different dimensions) in one proof but even different complexity criteria. This is illustrated in the next example.

Example 5.5. Consider the TRS $\mathcal{R}$ (Transformed_CSR_04/Ex16_Luc06_GM) with the rules:

$$\begin{array}{llll} \mathsf{c} \to \mathsf{a} & \mathsf{mark}(\mathsf{a}) \to \mathsf{a} & \mathsf{g}(x,y) \to \mathsf{f}(x,y) & \\ \mathsf{c} \to \mathsf{b} & \mathsf{mark}(\mathsf{b}) \to \mathsf{c} & \mathsf{g}(x,x) \to \mathsf{g}(\mathsf{a},\mathsf{b}) & \mathsf{mark}(\mathsf{f}(x,y)) \to \mathsf{g}(\mathsf{mark}(x),y) \end{array}$$

By using Corollary 4.14 twice, we can transform $\mathcal{R}$ into the relative TRS $\mathcal{R}'/\mathcal{S}'$ where $\mathcal{R}' = \mathcal{R}_1 \cup \mathcal{R}_2$, $\mathcal{S}' = \mathcal{R} \setminus \mathcal{R}'$, $\mathcal{R}_1 = \{\mathsf{g}(x,x) \to \mathsf{g}(\mathsf{a},\mathsf{b})\}$, and $\mathcal{R}_2 = \{\mathsf{mark}(\mathsf{f}(x,y)) \to \mathsf{g}(\mathsf{mark}(x),y)\}$. After that we can apply match-RT-bounds (extended to non-duplicating TRSs) to show that the derivational complexity induced by the relative TRS $\mathcal{R}_1/\mathcal{S}_1$ is at most linear. Finally, by applying Theorem 4.4 with TMIs of dimension two to the relative TRS $\mathcal{R}_2/\mathcal{S}_2$ we obtain a quadratic upper bound on the derivational complexity of $\mathcal{R}$. Further details of the proof can be accessed from the web site accompanying this paper (see Footnote 5 on page 398). The quadratic bound is tight as $\mathcal{R}$ admits derivations

$$\mathsf{mark}^n(x)\sigma^m \to^m \mathsf{mark}^{n-1}(x)\tau^m\gamma \to^m \mathsf{mark}^{n-1}(x)\sigma^m\gamma \to^{2m(n-1)} x\sigma^m\gamma^n$$

of length $2mn$ where $\sigma = \{x \mapsto \mathsf{f}(x,y)\}$, $\tau = \{x \mapsto \mathsf{g}(x,y)\}$, and $\gamma = \{x \mapsto \mathsf{mark}(x)\}$. Last but not least we remark that none of the involved techniques can establish an upper bound on its own. In case of match-bounds this follows from the fact that $\mathcal{R}$ admits quadratic derivational complexity. The same reason also holds for Corollary 4.14 because SLIs induce linear complexity bounds. Finally, TMIs fail because they cannot orient the rewrite rule $\mathsf{g}(x,x) \to \mathsf{g}(\mathsf{a},\mathsf{b})$.

Next we consider the TRS Zantema_04/z086. The question about the derivational complexity of it has been stated as problem #105 on the RTA LooP.[4]

Example 5.6. Consider the TRS $\mathcal{R}$ (Zantema_04/z086) consisting of the rules:

$$\mathsf{a}(\mathsf{a}(x)) \to \mathsf{c}(\mathsf{b}(x)) \qquad \mathsf{b}(\mathsf{b}(x)) \to \mathsf{c}(\mathsf{a}(x)) \qquad \mathsf{c}(\mathsf{c}(x)) \to \mathsf{b}(\mathsf{a}(x))$$

Adian showed in [1] that $\mathcal{R}$ admits at most quadratic derivational complexity. Since the proof is based on a low-level reasoning on the structure of $\mathcal{R}$, it is specific to this TRS and challenging for automation. With our approach we cannot prove the quadratic bound on the derivational complexity of $\mathcal{R}$. However, Corollary 4.14 permits to establish some progress. Using an SLI counting a's and b's it suffices to determine the derivational complexity of $\mathcal{R}_3/\mathcal{S}_3$. This means that it suffices to bound how often the rule $\mathsf{c}(\mathsf{c}(x)) \to \mathsf{b}(\mathsf{a}(x))$ is applied relative to the other rules. The benefit is that now, e.g., a TMI must only orient one rule strictly and the other two rules weakly (compared to all three rules strictly). It has to be clarified if the relative formulation of the problem can be used to simplify the proof in [1].

The next example shows that although the modular approach permits lower bounds compared to the old one further criteria for splitting TRSs should be investigated.

