Automated Complexity Analysis Based on the Dependency Pair Method*

Nao Hirokawa[1] and Georg Moser[2]

[1] School of Information Science, Japan Advanced Institute of Science and Technology, Japan, hirokawa@jaist.ac.jp
[2] Institute of Computer Science, University of Innsbruck, Austria
georg.moser@uibk.ac.at

Abstract. In this paper, we present a variant of the dependency pair method for analysing runtime complexities of term rewrite systems automatically. This method is easy to implement, but significantly extends the analytic power of existing direct methods. Our findings extend the class of TRSs whose linear or quadratic runtime complexity can be detected automatically. We provide ample numerical data for assessing the viability of the method.

1 Introduction

Term rewriting is a conceptually simple but powerful abstract model of computation that underlies much of declarative programming. In order to assess the complexity of a (terminating) term rewrite system (TRS for short) it is natural to look at the maximal length of derivation sequences, as suggested by Hofbauer and Lautemann in [1]. More precisely, the *derivational complexity function* with respect to a (terminating and finitely-branching) TRS $\mathcal{R}$ relates the length of the longest derivation sequence to the size of the initial term. For direct termination techniques it is often possible to establish upper-bounds on the growth rate of the derivational complexity function from the termination proof of $\mathcal{R}$, see for example [1,2,3,4,5,6].

However, if one is interested in methods that induce feasible (i.e., polynomial) complexity, the existing body of research is not directly applicable. On one hand this is due to the fact that for standard techniques the derivational complexity cannot be contained by polynomial growth rates. (See [6] for the exception to the rule.) Already termination proofs by polynomial interpretations induce a double-exponential upper-bound on the derivational complexity, cf. [1]. On the other hand this is—to some extent—the consequence of the *definition* of derivational complexity as this measure does not discriminate between different types of initial terms, while in modelling declarative programs the type of the initial term is usually quite restrictive. The following example clarifies the situation.

* This research is partly supported by FWF (Austrian Science Fund) project P20133, Leading Project e-Society (MEXT of Japan), and STARC.

Example 1. Consider the TRS $\mathcal{R}$

$$\begin{array}{lll} 1: & x - \mathsf{0} \to x & 3: \quad \mathsf{0} \div \mathsf{s}(y) \to \mathsf{0} \\ 2: & \mathsf{s}(x) - \mathsf{s}(y) \to x - y & 4: \quad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x-y) \div \mathsf{s}(y)) \end{array}$$

Although the functions *computed* by $\mathcal{R}$ are obviously feasible this is not reflected in the derivational complexity of $\mathcal{R}$. Consider rule 4, which we abbreviate as $C[y] \to D[y, y]$. Since the maximal derivation length starting with $C^n[y]$ equals 2^{n-1} for all $n > 0$, $\mathcal{R}$ admits (at least) exponential derivational complexity.

After a moment one sees that this behaviour is forced upon us, as the TRS $\mathcal{R}$ may duplicate variables, i.e., $\mathcal{R}$ is *duplicating*. Furthermore, in general the applicability of the above results is typically limited to simple termination. (But see [4,5,6] for exceptions to this rule.) To overcome the first mentioned restriction we propose to study *runtime complexities* of rewrite systems. The *runtime complexity function* with respect to a TRS $\mathcal{R}$ relates the length of the longest derivation sequence to the size of the arguments of the initial term, where the arguments are supposed to be in normal form. In order to overcome the second restriction, we base our study on a fresh analysis of the *dependency pair method*. The dependency pair method [7] is a powerful (and easily automatable) method for proving termination of term rewrite systems. In contrast to the above cited direct termination methods, this technique is a *transformation* technique, allowing for applicability beyond simple termination.

Studying (runtime) complexities induced by the dependency pair method is challenging. Below we give an (easy) example showing that the direct translations of original theorems formulated in the context of termination analysis is destined to failure in the context of runtime complexity analysis. If one recalls that the dependency pair method is based on the observation that from an arbitrary non-terminating term one can extract a minimal non-terminating subterm, this is not surprising. Through a very careful investigation of the original formulation of the dependency pair method (see [7,8], but also [9]), we establish a runtime complexity analysis based on the dependency pair method. In doing so, we introduce *weak dependency pairs* and *weak innermost dependency pairs* as a general adaption of dependency pairs to (innermost) runtime complexity analysis. Here the *innermost* runtime complexity function with respect to a TRS $\mathcal{R}$ relates the length of the longest innermost derivation sequence to the size of the arguments of the initial term, where again the arguments are supposed to be in normal form.

Our main result shows how natural improvements of the dependency pair method, like *usable rules*, *reduction pairs*, and *argument filterings* become applicable in this context. Moreover, for innermost rewriting, we establish an easy criteria to decide when weak innermost dependency pairs can be replaced by "standard" dependency pairs without introducing fallacies. Thus we establish (for the first time) a method to analyse the derivation length induced by the (standard) dependency pair method for innermost rewriting. We have implemented the technique and experimental evidence shows that the use of weak dependency pairs significantly increases the applicability of the body of existing

results on the estimation of derivation length via termination techniques. In particular, our findings extend the class of TRSs whose linear or quadratic runtime complexity can be detected automatically.

The remainder of this paper is organised as follows. In the next section we recall basic notions and starting points of this paper. Section 3 and 4 introduce weak dependency pairs and discuss the employability of the usable rule criteria. In Section 5 we show how to estimate runtime complexities through relative rewriting and in Section 6 we state our Main Theorem. The presented technique has been implemented and we provide ample numerical data for assessing the viability of the method. This evidence can be found in Section 7. Finally in Section 8 we conclude and mention possible future work.

2 Preliminaries

We assume familiarity with term rewriting [10,11] but briefly review basic concepts and notations. Let $\mathcal{V}$ denote a countably infinite set of variables and $\mathcal{F}$ a signature. The set of terms over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F},\mathcal{V})$. The *root symbol* of a term t is either t itself, if $t \in \mathcal{V}$, or the symbol f, if $t = f(t_1,\ldots,t_n)$. The *set of position* $\mathcal{P}\mathsf{os}(t)$ of a term t is defined as usual. We write $\mathcal{P}\mathsf{os}_{\mathcal{G}}(t) \subseteq \mathcal{P}\mathsf{os}(t)$ for the set of positions of subterms, whose root symbol is contained in $\mathcal{G} \subseteq \mathcal{F}$. The subterm relation is denoted as $\trianglelefteq$. $\mathcal{V}\mathsf{ar}(t)$ denotes the set of variables occurring in a term t and the *size* $|t|$ of a term is defined as the number of symbols in t.

A *term rewrite system* (*TRS* for short) $\mathcal{R}$ over $\mathcal{T}(\mathcal{F},\mathcal{V})$ is a *finite* set of rewrite rules $l \to r$, such that $l \notin \mathcal{V}$ and $\mathcal{V}\mathsf{ar}(l) \supseteq \mathcal{V}\mathsf{ar}(r)$. The smallest rewrite relation that contains $\mathcal{R}$ is denoted by $\to_{\mathcal{R}}$. The transitive closure of $\to_{\mathcal{R}}$ is denoted by $\to^+_{\mathcal{R}}$, and its transitive and reflexive closure by $\to^*_{\mathcal{R}}$. We simply write $\to$ for $\to_{\mathcal{R}}$ if $\mathcal{R}$ is clear from context. A term $s \in \mathcal{T}(\mathcal{F},\mathcal{V})$ is called a *normal form* if there is no $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ such that $s \to t$. With $\mathcal{NF}(\mathcal{R})$ we denote the set of all normal forms of a term rewrite system $\mathcal{R}$. The *innermost rewrite relation* $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ of a TRS $\mathcal{R}$ is defined on terms as follows: $s \xrightarrow{\mathrm{i}}_{\mathcal{R}} t$ if there exist a rewrite rule $l \to r \in \mathcal{R}$, a context C, and a substitution σ such that $s = C[l\sigma]$, $t = C[r\sigma]$, and all proper subterms of $l\sigma$ are normal forms of $\mathcal{R}$. The set of defined function symbols is denoted as $\mathcal{D}$, while the constructor symbols are collected in $\mathcal{C}$. We call a term $t = f(t_1,\ldots,t_n)$ *basic* if $f \in \mathcal{D}$ and $t_i \in \mathcal{T}(\mathcal{C},\mathcal{V})$ for all $1 \leqslant i \leqslant n$. A TRS $\mathcal{R}$ is called *duplicating* if there exists a rule $l \to r \in \mathcal{R}$ such that a variable occurs more often in r than in l. We call a TRS *terminating* if no infinite rewrite sequence exists. Let s and t be terms. If exactly n steps are performed to rewrite s to t we write $s \to^n t$. The *derivation length* of a terminating term t with respect to a TRS $\mathcal{R}$ and rewrite relation $\to_{\mathcal{R}}$ is defined as: $\mathsf{dl}(s,\to_{\mathcal{R}}) = \max\{n \mid \exists t\ s \to^n t\}$. Let $\mathcal{R}$ be a TRS and T be a set of terms. The *runtime complexity function with respect to a relation* $\to$ *on* T is defined as follows:

$$\mathsf{rc}(n,T,\to) = \max\{\mathsf{dl}(t,\to) \mid t \in T \text{ and } |t| \leqslant n\}\ .$$

