Complexity, Graphs, and 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. This paper builds on recent efforts (Hirokawa and Moser, 2008) to exploit the dependency pair method for verifying feasible, i.e., polynomial *runtime complexities* of term rewrite systems automatically. We extend our earlier results by revisiting dependency graphs in the context of complexity analysis. The obtained new results are easy to implement and considerably extend the analytic power of our existing methods. The gain in power is even more significant when compared to existing methods that directly, i.e., without the use of transformations, induce *feasible* runtime complexities. 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. *Runtime complexity* is a notion for capturing time complexities of functions defined by a term rewriting system (TRS for short) introduced in [1] (but see also [2,3,4]). In recent research we revisited the basic dependency pair method [5] in order to make it applicable for complexity analysis, cf. [1]. The dependency pair method introduced by Arts and Giesl [5] is one of the most powerful methods in termination analysis. The method enables us to use several powerful techniques including, usable rules, reduction pairs, argument filterings, and dependency graphs. Our main results in [1] show how natural improvements of the dependency pair method, like usable rules, reduction pairs, and argument filterings become applicable in the context of complexity analysis. In this paper, we will extend these recent results further.

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 that one considers *dependency pairs* that essentially encode recursive calls in a TRS. Notice that with respect to the TRS defined in Example 1 below, one finds 8 such pairs (see Section 4 for further details).

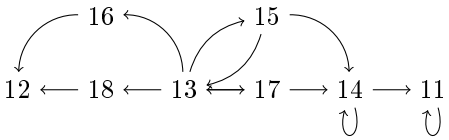
* This research is partly supported by FWF (Austrian Science Fund) project P20133, Grant-in-Aid for Young Scientists 20800022 of the Ministry of Education, Culture, Sports, Science and Technology of Japan, and STARC.

Example 1. Consider the TRS $\mathcal{R}$ which computes the greatest common divisor.[3]

$$
\begin{array}{rlrl}
1\colon & 0 \leqslant y \to \mathsf{true} & 6\colon & \mathsf{gcd}(0,y) \to y \\
2\colon & \mathsf{s}(x) \leqslant 0 \to \mathsf{false} & 7\colon & \mathsf{gcd}(\mathsf{s}(x),0) \to s(x) \\
3\colon & \mathsf{s}(x) \leqslant \mathsf{s}(y)) \to x \leqslant y & 8\colon & \mathsf{gcd}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{if}_{\mathsf{gcd}}(y \leqslant x, \mathsf{s}(x), \mathsf{s}(y)) \\
4\colon & x - 0 \to x & 9\colon & \mathsf{if}_{\mathsf{gcd}}(\mathsf{true},\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{gcd}(x-y,\mathsf{s}(y)) \\
5\colon & \mathsf{s}(x) - \mathsf{s}(y) \to x - y & 10\colon & \mathsf{if}_{\mathsf{gcd}}(\mathsf{false},\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{gcd}(y-x,\mathsf{s}(x))
\end{array}
$$

A very well-studied refinement of the dependency pair method are *dependency graphs*. To show termination of a TRS, it suffices to guarantee that none of the cycles in $\mathsf{DG}(\mathcal{R})$ [5] can give rise to an infinite rewrite sequence. (Here a *cycle* $\mathcal{C}$ is a nonempty set of dependency pairs of $\mathcal{R}$ such that for every two pairs $s \to t$ and $u \to v$ in $\mathcal{C}$ there exists a nonempty path in $\mathcal{C}$ from $s \to t$ to $u \to v$.) More precisely it suffices to prove for every cycle $\mathcal{C}$ in the dependency graph $\mathsf{DG}(\mathcal{R})$, that there are no $\mathcal{C}$-minimal rewrite sequences (see [7], but also [8,9]). To achieve this one may consider each cycle independently, i.e., for each cycle it suffices to find a reduction pair $(\gtrsim, \succ)$ (cf. Section 2) such that $\mathcal{R} \subseteq \gtrsim$, $\mathcal{C} \subseteq \gtrsim$ and $\mathcal{C} \cap \succ \neq \varnothing$, i.e., at least one dependency pair in $\mathcal{C}$ is strictly decreasing.

Example 2 (continued from Example 1). The *dependency graph* $\mathsf{DG}(\mathcal{R})$, whose nodes are the mentioned 8 dependency pairs, has the following form



This graph contains the maximal cycles $\{11\}$, $\{14\}$, and $\{13, 15, 17\}$, where the latter contains two sub-cycles. As already mentioned, it suffices to consider each of these five cycles individually.

The main contribution of this paper is to extend the dependency graph refinement of the dependency pair method to complexity analysis. This is a challenging task, and we face a couple of difficulties, documented via suitable examples below. To overcome these obstacles we adapt the standard notion of dependency graph suitably and introduce *weak (innermost) dependency graphs*, based on *weak dependency pairs*, which have been studied in [1]. Moreover, we observe that in the context of complexity analysis, it is not enough to focus on the (maximal) cycles of a (weak) dependency graph. Instead, we show how cycle detection is to be replaced by *path detection*, in order to salvage the (standard) technique of dependency graphs for runtime complexity considerations.

The remainder of the paper is organised as follows. After recalling basic notions in Section 2. We recall in Section 3 main results from [1] that will be extended in the sequel. In Section 4 we establish our dependency graph analysis for complexity analysis. Finally, we conclude in Section 5, where we assess the applicability of our method.