[4] `http://rtaloop.mancoosi.univ-paris-diderot.fr`

Example 5.7. Consider the TRS $\mathcal{R}$ (SK90/4.30) consisting of the following rules:

$$\begin{array}{lll} \mathsf{f}(\mathsf{nil}) \to \mathsf{nil} & \mathsf{f}(\mathsf{nil} \circ y) \to \mathsf{nil} \circ \mathsf{f}(y) & \mathsf{f}((x \circ y) \circ z) \to \mathsf{f}(x \circ (y \circ z)) \\ \mathsf{g}(\mathsf{nil}) \to \mathsf{nil} & \mathsf{g}(x \circ \mathsf{nil}) \to \mathsf{g}(x) \circ \mathsf{nil} & \mathsf{g}(x \circ (y \circ z)) \to \mathsf{g}((x \circ y) \circ z) \end{array}$$

In [19] a TMI compatible with $\mathcal{R}$ of dimension four is given showing that the derivational complexity is bounded by a polynomial of degree four. Using Theorem 4.4 with TMIs of dimension three yields a cubic upper bound (see web site in Footnote 5 on page 398). Although our approach enables showing a lower complexity than [19] this bound is still not tight since the derivational complexity of $\mathcal{R}$ is quadratic. For this particular TRS it is easy to observe that rules defining f do not interfere with rules defining g and vice versa. Thereby we could estimate the derivational complexity of $\mathcal{R}$ by bounding the two TRSs defining f and g separately.

6. Implementation

To estimate the derivational complexity of a TRS $\mathcal{R}$ we first transform $\mathcal{R}$ into the relative TRS $\mathcal{R}/\varnothing$. If the input is already a relative TRS this step is omitted. Afterwards we try to estimate the derivational complexity of $\mathcal{R}/\mathcal{S}$ by splitting $\mathcal{R}$ into TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $\mathsf{dl}(t, \to_{\mathcal{R}_1/\mathcal{S}_1}) + \mathsf{dl}(t, \to_{\mathcal{R}_2/\mathcal{S}_2}) \geqslant \mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}})$. This is done by moving step by step those rewrite rules from $\mathcal{R}$ to $\mathcal{S}$ of which the derivational complexity can be bounded. In each step a different technique can be applied. As soon as $\mathcal{R}$ is empty, we can compute the derivational complexity of $\mathcal{R}/\mathcal{S}$ by summing up all intermediate results. In the following we describe the presented approach more formally and refer to it as the *complexity framework*.

A *complexity problem* (CP problem for short) is a pair $(\mathcal{R}/\mathcal{S}, f)$ consisting of a relative TRS $\mathcal{R}/\mathcal{S}$ and a unary function f. In order to prove the derivational complexity of a CP problem so-called *CP processors* are applied. A CP processor is a function that takes a CP problem $(\mathcal{R}/\mathcal{S}, f)$ as input and returns a new CP problem $(\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S}), f')$ as output. Here $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$. In order to be employed to prove an upper bound on the complexity of a given CP problem they need to be *sound*: $f(n) + \mathsf{dc}(n, \to_{\mathcal{R}/\mathcal{S}}) \in \mathcal{O}(f'(n) + \mathsf{dc}(n, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}))$. To ensure that a CP processor can be used to prove a lower-bound on the derivational complexity of a given CP problem it must be *complete*, i.e., $f(n) + \mathsf{dc}(n, \to_{\mathcal{R}/\mathcal{S}}) \in \Omega(f'(n) + \mathsf{dc}(n, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}))$. From the preceding sections the following CP processors can be derived.

Theorem 6.1. *The CP processor*

$$(\mathcal{R}/\mathcal{S}, f) \mapsto \begin{cases} (\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S}), f') & \text{if } \mathcal{R}_2 \subseteq \succ \text{ and } \mathcal{R}_1 \cup \mathcal{S} \subseteq \succeq \\ & \text{for some complexity pair } (\succ, \succeq) \\ (\mathcal{R}/\mathcal{S}, f) & \text{otherwise} \end{cases}$$

where $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ *and* $f'(n) = f(n) + \mathsf{dc}(n, \succ)$ *is sound.*

Proof. Follows from Corollary 3.3 and Theorem 4.4. ∎

Table 1: Derivational complexity of 1172 TRSs

	p	l	q	c	time	timeout
direct	287	180	87	20	0.72	192
modular	311	198	86	27	1.19	340

Note that the above theorem defines a general version of a CP processor based on a complexity pair $(\succ, \succeq)$. To obtain concrete instances of this CP processor one can use for example the complexity pairs from Theorem 3.4 and Corollary 4.10.