In particular we are interested in the (innermost) runtime complexity with respect to $\to_{\mathcal{R}}$ ($\xrightarrow{\mathsf{i}}_{\mathcal{R}}$) on the set $\mathcal{T}_{\mathsf{b}}$ of all *basic* terms.[3] More precisely, the *runtime complexity function* (with respect to $\mathcal{R}$) is defined as $\mathsf{rc}_{\mathcal{R}}(n) := \mathsf{rc}(n, \mathcal{T}_{\mathsf{b}}, \to_{\mathcal{R}})$ and we define the *innermost runtime complexity function* as $\mathsf{rc}^{\mathsf{i}}_{\mathcal{R}}(n) := \mathsf{rc}(n, \mathcal{T}_{\mathsf{b}}, \xrightarrow{\mathsf{i}}_{\mathcal{R}})$. Finally, the *derivational complexity function* (with respect to $\mathcal{R}$) becomes definable as follows: $\mathsf{dc}_{\mathcal{R}}(n) = \mathsf{rc}(n, \mathcal{T}, \to_{\mathcal{R}})$, where $\mathcal{T}$ denotes the set of all terms $\mathcal{T}(\mathcal{F}, \mathcal{V})$. We sometimes say the (innermost) runtime complexity of $\mathcal{R}$ is *linear*, *quadratic*, or *polynomial* if $\mathsf{rc}^{(\mathsf{i})}_{\mathcal{R}}(n)$ is bounded linearly, quadratically, or polynomially in n, respectively. Note that the derivational complexity and the runtime complexity of a TRS $\mathcal{R}$ may be quite different: In general it is not possible to bound $\mathsf{dc}_{\mathcal{R}}$ polynomially in $\mathsf{rc}_{\mathcal{R}}$, as witnessed by Example 1 and the observation that the runtime complexity of $\mathcal{R}$ is linear (see Example 34, below).

A *proper order* is a transitive and irreflexive relation and a *preorder* is a transitive and reflexive relation. A proper order $\succ$ is *well-founded* if there is no infinite decreasing sequence $t_1 \succ t_2 \succ t_3 \cdots$. A well-founded proper order that is also a rewrite relation is called a *reduction order*. We say a reduction order $\succ$ and a TRS $\mathcal{R}$ are *compatible* if $\mathcal{R} \subseteq \succ$. It is well-known that a TRS is terminating if and only if there exists a compatible reduction order. An *$\mathcal{F}$-algebra* $\mathcal{A}$ consists of a carrier set A and a collection of interpretations $f_{\mathcal{A}}$ for each function symbol in $\mathcal{F}$. A *well-founded* and *monotone* algebra (*WMA* for short) is a pair $(\mathcal{A}, >)$, where $\mathcal{A}$ is an algebra and $>$ is a well-founded partial order on A such that every $f_{\mathcal{A}}$ is monotone in all arguments. An *assignment* $\alpha\colon \mathcal{V} \to A$ is a function mapping variables to elements in the carrier. A WMA naturally induces a proper order $>_{\mathcal{A}}$ on terms: $s >_{\mathcal{A}} t$ if $[\alpha]_{\mathcal{A}}(s) > [\alpha]_{\mathcal{A}}(t)$ for all assignments $\alpha\colon \mathcal{V} \to A$.

3 The Dependency Pair Method

The purpose of this section is to take a fresh look at the dependency pair method from the point of complexity analysis. Familiarity with [7,9] will be helpful. The dependency pair method for termination analysis is based on the observation that from an arbitrary non-terminating term one can extract a minimal non-terminating subterm. For complexity analysis we employ a similar observation: From a given term t one can extract a list of subterms whose sum of the derivation lengths is equal to the derivational length of t.

Let X be a set of symbols. We write $C\langle t_1, \ldots, t_n\rangle_X$ to denote $C[t_1, \ldots, t_n]$, whenever $\mathsf{root}(t_i) \in X$ for all $1 \leqslant i \leqslant n$ and C is an n-hole context containing no X-symbols. (Note that the context C may be degenerate and doesn't contain a hole $\Box$ or it may be that C is a hole.) Then, every term t can be uniquely written in the form $C\langle t_1, \ldots, t_n\rangle_X$.

Lemma 2. *Let t be a terminating term, and let σ be a substitution. Then $\mathsf{dl}(t\sigma, \to_{\mathcal{R}}) = \sum_{1 \leqslant i \leqslant n} \mathsf{dl}(t_i\sigma, \to_{\mathcal{R}})$, whenever $t = C\langle t_1, \ldots, t_n\rangle_{\mathcal{D}\cup\mathcal{V}}$.*

[3] We can replace $\mathcal{T}_{\mathsf{b}}$ by the set of terms $f(t_1, \ldots, t_n)$ with $f \in \mathcal{D}$, whose arguments t_i are in normal form, while keeping all results in this paper.

We define the function COM as a mapping from tuples of terms to terms as follows: $\mathrm{COM}(t_1,\dots,t_n)$ is t_1 if $n = 1$, and $c(t_1,\dots,t_n)$ otherwise. Here c is a fresh n-ary function symbol called *compound symbol*. The above lemma motivates the next definition of *weak dependency pairs*.

Definition 3. *Let t be a term. We set $t^\sharp := t$ if $t \in \mathcal{V}$, and $t^\sharp := f^\sharp(t_1,\dots,t_n)$ if $t = f(t_1,\dots,t_n)$. Here $f^\sharp$ is a new n-ary function symbol called* dependency pair symbol. *For a signature $\mathcal{F}$, we define $\mathcal{F}^\sharp = \mathcal{F} \cup \{f^\sharp \mid f \in \mathcal{F}\}$. Let $\mathcal{R}$ be a TRS. If $l \to r \in \mathcal{R}$ and $r = C\langle u_1,\dots,u_n\rangle_{\mathcal{D}\cup\mathcal{V}}$ then the rewrite rule $l^\sharp \to \mathrm{COM}(u_1^\sharp,\dots,u_n^\sharp)$ is called a* weak dependency pair *of $\mathcal{R}$. The set of all weak dependency pairs is denoted by $\mathsf{WDP}(\mathcal{R})$.*

Example 4 (continued from Example 1). The set $\mathsf{WDP}(\mathcal{R})$ consists of the next four weak dependency pairs:

$$\begin{array}{lll} 5\colon & x -^\sharp 0 \to x & 7\colon \quad 0 \div^\sharp \mathsf{s}(y) \to \mathsf{c}_1 \\ 6\colon & \mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y & 8\colon \quad \mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to (x-y) \div^\sharp \mathsf{s}(y) \end{array}$$

Lemma 5. *Let $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ be a terminating term with $\mathsf{root}(t) \in \mathcal{D}$. We have $\mathsf{dl}(t,\to_{\mathcal{R}}) = \mathsf{dl}(t^\sharp,\to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}})$.*

Proof. We show $\mathsf{dl}(t,\to_{\mathcal{R}}) \leqslant \mathsf{dl}(t^\sharp,\to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}})$ by induction on $\ell = \mathsf{dl}(t,\to_{\mathcal{R}})$. If $\ell = 0$, the inequality is trivial. Suppose $\ell > 0$. Then there exists a term u such that $t \to_{\mathcal{R}} u$ and $\mathsf{dl}(u,\to_{\mathcal{R}}) = \ell - 1$. We distinguish two cases depending on the rewrite position p.

- If p is a position below the root, then clearly $\mathsf{root}(u) = \mathsf{root}(t) \in \mathcal{D}$ and $t^\sharp \to_{\mathcal{R}} u^\sharp$. The induction hypothesis yields $\mathsf{dl}(u,\to_{\mathcal{R}}) \leqslant \mathsf{dl}(u^\sharp,\to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}})$, and we obtain $\ell \leqslant \mathsf{dl}(t^\sharp,\to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}})$.
- If p is a root position, then there exist a rewrite rule $l \to r \in \mathcal{R}$ and a substitution σ such that $t = l\sigma$ and $u = r\sigma$. We have $r = C\langle u_1,\dots,u_n\rangle_{\mathcal{D}\cup\mathcal{V}}$ and thus by definition $l^\sharp \to \mathrm{COM}(u_1^\sharp,\dots,u_n^\sharp) \in \mathsf{WDP}(\mathcal{R})$ such that $t^\sharp = l^\sharp\sigma$. Now, either $u_i \in \mathcal{V}$ or $\mathsf{root}(u_i) \in \mathcal{D}$ for every $1 \leqslant i \leqslant n$. Suppose $u_i \in \mathcal{V}$. Then $u_i^\sharp\sigma = u_i\sigma$ and clearly no dependency pair symbol can occur and thus,

  $$\mathsf{dl}(u_i\sigma,\to_{\mathcal{R}}) = \mathsf{dl}(u_i^\sharp\sigma,\to_{\mathcal{R}}) = \mathsf{dl}(u_i^\sharp\sigma,\to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}})\,.$$

  Otherwise, $\mathsf{root}(u_i) \in \mathcal{D}$ and thus $u_i^\sharp\sigma = (u_i\sigma)^\sharp$. We have $\mathsf{dl}(u_i\sigma,\to_{\mathcal{R}}) \leqslant \mathsf{dl}(u,\to_{\mathcal{R}}) < l$, and conclude $\mathsf{dl}(u_i\sigma,\to_{\mathcal{R}}) \leqslant \mathsf{dl}(u_i^\sharp\sigma,\to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}})$ by the induction hypothesis. Therefore,

  $$\begin{aligned} \ell = \mathsf{dl}(u,\to_{\mathcal{R}}) + 1 &= \sum_{1\leqslant i\leqslant n} \mathsf{dl}(u_i\sigma,\mathcal{R}) + 1 \leqslant \sum_{1\leqslant i\leqslant n} \mathsf{dl}(u_i^\sharp\sigma,\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}) + 1 \\ &\leqslant \mathsf{dl}(\mathrm{COM}(u_1^\sharp,\dots,u_n^\sharp)\sigma,\to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}}) + 1 = \mathsf{dl}(t^\sharp,\to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}})\,. \end{aligned}$$

  Here we used Lemma 2 for the second equality.

Note that t is $\mathcal{R}$-reducible if and only if $t^\sharp$ is $\mathsf{WDP}(\mathcal{R}) \cup \mathcal{R}$-reducible. Hence as t is terminating, $t^\sharp$ is terminating on $\to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}}$. Thus, similarly, $\mathsf{dl}(t, \to_{\mathcal{R}}) \geqslant \mathsf{dl}(t^\sharp, \to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}})$ is shown by induction on $\mathsf{dl}(t^\sharp, \to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}})$. □

Lemma 6. *Let t be a terminating term and σ a substitution such that $x\sigma$ is a normal form of $\mathcal{R}$ for all $x \in \mathcal{V}\mathsf{ar}(t)$. Then $\mathsf{dl}(t\sigma, \to_{\mathcal{R}}) = \sum_{1\leqslant i\leqslant n} \mathsf{dl}(t_i\sigma, \to_{\mathcal{R}})$, whenever $t = C\langle t_1, \ldots, t_n\rangle_{\mathcal{D}}$.*

Definition 7. *Let $\mathcal{R}$ be a TRS. If $l \to r \in \mathcal{R}$ and $r = C\langle u_1, \ldots, u_n\rangle_{\mathcal{D}}$ then the rewrite rule $l^\sharp \to \mathrm{COM}(u_1^\sharp, \ldots, u_n^\sharp)$ is called a* weak innermost dependency pair *of $\mathcal{R}$. The set of all weak innermost dependency pairs is denoted by $\mathsf{WIDP}(\mathcal{R})$.*

Example 8 (continued from Example 1). The set $\mathsf{WIDP}(\mathcal{R})$ consists of the next four weak dependency pairs (with respect to $\xrightarrow{i}$):

$$x -^\sharp \mathsf{0} \to \mathsf{c}_1 \qquad \mathsf{0} \div^\sharp y \to \mathsf{c}_2$$
$$\mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y \qquad \mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to (x - y) \div^\sharp \mathsf{s}(y)$$

The next lemma adapts Lemma 5 to innermost rewriting.

Lemma 9. *Let t be an innermost terminating term in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ with $\mathsf{root}(t) \in \mathcal{D}$. We have $\mathsf{dl}(t, \xrightarrow{i}_{\mathcal{R}}) = \mathsf{dl}(t^\sharp, \xrightarrow{i}_{\mathsf{WIDP}(\mathcal{R})\cup\mathcal{R}})$.*

We conclude this section by discussing the applicability of standard dependency pairs ([7]) in complexity analysis. For that we recall the standard definition of dependency pairs.

Definition 10 ([7]). *The set $\mathsf{DP}(\mathcal{R})$ of (standard)* dependency pairs *of a TRS $\mathcal{R}$ is defined as $\{l^\sharp \to u^\sharp \mid l \to r \in \mathcal{R}, u \trianglelefteq r, \mathsf{root}(u) \in \mathcal{D}\}$.*

The next example shows that Lemma 5 (Lemma 9) does not hold if we replace weak (innermost) dependency pairs with standard dependency pairs.

Example 11. Consider the one-rule TRS $\mathcal{R}$: $\mathsf{f}(\mathsf{s}(x)) \to \mathsf{g}(\mathsf{f}(x), \mathsf{f}(x))$. $\mathsf{DP}(\mathcal{R})$ is the singleton of $\mathsf{f}^\sharp(\mathsf{s}(x)) \to \mathsf{f}^\sharp(x)$. Let $t_n = \mathsf{f}(\mathsf{s}^n(x))$ for each $n \geqslant 0$. Since $t_{n+1} \to_{\mathcal{R}} \mathsf{g}(t_n, t_n)$ holds for all $n \geqslant 0$, it is easy to see $\mathsf{dl}(t_{n+1}, \to_{\mathcal{R}}) \geqslant 2^n$, while $\mathsf{dl}(t_{n+1}^\sharp, \to_{\mathsf{DP}(\mathcal{R})\cup\mathcal{R}}) = n$.

Hence, in general we cannot replace weak dependency pairs with (standard) dependency pairs. However, if we restrict our attention to innermost rewriting, we can employ dependency pairs in complexity analysis without introducing fallacies, when specific conditions are met.

Lemma 12. *Let t be an innermost terminating term with $\mathsf{root}(t) \in \mathcal{D}$. If all compound symbols in $\mathsf{WIDP}(\mathcal{R})$ are nullary, $\mathsf{dl}(t, \xrightarrow{i}_{\mathcal{R}}) \leqslant \mathsf{dl}(t^\sharp, \xrightarrow{i}_{\mathsf{DP}(\mathcal{R})\cup\mathcal{R}}) + 1$ holds.*

Example 13 (continued from Example 8). The occurring compound symbols are nullary. $\mathsf{DP}(\mathcal{R})$ consists of the next three dependency pairs:

$$\mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y \qquad \mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to x -^\sharp y$$
$$\mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to (x - y) \div^\sharp \mathsf{s}(y)$$

4 Usable Rules

In the previous section, we studied the dependency pair method in the light of complexity analysis. Let $\mathcal{R}$ be a TRS and $\mathcal{P}$ a set of weak dependency pairs, weak innermost dependency pairs, or standard dependency pairs of $\mathcal{R}$. Lemmata 5, 9, and 12 describe a strong connection between the length of derivations in the original TRSs $\mathcal{R}$ and the transformed (and extended) system $\mathcal{P} \cup \mathcal{R}$. In this section we show how we can simplify the new TRS $\mathcal{P} \cup \mathcal{R}$ by employing *usable rules*.

Definition 14. *We write $f \rhd_{\mathrm{d}} g$ if there exists a rewrite rule $l \to r \in \mathcal{R}$ such that $f = \mathsf{root}(l)$ and g is a defined function symbol in $\mathcal{F}\mathsf{un}(r)$. For a set $\mathcal{G}$ of defined function symbols we denote by $\mathcal{R}{\restriction}\mathcal{G}$ the set of rewrite rules $l \to r \in \mathcal{R}$ with $\mathsf{root}(l) \in \mathcal{G}$. The set $\mathcal{U}(t)$ of usable rules of a term t is defined as $\mathcal{R}{\restriction}\{g \mid f \rhd_{\mathrm{d}}^* g$ for some $f \in \mathcal{F}\mathsf{un}(t)\}$. Finally, if $\mathcal{P}$ is a set of (weak) dependency pairs then $\mathcal{U}(\mathcal{P}) = \bigcup_{l \to r \in \mathcal{P}} \mathcal{U}(r)$.*

Example 15 (continued from Examples 4 and 8). The sets of usable rules are equal for the weak dependency pairs and for the weak innermost dependency pairs, i.e., we have $\mathcal{U}(\mathsf{WDP}(\mathcal{R})) = \mathcal{U}(\mathsf{WIDP}(\mathcal{R})) = \{1, 2\}$.