Theorem 6.2. *The CP processor*

$$(\mathcal{R}/\mathcal{S}, f) \mapsto \begin{cases} (\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S}), f') & \text{if } \mathcal{R}_1 \text{ is non-duplicating and} \\ & \mathcal{R}_2 \subseteq \succ_{\mathcal{M}} \text{ and } \mathcal{S} \subseteq \succeq_{\mathcal{M}} \text{ for some SLI } \mathcal{M} \\ (\mathcal{R}/\mathcal{S}, f) & \text{otherwise} \end{cases}$$

where $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ *and* $f'(n) = f(n) + n$ *is sound.*

Proof. Straightforward by Corollary 4.14. ∎

7. Experimental Results

The techniques described in the preceding sections are implemented in the complexity analyzer CaT (freely available from `http://cl-informatik.uibk.ac.at/software/cat`) which is built on top of TTT2 [18], a powerful termination tool for TRSs. Both tools are written in OCaml and interface MiniSat [6] and Yices [5].

Below we report on the experiments we performed with CaT on the 1172 TRSs in version 7.0.2 of the TPDB that are non-duplicating.[5] All tests have been performed on a server equipped with eight dual-core AMD Opteron® processors 885 running at a clock rate of 2.6 GHz and on 64 GB of main memory. We remark that similar results have been obtained on a dual-core laptop. Our results are summarized in Table 1. Here, direct refers to the conventional setting where all rules must be oriented at once whereas modular first transforms a TRS $\mathcal{R}$ into a relative TRS $\mathcal{R}/\varnothing$ before the CP processors from Section 6 are employed. As base methods we use the match-bounds technique as well as TMIs and arctic matrix interpretations [16] of dimensions one to three. The latter two are implemented by bit-blasting arithmetic operations to SAT [7, 16]. The coefficients of a matrix of dimension d are represented in $6 - d$ bits, one additional bit is used for intermediate results. All base methods are run in parallel, but criteria that yield larger derivational complexity are executed slightly delayed. This allows to maximize the number of low bounds.

Table 1 shows that the modular approach allows to prove significantly more polynomial bounds (column p) for the systems; furthermore these bounds are also smaller. Here column l refers to linear, q to quadratic, and c to cubic derivational complexity. We also list the average time (in seconds) needed for finding a bound and the number of timeouts, i.e, when the tool did not deliver an answer within 60 seconds. The modular setting is slower since there typically more proofs are required to succeed. However, we anticipate that by making use of incremental SAT solvers (for the matrix methods) the time can be reduced.

[5] Full details available from `http://cl-informatik.uibk.ac.at/software/cat/modular`.

Table 2: Termination competition 2009 (derivational complexity)

	points	p	l	q	c	r
CaT	540	137	84	41	7	5
Matchbox	397	102	44	52	5	1
TcT	380	109	32	69	8	0

For a comparison of our method with other tools we refer the reader to the international termination competition (referenced in Footnote 1 on page 385). In 2008 and 2009, CaT won all categories related to derivational complexity analysis. Note that in 2009 it used a slightly weaker implementation of Theorem 4.4 than presented here and did not implement Corollary 4.14; especially the latter speeds up proofs. Some statistics from the 2009 competition are listed in Table 2. The total points are computed by a scoring scheme that favors tools that yield small upper bounds. The column r indicates polynomial bounds of degree four.

8. Conclusion

In this paper we have introduced a modular approach for estimating the derivational complexity of TRSs by considering relative rewriting. We showed how existing criteria (for full rewriting) can be lifted into the relative setting without much effort. The modular approach is easy to implement and has been proved strictly more powerful than traditional methods in theory and practice. Since the modular method allows to combine different criteria, typically smaller complexity bounds are achieved. All our results directly extend to more restrictive notions of complexity, e.g., runtime complexity (see below). We stress that as a by-product of our investigations we obtained an alternative approach to [23] that can be used to prove relative termination using match-bounds. It remains open which of the two approaches is more powerful. Finally we remark that our setting allows a more fine-grained complexity analysis, i.e., while traditionally a quadratic derivational complexity ensures that any rule is applied at most quadratically often, our approach can make different statements about single rules. Hence even if a proof attempt does not succeed completely, it may highlight the problematic rules.

As related work we mention [15] which also considers relative rewriting for complexity analysis. However, there the complexity of $\mathcal{R}_1 \cup \mathcal{R}_2$ is investigated by considering $\mathcal{R}_1/\mathcal{R}_2$ and $\mathcal{R}_2$. Hence [15] also gives rise to a modular reasoning but the obtained complexities are typically beyond polynomials. For runtime complexity analysis Hirokawa and Moser [12,13] consider weak dependency pair steps relative to the usable rules, i.e., $\mathsf{WDP}(\mathcal{R})/\mathsf{UR}(\mathcal{R})$. However, since in the current formulation of weak dependency pairs some complexity might be hidden in the usable rules they do not really obtain a relative problem. As a consequence they can only apply restricted criteria for the usable rules. Note that our approach can directly be used to show bounds on $\mathsf{WDP}(\mathcal{R})/\mathsf{UR}(\mathcal{R})$ by considering $\mathsf{WDP}(\mathcal{R})\cup\mathsf{UR}(\mathcal{R})$. Due to Corollary 4.14 this problem can be transformed into an (unrestricted) relative problem $\mathsf{WDP}(\mathcal{R})/\mathsf{UR}(\mathcal{R})$ whenever the constraints in [12] are satisfied. Moreover, if somehow the *problematic* usable rules could be determined and shifted into the $\mathsf{WDP}(\mathcal{R})$ component, then this *improved* version of weak dependency pairs corresponds to a relative problem without additional restrictions, admitting further benefit from our contributions.