The usable rule criterion in termination analysis (cf. [12,9]) asserts that a non-terminating rewrite sequence of $\mathcal{R} \cup \mathsf{DP}(\mathcal{R})$ can be transformed into a non-terminating rewrite sequence of $\mathcal{U}(\mathsf{DP}(\mathcal{R})) \cup \mathsf{DP}(\mathcal{R}) \cup \{\mathsf{g}(x, y) \to x, \mathsf{g}(x, y) \to y\}$, where g is a fresh function symbol. Because $\mathcal{U}(\mathsf{DP}(\mathcal{R}))$ is a (small) subset of $\mathcal{R}$ and most termination methods can handle $\mathsf{g}(x, y) \to x$ and $\mathsf{g}(x, y) \to y$ easily, the termination analysis often becomes easy by switching the target of analysis from the former TRS to the latter TRS. Unfortunately the transformation used in [12,9] increases the size of starting terms, therefore we cannot adopt this transformation approach. Note, however that the usable rule criteria for innermost termination [8] can be directly applied in the context of complexity analysis. Nevertheless, one may show a new type of usable rule criterion by exploiting the basic property of a starting term. Recall that $\mathcal{T}_{\mathsf{b}}$ denotes the set of basic terms; we set $\mathcal{T}_{\mathsf{b}}^{\sharp} = \{t^{\sharp} \mid t \in \mathcal{T}_{\mathsf{b}}\}$.

Lemma 16. *Let $\mathcal{P}$ be a set of (weak) dependency pairs and let $(t_i)_{i=0,1,\ldots}$ be a (finite or infinite) derivation of $\mathcal{R} \cup \mathcal{P}$. If $t_0 \in \mathcal{T}_{\mathsf{b}}^{\sharp}$ then $(t_i)_{i=0,1,\ldots}$ is a derivation of $\mathcal{U}(\mathcal{P}) \cup \mathcal{P}$.*

Proof. Let $\mathcal{G}$ be the set of all non-usable symbols with respect to $\mathcal{P}$. We write $P(t)$ if $t|_q \in \mathcal{NF}(\mathcal{R})$ for all $q \in \mathcal{P}\mathsf{os}_{\mathcal{G}}(t)$. Since $t_i \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} t_{i+1}$ holds whenever $P(t_i)$ and $t_i \to_{\mathcal{R} \cup \mathcal{P}} t_{i+1}$, it is sufficient to show $P(t_i)$ for all i. We perform induction on i.

1. Assume $i = 0$. Since $t_0 \in \mathcal{T}_{\mathsf{b}}^{\sharp}$, we have $t_0 \in \mathcal{NF}(\mathcal{R})$ and thus $t|_p \in \mathcal{NF}(\mathcal{R})$ for all positions p. The assertion P follows trivially.

2. Suppose $i > 0$. By induction hypothesis, there exist $l \to r \in \mathcal{U}(\mathcal{P}) \cup \mathcal{P}$, $p \in \mathcal{P}\mathsf{os}(t_{i-1})$, and a substitution σ such that $t_{i-1}|_p = l\sigma$ and $t_i|_p = r\sigma$. In order to show property P for t_i, we fix a position $q \in \mathcal{G}$. We have to show $t_i|_q \in \mathcal{NF}(\mathcal{R})$. We distinguish three cases:
   - Suppose that q is above p. Then $t_{i-1}|_q$ is reducible, but this contradicts the induction hypothesis $P(t_{i-1})$.
   - Suppose p and q are parallel but distinct. Since $t_{i-1}|_q = t_i|_q \in \mathcal{NF}(\mathcal{R})$ holds, we obtain $P(t_i)$.
   - Otherwise, q is below p. Then, $t_i|_q$ is a subterm of $r\sigma$. Because r contains no $\mathcal{G}$-symbols by the definition of usable symbols, $t_i|_q$ is a subterm of $x\sigma$ for some $x \in \mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(l)$. Therefore, $t_i|_q$ is also a subterm of t_{i-1}, from which $t_i|_q \in \mathcal{NF}(\mathcal{R})$ follows. We obtain $P(t_i)$.

□

The following theorem follows from Lemmata 5, 9, and 12 in conjunction with the above Lemma 16. It adapts the usable rule criteria to complexity analysis.[4]

Theorem 17. *Let $\mathcal{R}$ be a TRS and let $t \in \mathcal{T}_\mathsf{b}$. If t is terminating with respect to $\to$ then $\mathsf{dl}(t, \to) \leqslant \mathsf{dl}(t^\sharp, \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}})$, where $\to$ denotes $\to_\mathcal{R}$ or $\xrightarrow{i}_\mathcal{R}$ depending on whether $\mathcal{P} = \mathsf{WDP}(\mathcal{R})$ or $\mathcal{P} = \mathsf{WIDP}(\mathcal{R})$. Moreover, suppose all compound symbols in $\mathsf{WIDP}(\mathcal{R})$ are nullary then $\mathsf{dl}(t, \xrightarrow{i}_\mathcal{R}) \leqslant \mathsf{dl}(t^\sharp, \to_{\mathcal{U}(\mathsf{DP}(\mathcal{R})) \cup \mathsf{DP}(\mathcal{R})}) + 1$.*

It is worth stressing that it is (often) easier to analyse the complexity of $\mathcal{U}(\mathcal{P}) \cup \mathcal{P}$ than the complexity of $\mathcal{R}$. To clarify the applicability of the theorem in complexity analysis, we consider two restrictive classes of polynomial interpretations, whose definitions are motivated by [13].

A polynomial $P(x_1, \ldots, x_n)$ (over the natural numbers) is called *strongly linear* if $P(x_1, \ldots, x_n) = x_1 + \cdots + x_n + c$ where $c \in \mathbb{N}$. A polynomial interpretation is called *linear restricted* if all constructor symbols are interpreted by strongly linear polynomials and all other function symbols by a linear polynomial. If on the other hand the non-constructor symbols are interpreted by quadratic polynomials, the polynomial interpretation is called *quadratic restricted*. Here a polynomial is *quadratic* if it is a sum of monomials of degree at most 2. It is easy to see that if a TRS $\mathcal{R}$ is compatible with a linear or quadratic restricted interpretation, the runtime complexity of $\mathcal{R}$ is linear or quadratic, respectively (see also [13]).

Corollary 18. *Let $\mathcal{R}$ be a TRS and let $\mathcal{P} = \mathsf{WDP}(\mathcal{R})$ or $\mathcal{P} = \mathsf{WIDP}(\mathcal{R})$. If $\mathcal{U}(\mathcal{P}) \cup \mathcal{P}$ is compatible with a linear or quadratic restricted interpretation, the (innermost) runtime complexity function $\mathsf{rc}^{(\mathrm{i})}_\mathcal{R}$ with respect to $\mathcal{R}$ is linear or quadratic, respectively. Moreover, suppose all compound symbols in $\mathsf{WIDP}(\mathcal{R})$ are nullary and $\mathcal{U}(\mathsf{DP}(\mathcal{R})) \cup \mathsf{DP}(\mathcal{R})$ is compatible with a linear (quadratic) restricted interpretation, then $\mathcal{R}$ admits at most linear (quadratic) innermost runtime complexity.*

[4] Note that Theorem 17 only holds for *basic* terms $t \in \mathcal{T}^\sharp_\mathsf{b}$. In order to show this, we need some additional technical lemmas, which are the subject of the next section.

Proof. Let $\mathcal{R}$ be a TRS. For simplicity we suppose $\mathcal{P} = \mathsf{WDP}(\mathcal{R})$ and assume the existence of a linear restricted interpretation $\mathcal{A}$, compatible with $\mathcal{U}(\mathcal{P}) \cup \mathcal{P}$. Clearly this implies the well-foundedness of the relation $\to_{\mathcal{U}(\mathcal{P})\cup\mathcal{P}}$, which in turn implies the well-foundedness of $\to_{\mathcal{R}}$, cf. Lemma 16. Hence Theorem 17 is applicable and we conclude $\mathsf{dl}(t, \to_{\mathcal{R}}) \leqslant \mathsf{dl}(t^\sharp, \to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}})$. On the other hand, compatibility with $\mathcal{A}$ implies that $\mathsf{dl}(t^\sharp, \to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}}) = \mathcal{O}(|t^\sharp|)$. As $|t^\sharp| = |t|$, we can combine these equalities to conclude linear runtime complexity of $\mathcal{R}$. □

5 The Weight Gap Principle

We recall the notion of *relative rewriting* ([14,11]).

Definition 19. *Let $\mathcal{R}$ and $\mathcal{S}$ be TRSs. We write $\to_{\mathcal{R}/\mathcal{S}}$ for $\to_{\mathcal{S}}^* \cdot \to_{\mathcal{R}} \cdot \to_{\mathcal{S}}^*$ and we call $\to_{\mathcal{R}/\mathcal{S}}$ the* relative rewrite relation *of $\mathcal{R}$ over $\mathcal{S}$.*[5]

Since $\mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}})$ corresponds to the number of $\to_{\mathcal{R}}$-steps in a maximal derivation of $\to_{\mathcal{R}\cup\mathcal{S}}$ from t, we easily see the bound $\mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}}) \leqslant \mathsf{dl}(t, \to_{\mathcal{R}\cup\mathcal{S}})$. In this section we study this opposite, i.e., we figure out a way to give an upper-bound of $\mathsf{dl}(t, \to_{\mathcal{R}\cup\mathcal{S}})$ by a function of $\mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}})$.

First we introduce the key ingredient, *strongly linear interpretations*, a very restrictive form of polynomial interpretations. Let $\mathcal{F}$ denote a signature. A *strongly linear interpretation* (*SLI* for short) is a WMA $(\mathcal{A}, \succ)$ that satisfies the following properties: (i) the carrier of $\mathcal{A}$ is the set of natural numbers $\mathbb{N}$, (ii) all interpretation functions $f_{\mathcal{A}}$ are strongly linear, (iii) the proper order $\succ$ is the standard order $>$ on $\mathbb{N}$. Note that an SLI $\mathcal{A}$ is conceivable as a weight function. We define the maximum weight $\mathsf{M}_{\mathcal{A}}$ of $\mathcal{A}$ as $\max\{f_{\mathcal{A}}(0,\ldots,0) \mid f \in \mathcal{F}\}$. Let $\mathcal{A}$ denote an SLI, let α_0 denote the assignment mapping any variable to 0, i.e., $\alpha_0(x) = 0$ for all $x \in \mathcal{V}$, and let t be a term. We write $[t]$ as an abbreviation for $[\alpha_0]_{\mathcal{A}}(t)$.

Lemma 20. *Let $\mathcal{A}$ be an SLI and let t be a term. Then $[t] \leqslant \mathsf{M}_{\mathcal{A}} \cdot |t|$ holds.*

Proof. By induction on t. If $t \in \mathcal{V}$ then $[t] = 0 \leqslant \mathsf{M}_{\mathcal{A}} \cdot |t|$. Otherwise, suppose $t = f(t_1,\ldots,t_n)$, where $f_{\mathcal{A}}(x_1,\ldots,x_n) = x_1 + \ldots + x_n + c$. By the induction hypothesis and $c \leqslant \mathsf{M}_{\mathcal{A}}$ we obtain the following inequalities:

$$\begin{aligned}[t] = f_{\mathcal{A}}([t_1],\ldots,[t_n]) &\leqslant [t_1] + \cdots + [t_n] + c \\ &\leqslant \mathsf{M}_{\mathcal{A}} \cdot |t_1| + \cdots + \mathsf{M}_{\mathcal{A}} \cdot |t_n| + \mathsf{M}_{\mathcal{A}} = \mathsf{M}_{\mathcal{A}} \cdot |t| \,.\end{aligned}$$

□

The conception of strongly linear interpretations as weight functions allows us to study (possible) weight increase throughout a rewrite derivation. This observation is reflected in the next definition.

[5] Note that $\to_{\mathcal{R}/\mathcal{S}} = \to_{\mathcal{R}}$, if $\mathcal{S} = \varnothing$.

Definition 21. *Let $\mathcal{A}$ be an algebra and let $\mathcal{R}$ be a TRS. The* weight gap $\Delta(\mathcal{A},\mathcal{R})$ of $\mathcal{A}$ with respect to $\mathcal{R}$ *is defined on* $\mathbb{N}$ *as follows:* $\Delta(\mathcal{A},\mathcal{R}) = \max\{[r] \dot{-} [l] \mid l \to r \in \mathcal{R}\}$, *where* $\dot{-}$ *is defined as usual:* $m \dot{-} n := \max\{m-n, 0\}$

The following *weight gap principle* is a direct consequence of the definitions.

Lemma 22. *Let $\mathcal{R}$ be a TRS and $\mathcal{A}$ an SLI. If $s \to_{\mathcal{R}} t$ then $[s] + \Delta(\mathcal{A},\mathcal{R}) \geqslant [t]$.*

We stress that the lemma does not require any condition. Indeed, the implication in the lemma holds even if TRS $\mathcal{R}$ is *not* compatible with a strongly linear interpretation. This principle brings us to the next theorem.

Theorem 23. *Let $\mathcal{R}$ and $\mathcal{S}$ be TRSs, and $\mathcal{A}$ an SLI compatible with $\mathcal{S}$. Then we have $\mathsf{dl}(t, \to_{\mathcal{R}\cup\mathcal{S}}) \leqslant (1 + \Delta(\mathcal{A},\mathcal{R})) \cdot \mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}}) + \mathsf{M}_{\mathcal{A}} \cdot |t|$, whenever t is terminating on $\mathcal{R} \cup \mathcal{S}$.*

Proof. Let $m = \mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}})$, let $n = |t|$, and set $\Delta = \Delta(\mathcal{A},\mathcal{R})$. Any derivation of $\to_{\mathcal{R}\cup\mathcal{S}}$ is representable as follows

$$s_0 \to_{\mathcal{S}}^{k_0} t_0 \to_{\mathcal{R}} s_1 \to_{\mathcal{S}}^{k_1} t_1 \to_{\mathcal{R}} \cdots \to_{\mathcal{S}}^{k_m} t_m \ ,$$

and without loss of generality we may assume that the derivation is maximal. We observe the next two facts.

(a) $k_i \leqslant [s_i] - [t_i]$ holds for all $0 \leqslant i \leqslant m$. This is because $[s] \geqslant [t] + 1$ whenever $s \to_{\mathcal{S}} t$ by the assumption $\mathcal{S} \subseteq >_{\mathcal{A}}$, and we have $s_i \to_{\mathcal{S}}^{k_i} t_i$.
(b) $[s_{i+1}] - [t_i] \leqslant \Delta$ holds for all $0 \leqslant i < m$ as due to Lemma 22 we have $[t_i] + \Delta \geqslant [s_{i+1}]$.

We obtain the following inequalities:

$$\begin{aligned}
\mathsf{dl}(s_0, \to_{\mathcal{R}\cup\mathcal{S}}) &= m + k_0 + \cdots + k_m \\
&\leqslant m + ([s_0] - [t_0]) + \cdots + ([s_m] - [t_m]) \\
&= m + [s_0] + ([s_1] - [t_0]) + \cdots + ([s_m] - [t_{m-1}]) - [t_m] \\
&\leqslant m + [s_0] + m\Delta - [t_m] \\
&\leqslant m + [s_0] + m\Delta \\
&\leqslant m + \mathsf{M}_{\mathcal{A}} \cdot n + m\Delta = (1+\Delta)m + \mathsf{M}_{\mathcal{A}} \cdot n \ .
\end{aligned}$$

Here we used (a) m-times in the second line, (b) $m-1$-times in the fourth line, and Lemma 20 in the last line. □

The next example clarifies that the conditions expressed in Theorem 23 are optimal: We cannot replace the assumption that the algebra $\mathcal{A}$ is *strongly* linear with a weaker assumption: Already if $\mathcal{A}$ is a linear polynomial interpretation, the derivation height of $\mathcal{R} \cup \mathcal{S}$ cannot be bounded *polynomially* in $\mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}})$ and $|t|$ alone.