We plan to investigate if the arctic matrix method also satisfies the weight gap principle. Other promising techniques that (might) allow a modular processing of subsystems comprise criteria for constructor sharing TRSs [20]. Last but not least it seems feasible to combine the approach for relative match-bounds introduced in [23] with the one presented in Section 4.1. By doing so, a stronger version of relative match-bounds could be obtained.

Acknowledgments. We thank Johannes Waldmann for communicating Example 4.12.

References

[1] Adian, S.I.: Upper bound on the derivational complexity in some word rewriting system. Doklady Math. 80(2), 679–683 (2009)
[2] Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. TCS 236(1-2), 133–178 (2000)
[3] Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, Cambridge (1998)
[4] Dershowitz, N., Manna, Z.: Proving termination with multiset orderings. Comm. ACM 22(8), 465–476 (1979)
[5] Dutertre, B., de Moura, L.: A fast linear-arithmetic solver for DPLL(T). In: CAV 2006. LNCS, vol. 4144, pp. 81–94 (2006)
[6] Eén, N., Sörensson, N.: An extensible SAT-solver. In: SAT 2004. LNCS, vol. 2919, pp. 502–518 (2004)
[7] Endrullis, J., Waldmann, J., Zantema, H.: Matrix interpretations for proving termination of term rewriting. JAR 40(2-3), 195–220 (2008)
[8] Genet, T.: Decidable approximations of sets of descendants and sets of normal forms. In: RTA 1998. LNCS, vol. 1379, pp. 151–165 (1998)
[9] Geser, A., Hofbauer, D., Waldmann, J., Zantema, H.: On tree automata that certify termination of left-linear term rewriting systems. I&C 205(4), 512–534 (2007)
[10] Geser, A.: Relative termination. PhD thesis, Universität Passau, Germany (1990). Available as: Report 91-03, Ulmer Informatik-Berichte, Universität Ulm, 1991
[11] Hirokawa, N., Middeldorp, A.: Automating the dependency pair method. I&C 199(1-2), 172–199 (2005)
[12] Hirokawa, N., Moser, G.: Automated complexity analysis based on the dependency pair method. In: IJCAR 2008. LNCS, vol. 5195, pp. 364–379 (2008)
[13] Hirokawa, N., Moser, G.: Complexity, graphs, and the dependency pair method. In: LPAR 2008. LNCS (LNAI), vol. 5330, pp. 652–666 (2008)
[14] Hofbauer, D., Lautemann, C.: Termination proofs and the length of derivations (preliminary version). In: RTA 1989. LNCS, vol. 355, pp. 167–177 (1989)
[15] Hofbauer, D., Waldmann, J.: Complexity bounds from relative termination proofs. Talk at the Workshop on Proof Theory and Rewriting, Obergurgl (2006). Available from `http://www.imn.htwk-leipzig.de/~waldmann/talk/06/rpt/rel/main.pdf`
[16] Koprowski, A., Waldmann, J.: Max/plus tree automata for termination of term rewriting. AC 19(2), 357–392 (2009)
[17] Korp, M., Middeldorp, A.: Match-bounds revisited. I&C 207(11), 1259–1283 (2009)
[18] Korp, M., Sternagel, C., Zankl, H., Middeldorp, A.: Tyrolean Termination Tool 2. In: RTA 2009. LNCS, vol. 5595, pp. 295–304 (2009)
[19] Moser, G., Schnabl, A., Waldmann, J.: Complexity analysis of term rewriting based on matrix and context dependent interpretations. In: FSTTCS 2008. LIPIcs, vol. 2, pp. 304–315 (2008)
[20] Ohlebusch, E.: On the modularity of confluence of constructor-sharing term rewriting systems. In: CAAP 1994. LNCS, vol. 787, pp. 261–275 (1994)
[21] Terese: Term Rewriting Systems. vol. 55 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (2003)
[22] Thiemann, R.: The DP Framework for Proving Termination of Term Rewriting. PhD thesis, RWTH Aachen (2007). Available as technical report AIB-2007-17
[23] Waldmann, J.: Weighted automata for proving termination of string rewriting. J. Autom. Lang. Comb. 12(4), 545–570 (2007)