Example 24. Consider the TRSs $\mathcal{R}$

$$\begin{aligned} \mathsf{exp}(\mathsf{0}) &\to \mathsf{s}(\mathsf{0}) & \mathsf{d}(\mathsf{0}) &\to \mathsf{0} \\ \mathsf{exp}(\mathsf{r}(x)) &\to \mathsf{d}(\mathsf{exp}(x)) & \mathsf{d}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{s}(\mathsf{d}(x))) \end{aligned}$$

This TRS formalises the exponentiation function. Setting $t_n = \mathsf{exp}(\mathsf{r}^n(\mathsf{0}))$ we obtain $\mathsf{dl}(t_n, \to_{\mathcal{R}}) \geqslant 2^n$ for each $n \geqslant 0$. Thus the runtime complexity of $\mathcal{R}$ is (at least) exponential. In order to show the claim, we split $\mathcal{R}$ into two TRSs $\mathcal{R}_1 = \{\mathsf{exp}(\mathsf{0}) \to \mathsf{s}(\mathsf{0}), \mathsf{exp}(\mathsf{r}(x)) \to \mathsf{d}(\mathsf{exp}(x))\}$ and $\mathcal{R}_2 = \{\mathsf{d}(\mathsf{0}) \to \mathsf{0}, \mathsf{d}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{d}(x)))\}$. Then it is easy to verify that the next linear polynomial interpretation $\mathcal{A}$ is compatible with $\mathcal{R}_2$: $\mathsf{0}_{\mathcal{A}} = 0$, $\mathsf{d}_{\mathcal{A}}(x) = 3x$, and $\mathsf{s}_{\mathcal{A}}(x) = x+1$. Moreover an upper-bound of $\mathsf{dl}(t_n, \to_{\mathcal{R}_1/\mathcal{R}_2})$ can be estimated by using the following polynomial interpretation $\mathcal{B}$: $\mathsf{0}_{\mathcal{B}} = 0$, $\mathsf{d}_{\mathcal{B}}(x) = \mathsf{s}_{\mathcal{B}}(x) = x$, and $\mathsf{exp}_{\mathcal{B}}(x) = \mathsf{r}_{\mathcal{B}}(x) = x + 1$. Since $\to_{\mathcal{R}_1} \subseteq >_{\mathcal{B}}$ and $\to^*_{\mathcal{R}_2} \subseteq \geqslant_{\mathcal{B}}$ hold, we have $\to_{\mathcal{R}_1/\mathcal{R}_2} \subseteq >_{\mathcal{B}}$. Hence $\mathsf{dl}(t_n, \to_{\mathcal{R}_1/\mathcal{R}_2}) \leqslant [\alpha_0]_{\mathcal{B}}(t_n) = n + 2$. But clearly from this we cannot conclude a polynomial bound on the derivation length of $\mathcal{R}_1 \cup \mathcal{R}_2 = \mathcal{R}$, as the runtime complexity of $\mathcal{R}$ is exponential, at least.

To conclude this section, we show that Theorem 17 can only hold for *basic* terms $t \in \mathcal{T}^\sharp_{\mathsf{b}}$.

Example 25. Consider the one-rule TRS $\mathcal{R} = \{\mathsf{a}(\mathsf{b}(x)) \to \mathsf{b}(\mathsf{b}(\mathsf{a}(x)))\}$ from [15, Example 2.50]. It is not difficult to see that $\mathsf{dl}(\mathsf{a}^n(\mathsf{b}(x)), \to_{\mathcal{R}}) = 2^n - 1$, see [4]. The set $\mathsf{WDP}(\mathcal{R})$ consists of just one dependency pair $\mathsf{a}^\sharp(\mathsf{b}(x)) \to \mathsf{a}^\sharp(x))$. In particular the set of usable rules is empty. The following SLI $\mathcal{A}$ is compatible with $\mathsf{WDP}(\mathcal{R})$: $\mathsf{a}^\sharp_{\mathcal{A}}(x) = \mathsf{a}_{\mathcal{A}}(x) = x$ and $\mathsf{b}_{\mathcal{A}}(x) = 1$. Hence, due to Lemma 20 we can conclude the existence of a constant K such that $\mathsf{dl}(t^\sharp, \to_{\mathsf{WDP}(\mathcal{R})}) \leqslant K \cdot |t|$. Due to Theorem 17 we conclude linear runtime complexity of $\mathcal{R}$.

6 Reduction Pairs and Argument Filterings

In this section we study the consequences of combining Theorem 17 and Theorem 23. In doing so, we adapt reduction pairs and argument filterings ([7]) to runtime complexity analysis. Let $\mathcal{R}$ be a TRS, and let $\mathcal{A}$ be a strongly linear interpretation and suppose we consider weak, weak innermost, or (standard) dependency pairs $\mathcal{P}$. If $\mathcal{U}(\mathcal{P}) \subseteq >_{\mathcal{A}}$ then there exist constants $K, L \geqslant 0$ (depending on $\mathcal{P}$ and $\mathcal{A}$ only) such that

$$\mathsf{dl}(t, \to_{\mathcal{R}}) \leqslant K \cdot \mathsf{dl}(t^\sharp, \to_{\mathcal{P}/\mathcal{U}(\mathcal{P})}) + L \cdot |t^\sharp| \,,$$

for all terminating basic terms $t \in \mathcal{T}_{\mathsf{b}}$. This follows from the combination of Theorems 17 and 23. Thus, in order to estimate the derivation length of t with respect to $\mathcal{R}$ it suffices to estimate the maximal $\mathcal{P}$ steps, i.e., we have to estimate $\mathsf{dl}(t^\sharp, \to_{\mathcal{P}/\mathcal{U}(\mathcal{P})})$ suitably. Consider a maximal derivation $(t_i)_{i=0,\dots,n}$ of $\to_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$ with $t_0 = t^\sharp$. For every $0 \leqslant i < n$ there exist terms u_i and v_i such that

$$t_i \to^*_{\mathcal{U}(\mathcal{P})} u_i \to_{\mathcal{P}} v_i \to^*_{\mathcal{U}(\mathcal{P})} t_{i+1} \,. \tag{1}$$

Let $\gtrsim$ and $\succ$ be a pair of orders with $\gtrsim \cdot \succ \cdot \gtrsim \subseteq \succ$. If $t_i \gtrsim u_i \succ v_i \gtrsim t_{i+1}$ holds for all $0 \leqslant i < n$, we obtain $t^\sharp = t_0 \succ t_1 \succ \cdots \succ t_n$. Therefore, $\mathsf{dl}(t^\sharp, \to_{\mathcal{P}/\mathcal{U}(\mathcal{P})})$ can be bounded in the maximal length of $\succ$-descending steps. We formalise these observations through the use of *reduction pairs* and *collapsible orders*.

Definition 26. *Let $\mathcal{R}$ be a TRS, let $\mathcal{P}$ a set of weak dependency pairs of $\mathcal{R}$ and let G denote a mapping associating a term (over $\mathcal{F}^\sharp$ and $\mathcal{V}$) and a proper order $\succ$ with a natural number. An order $\succ$ on terms is* G-collapsible *for a TRS $\mathcal{R}$ if $s \to_{\mathcal{P} \cup \mathcal{U}(\mathcal{P})} t$ and $s \succ t$ implies $\mathsf{G}(s, \succ) > \mathsf{G}(t, \succ)$. An order $\succ$ is* collapsible *for a TRS $\mathcal{R}$, if there is a mapping G such that $\succ$ is G-collapsible for $\mathcal{R}$.*

Note that most reduction orders are collapsible. For instance, if $\mathcal{A}$ is a polynomial interpretation then $>_\mathcal{A}$ is collapsible, as witnessed by the evaluation function $[\alpha_0]_\mathcal{A}$. Furthermore, simplification orders like MPO, LPO and KBO are collapsible (cf. [2,3,5]).[6]

Definition 27. *A* rewrite preorder *is a preorder on terms which is closed under contexts and substitutions. A* reduction pair $(\gtrsim, \succ)$ *consists of a rewrite preorder $\gtrsim$ and a compatible well-founded order $\succ$ which is closed under substitutions. Here compatibility means the inclusion $\gtrsim \cdot \succ \cdot \gtrsim \subseteq \succ$. A reduction pair $(\gtrsim, \succ)$ is called* collapsible *for a TRS $\mathcal{R}$ if $\succ$ is collapsible for $\mathcal{R}$.*

Recall the derivation in (1): Due to compound symbols the rewrite step $u_i \to_\mathcal{P} v_i$ may take place below the root. Hence $\mathcal{P} \subseteq \succ$ does not ensure $u_i \succ v_i$. To address this problem we introduce a notion of *safety* that is based on the next definitions.

Definition 28. *The set $\mathcal{T}^\sharp_\mathsf{c}$ is inductively defined as follows (i) $\mathcal{T}^\sharp_\mathsf{b} \subseteq \mathcal{T}^\sharp_\mathsf{c}$ and (ii) $c(t_1, \ldots, t_n) \in \mathcal{T}^\sharp_\mathsf{c}$, whenever $t_1, \ldots, t_n \in \mathcal{T}^\sharp_\mathsf{c}$ and c a compound symbol.*

Definition 29. *A proper order $\succ$ on $\mathcal{T}^\sharp_\mathsf{c}$ is called* safe *if $c(s_1, \ldots, s_i, \ldots, s_n) \succ c(s_1, \ldots, t, \ldots, s_n)$ for all n-ary compound symbols c and all terms $s_1, \ldots, s_n, t$ with $s_i \succ t$. A reduction pair $(\gtrsim, \succ)$ is called* safe *if $\succ$ is safe.*

Lemma 30. *Let $\mathcal{P}$ be a set of weak, weak innermost, or standard dependency pairs, and $(\gtrsim, \succ)$ be a safe reduction pair such that $\mathcal{U}(\mathcal{P}) \subseteq \gtrsim$ and $\mathcal{P} \subseteq \succ$. If $s \in \mathcal{T}^\sharp_\mathsf{c}$ and $s \to_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t$ then $s \succ t$ and $t \in \mathcal{T}^\sharp_\mathsf{c}$.*

Employing Theorem 17, Theorem 23, and Lemma 30 we arrive at our Main Theorem.

Theorem 31. *Let $\mathcal{R}$ be a TRS, let $\mathcal{A}$ an SLI, let $\mathcal{P}$ be the set of weak, weak innermost, or (standard) dependency pairs, and let $(\gtrsim, \succ)$ be a safe and G-collapsible reduction pair such that $\mathcal{U}(\mathcal{P}) \subseteq \gtrsim$ and $\mathcal{P} \subseteq \succ$. If in addition $\mathcal{U}(\mathcal{P}) \subseteq >_\mathcal{A}$ then for any $t \in \mathcal{T}_\mathsf{b}$, we have $\mathsf{dl}(t, \to) \leqslant p(\mathsf{G}(t^\sharp, >_\mathcal{A}), |t|)$, where $p(m, n) := (1 + \Delta(\mathcal{A}, \mathcal{P})) \cdot m + \mathsf{M}_\mathcal{A} \cdot n$ and $\to$ denotes $\to_\mathcal{R}$ or $\xrightarrow{i}_\mathcal{R}$ depending on whether $\mathcal{P} = \mathsf{WDP}(\mathcal{R})$ or $\mathcal{P} = \mathsf{WIDP}(\mathcal{R})$. Moreover if all compound symbols in $\mathsf{WIDP}(\mathcal{R})$ are nullary we have $\mathsf{dl}(t, \xrightarrow{i}_\mathcal{R}) \leqslant p(\mathsf{G}(t^\sharp, >_\mathcal{A}), |t|) + 1$.*

[6] On the other hand it is easy to construct non-collapsible orders: Suppose we extend the natural numbers $\mathbb{N}$ by a non-standard element ∞ such that for any $n \in \mathbb{N}$ we set $\infty > n$. Clearly we cannot collapse ∞ to a natural number.

Proof. First, observe that the assumptions imply that any basic term $t \in \mathcal{T}_\mathsf{b}$ is terminating with respect to $\mathcal{R}$. This is a direct consequence of Lemma 16 and Lemma 30 in conjunction with the assumptions of the theorem. Without loss of generality, we assume $\mathcal{P} = \mathsf{WDP}(\mathcal{P})$. By Theorem 17 and 23 we obtain:

$$\begin{aligned}\mathsf{dl}(t, \to) &\leqslant \mathsf{dl}(t^\sharp, \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}}) \leqslant p(\mathsf{dl}(t^\sharp, \to_{\mathcal{P}/\mathcal{U}(\mathcal{P})}), |t^\sharp|) \\ &\leqslant p(\mathsf{G}(t^\sharp, >_\mathcal{A}), |t^\sharp|) = p(\mathsf{G}(t^\sharp, >_\mathcal{A}), |t|) \,.\end{aligned}$$

In the last line we exploit that $|t^\sharp| = |t|$. □

Note that there exists a subtle disadvantage of Theorem 31 in comparison to Theorem 17. Since the Main Theorem requires compatibility of usable rules with some strongly linear interpretation, all usable rules must be non-duplicating. This is not necessary to meet the requirements of Theorem 17.

In order to construct safe reduction pairs one may use *safe algebras*, i.e., weakly monotone well-founded algebras $(\mathcal{A}, \succ)$ such that the interpretations of compound symbols are strictly monotone with respect to $\succ$. Another way is to apply an argument filtering to a reduction pair.

Definition 32. *An* argument filtering *for a signature $\mathcal{F}$ is a mapping π that assigns to every n-ary function symbol $f \in \mathcal{F}$ an argument position $i \in \{1, \dots, n\}$ or a (possibly empty) list $[i_1, \dots, i_m]$ of argument positions with $1 \leqslant i_1 < \cdots < i_m \leqslant n$. The signature $\mathcal{F}_\pi$ consists of all function symbols f such that $\pi(f)$ is some list $[i_1, \dots, i_m]$, where in $\mathcal{F}_\pi$ the arity of f is m. Every argument filtering π induces a mapping from $\mathcal{T}(\mathcal{F}, \mathcal{V})$ to $\mathcal{T}(\mathcal{F}_\pi, \mathcal{V})$, also denoted by π:*

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

An argument filtering π is called safe *if $\pi(c) = [1, \dots, n]$ for all n-ary compound symbol. For a relation R on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ we define R^π on $\mathcal{T}(\mathcal{F}_\pi, \mathcal{V})$ as follows: $s \mathrel{R^\pi} t$ if and only if $\pi(s) \mathrel{R} \pi(t)$.*

Lemma 33. *If $(\mathcal{A}, \succ)$ is a safe algebra then $(\succcurlyeq_\mathcal{A}, >_\mathcal{A})$ is a safe reduction pair. Furthermore, $(\gtrsim^\pi, \succ^\pi)$ is a safe reduction pair if $(\gtrsim, \succ)$ is a reduction pair and π is a safe argument filtering.*

The below given example applies the Main Theorem to the motivating Example 1 introduced in Section 1.

Example 34 (continued from Example 4.). By taking the SLI $\mathcal{A}$ with interpretation $\mathsf{0}_\mathcal{A} = 0$, $\mathsf{s}_\mathcal{A}(x) = x+1$, and $x -_\mathcal{A} y = x+y+1$, we obtain $\mathcal{U}(\mathsf{WDP}(\mathcal{R})) \subseteq >_\mathcal{A}$. Moreover, we take the safe algebra $\mathcal{B}$ with $\mathsf{0}_\mathcal{B} = 0$, $\mathsf{s}_\mathcal{B}(x) = x + 2$, $x -_\mathcal{B} y = x -^\sharp_\mathcal{B} y = x \mathbin{\dot{-}^\sharp_\mathcal{B}} y = x + 1$, and $(\mathsf{c}_1)_\mathcal{B} = 0$. $\mathcal{B}$ interprets $\mathsf{WDP}(\mathcal{R})$ and $\mathcal{U}(\mathsf{WDP}(\mathcal{R}))$ as follows:

$$\begin{array}{lll} 1\colon\ x + 1 \geqslant x & 5\colon\ x + 1 > x & 7\colon\quad 1 > 0 \\ 2\colon\ x + 3 \geqslant x + 1 & 6\colon\ x + 3 > x + 1 & 8\colon\ x + 3 > x + 2\,. \end{array}$$

Therefore, $\mathsf{WDP}(\mathcal{R}) \subseteq >_{\mathcal{B}}$ and $\mathcal{U}(\mathsf{WDP}(\mathcal{R})) \subseteq \geqslant_{\mathcal{B}}$ hold. Hence, the runtime complexity of $\mathcal{R}$ for full rewriting is linear.

Following the pattern of the proof of Corollary 18 it is an easy exercise to extend Theorem 31 to a method for complexity analysis. In the above example, we have already used the easy fact that (obvious extensions of) linear restricted interpretation may be used as safe reduction pairs.

Corollary 35. *Let $\mathcal{R}$ be a TRS, let $\mathcal{A}$ be an SLI, let $\mathcal{P}$ be the set of weak, weak innermost, or standard dependency pairs, where the compound symbols in $\mathsf{WIDP}(\mathcal{R})$ are nullary, if $\mathcal{P} = \mathsf{DP}(\mathcal{R})$. Moreover let $\mathcal{B}$ be a linear or quadratic restricted interpretation such that $(\geqslant_{\mathcal{B}}, >_{\mathcal{B}})$ forms a safe reduction pair with $\mathcal{U}(\mathcal{P}) \subseteq \geqslant_{\mathcal{B}}$ and $\mathcal{P} \subseteq >_{\mathcal{B}}$. If $\mathcal{U}(\mathcal{P}) \subseteq >_{\mathcal{A}}$ then the (innermost) runtime complexity function $\mathsf{rc}^{(\mathsf{i})}_{\mathcal{R}}$ with respect to $\mathcal{R}$ is linear or quadratic, respectively.*

Note that if $\mathcal{U}(\mathcal{P}) = \varnothing$, the compatibility of $\mathcal{U}(\mathcal{P})$ with an SLI is trivially satisfiable. In this special case by taking the SLI $\mathcal{A}$ that interprets all symbols with the zero function, we obtain $\mathsf{dl}(t, \to) \leqslant \mathsf{G}(t^\sharp, >_{\mathcal{A}})$ because $\Delta(\mathcal{A}, \varnothing) = \mathsf{M}_{\mathcal{A}} = 0$. As a consequence of Theorem 31 and Lemma 12 we obtain the following corollary.

Corollary 36. *Let $\mathcal{R}$ be a TRS, let $\mathcal{A}$ be an SLI, let all compound symbols in $\mathsf{WIDP}(\mathcal{R})$ be nullary and let $\mathcal{B}$ be a linear or quadratic restricted interpretation such that $(\geqslant_{\mathcal{B}}, >_{\mathcal{B}})$ forms a reduction pair with $\mathcal{U}(\mathsf{DP}(\mathcal{R})) \subseteq \geqslant_{\mathcal{B}}$ and $\mathsf{DP}(\mathcal{R}) \subseteq >_{\mathcal{B}}$. If in addition $\mathcal{U}(\mathsf{DP}(\mathcal{R})) \subseteq >_{\mathcal{A}}$ then the innermost runtime complexity function $\mathsf{rc}^{\mathsf{i}}_{\mathcal{R}}$ with respect to $\mathcal{R}$ is linear or quadratic, respectively.*

Corollary 36 establishes (for the first time) a method to analyse the derivation length induced by the standard dependency pair method for innermost rewriting. More general, if all compound symbols in $\mathsf{WIDP}(\mathcal{R})$ are nullary and there exists a collapsible reduction pair $(\gtrsim, \succ)$ such that $\mathcal{U}(\mathcal{P}) \subseteq \gtrsim$ and $\mathcal{P} \subseteq \succ$, then the innermost runtime complexity of $\mathcal{R}$ is linear in the maximal length of $\succ$-descending steps. Clearly for *string rewriting* (cf. [11]) the compound symbols in $\mathsf{WIDP}(\mathcal{R})$ are always nullary and the conditions works quite well for TRSs, too (see Section 7).

7 Experiments

We implemented a complexity analyser based on syntactical transformations for dependency pairs and usable rules together with polynomial orders (based on [16]). To deal efficiently with polynomial interpretations, the issuing constraints are encoded in *propositional logic* in a similar spirit as in [17]. Assignments are found by employing a state-of-the-art SAT solver, in our case MiniSat[7]. Furthermore, strongly linear interpretations are handled by a decision procedure for Presburger arithmetic. As suitable test bed we used the rewrite systems in the Termination Problem Data Base version 4.0.[8] This test bed comprises 1679

[7] `http://minisat.se/`.
[8] `http://www.lri.fr/~marche/tpdb/`

Table 1. Experimental Results for TRSs

Linear Runtime Complexities							
	full rewriting				*innermost rewriting*		
	LC	Cor. 18	Cor. 35	both	Cor. 18 (DP)	Cor. 35 (DP)	both
success	139	138	119	161	143 (135)	128 (113)	170
	15	*21*	*18*	*33*	*21* (*20*)	*21* (*15*)	*34*
failure	1529	1501	1560	1478	1495 (1502)	1550 (1565)	1467
	1564	*2517*	*152*	*2612*	*2489* (*2593*)	*180* (*149*)	*2580*
timeout	11	40	0	40	41 (42)	1 (1)	42

Quadratic Runtime Complexities							
	full rewriting				*innermost rewriting*		
	QC	Cor. 18	Cor. 35	both	Cor. 18 (DP)	Cor. 35 (DP)	both
success	176	169	124	188	168 (152)	125 (109)	189
	473	*598*	*254*	*781*	*564* (*457*)	*237* (*128*)	*755*
failure	702	657	1486	601	654 (707)	1486 (1502)	593
	2569	*2591*	*527*	*2700*	*2522* (*2461*)	*602* (*546*)	*2570*
timeout	799	852	69	890	856 (816)	68 (68)	896

TRSs, including 358 TRSs for innermost rewriting. The presented tests were performed single-threaded on a 1.50 GHz Intel® Core™ Duo Processor L2300 and 1.5 GB of memory. For each system we used a timeout of 30 seconds, the times in the tables are given in seconds. Table 1 summarises the results of the conducted experiments.[9] Text written in *italics* below the number of successes or failures indicates total time of success cases or failure cases, respectively.[10]

We use the following abbreviations: The method LC (QC) refers to compatibility with *linear (quadratic) restricted interpretation*, cf. Section 3. In interpreting defined and dependency pair functions, we restrict the search to polynomials in the range $\{0, 1, \ldots, 5\}$. The upper half of Table 1 shows the experimental results for linear runtime complexities based on LC. The columns marked "Cor. 18" and "Cor. 35" refer to the applicability of the respective corollaries. In the column marked "both" we indicate the results, we obtain when we first try to apply Corollary 35 and if this fails Corollary 18. The lower half summarises experimental results for quadratic runtime complexities based on QC. On the studied test bed there are 1567 TRSs such that one may switch from $\mathsf{WIDP}(\mathcal{R})$ to $\mathsf{DP}(\mathcal{R})$. For the individual tests, we indicated the results in parentheses for this versions of Corollary 18 and Corollary 35.

8 Conclusion

In this paper we studied the runtime complexity of rewrite systems. We have established a variant of the dependency pair method that is applicable in this context and is easily mechanisable. In particular our findings extend the class of

[9] For full experimental evidence see `http://www.jaist.ac.jp/~hirokawa/08a/`.
[10] Sum of numbers in each column may be less than 1679 because of stack overflow.

TRSs whose *linear* or *quadratic* runtime complexity can be detected automatically. We provided ample numerical data for assessing the viability of the method. To conclude, we mention possible future work. In the experiments presented, we have restricted our attention to interpretation based methods inducing linear or quadratic (innermost) runtime complexity. Recently in [18] a restriction of the multiset path order, called *polynomial path order* has been introduced that induces polynomial runtime complexity. In future work we will test to what extent this is effectively combinable with our Main Theorem. Furthermore, we strive to extend the approach presented here to handle dependency graphs [7].

References

1. Hofbauer, D., Lautemann, C.: Termination proofs and the length of derivations. In: Proc. 3rd RTA. Volume 355 of LNCS. (1989) 167–177
2. Hofbauer, D.: Termination proofs by multiset path orderings imply primitive recursive derivation lengths. TCS **105**(1) (1992) 129–140
3. Weiermann, A.: Termination proofs for term rewriting systems with lexicographic path orderings imply multiply recursive derivation lengths. TCS **139** (1995) 355–362
4. Hofbauer, D.: Termination proofs by context-dependent interpretations. In: Proc. 12th RTA. Volume 2051 of LNCS. (2001) 108–121
5. Moser, G.: Derivational complexity of Knuth Bendix orders revisited. In: Proc. 13th LPAR. Volume 4246 of LNCS. (2006) 75–89
6. Geser, A., Hofbauer, D., Waldmann, J., Zantema, H.: On tree automata that certify termination of left-linear term rewriting systems. IC **205**(4) (2007) 512–534
7. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. TCS **236** (2000) 133–178
8. Giesl, J., Arts, T., Ohlebusch, E.: Modular termination proofs for rewriting using dependency pairs. JSC **34**(1) (2002) 21–58
9. Hirokawa, N., Middeldorp, A.: Tyrolean termination tool: Techniques and features. IC **205** (2007) 474–511
10. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
11. Terese: Term Rewriting Systems. Volume 55 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (2003)
12. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and improving dependency pairs. JAR **37**(3) (2006) 155–203
13. Bonfante, G., Cichon, A., Marion, J.Y., Touzet, H.: Algorithms with polynomial interpretation termination proof. JFP **11**(1) (2001) 33–53
14. Geser, A.: Relative Termination. PhD thesis, Universität Passau (1990)
15. Steinbach, J., Kühler, U.: Check your ordering – termination proofs and open problems. Technical Report SR-90-25, Universität Kaiserslautern (1990)
16. Contejean, E., Marché, C., Tomás, A.P., Urbain, X.: Mechanically proving termination using polynomial interpretations. JAR **34**(4) (2005) 325–363
17. Fuhs, C., Giesl, J., Middeldorp, A., Schneider-Kamp, P., Thiemann, R., Zankl, H.: SAT solving for termination analysis with polynomial interpretations. In: Proc. 10th SAT. Volume 4501 of LNCS. (2007) 340–354
18. Avanzini, M., Moser, G.: Complexity analysis by rewriting. In: Proc. 9th FLOPS. Volume 4989 of LNCS. (2008) 130–146