[3] This is Example 3.6a in Arts and Giesl's collection of TRSs [6].

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 $\mathcal{T}(\mathcal{F},\mathcal{V})$ ($\mathcal{T}$ for short). The *root symbol* of a term t is either t itself, if $t \in \mathcal{V}$, or the symbol f, if $t = f(t_1,\dots,t_n)$. The *set of positions* $\mathcal{P}\mathsf{os}(t)$ 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 *descendants* of a position with respect to a rewrite sequence are defined as usual, cf. [11]. The subterm relation is denoted as $\trianglelefteq$. $\mathcal{V}\mathsf{ar}(t)$ ($\mathcal{F}\mathsf{un}(t)$) denotes the set of variables (functions) occurring in a term t. The *size* $|t|$ of a term is defined as the number of symbols in t. A *term rewrite system* $\mathcal{R}$ 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 $\to_{\mathcal{R}}$, 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 $s \to t$. The *innermost rewrite relation* $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ of a TRS $\mathcal{R}$ is defined on terms as follows: $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$ if there exists 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 symbols is denoted as $\mathcal{D}$, while the constructor symbols are collected in $\mathcal{C}$. We call a term $t = f(t_1,\dots,t_n)$ *basic* if $f \in \mathcal{D}$ and $t_i \in \mathcal{T}(\mathcal{C},\mathcal{V})$ for all $1 \leqslant i \leqslant n$.

We call a TRS *terminating* if no infinite rewrite sequence exists. The n-fold composition of $\to$ is denoted as $\to^n$ and 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.[4] 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 $\mathsf{rc}^{\mathsf{i}}_{\mathcal{R}}(n) := \mathsf{rc}(n,\mathcal{T}_{\mathsf{b}},\xrightarrow{\mathsf{i}}_{\mathcal{R}})$. Notice that 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 $\mathcal{T}(\mathcal{F},\mathcal{V})$, compare [12]. We sometimes say the (innermost) runtime complexity of $\mathcal{R}$ is *linear*, *quadratic*, or *polynomial* if $\mathrm{rc}^{(\mathsf{i})}_{\mathcal{R}}$ is bounded by a linear, quadratic, or polynomial function in n, respectively.

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 $t_1 \succ t_2 \succ t_3 \cdots$. An $\mathcal{F}$*-algebra* $\mathcal{A}$ consists of a

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

carrier set A and an interpretation $f_{\mathcal{A}}$ 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 proper 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, and $[\alpha]_{\mathcal{A}}(\cdot)$ denotes the usual evaluation function associated with $\mathcal{A}$. 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 $\alpha\colon \mathcal{V} \to A$. For the reflexive closure $\geqslant$ of $>$, the preorder $\geqslant_{\mathcal{A}}$ is similarly defined. Clearly the proper order $>_{\mathcal{A}}$ is a reduction order, i.e., if $\mathcal{R} \subseteq >_{\mathcal{A}}$, for a TRS $\mathcal{R}$, then we can conclude termination of $\mathcal{R}$. 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 $\gtrsim \cdot \succ \cdot \gtrsim \subseteq \succ$. Note that for any WMA $\mathcal{A}$ the pair $(\geqslant_{\mathcal{A}}, >_{\mathcal{A}})$ constitutes a reduction pair.

We call a WMA $\mathcal{A}$ based on the natural numbers $\mathbb{N}$ a *polynomial interpretation*, if all functions $f_{\mathcal{A}}$ are polynomials. 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 linear polynomials. 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 (see [13]). 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 [1] but also [3]). Finally, we introduce a very restrictive class of polynomial interpretations: *strongly linear interpretations* (*SLI* for short). A polynomial interpretation is called *strongly linear* if all functions $f_{\mathcal{A}}$ are interpreted as strongly linear polynomials.

3 Complexity Analysis Based on the Dependency Pair Method

In this section, we recall central definitions and results established in [1]. We kindly refer the reader to [1] for additional examples and underlying intuitions.

We write $C\langle t_1, \ldots, t_n\rangle_X$ to denote $C[t_1, \ldots, t_n]$, whenever $\mathrm{root}(t_i) \in X$ for all $1 \leqslant i \leqslant n$ and C is an n-hole context containing no X-symbols. Let t be a term. We set $t^\sharp := t$ if $t \in \mathcal{V}$, and $t^\sharp := f^\sharp(t_1, \ldots, t_n)$ if $t = f(t_1, \ldots, t_n)$. Here $f^\sharp$ is a new n-ary function symbol called *dependency pair symbol*. For a signature $\mathcal{F}$, we define $\mathcal{F}^\sharp = \mathcal{F} \cup \{f^\sharp \mid f \in \mathcal{F}\}$.

Definition 3. *Let $\mathcal{R}$ be a TRS. If $l \to r \in \mathcal{R}$ and $r = C\langle u_1, \ldots, u_n\rangle_{\mathcal{D}\cup\mathcal{V}}$ then the rewrite rule $l^\sharp \to \mathrm{COM}(u_1^\sharp, \ldots, u_n^\sharp)$ is called a* weak dependency pair *of $\mathcal{R}$. Here* COM *is defined with a fresh n-ary function symbol c (corresponding to $l \to r$) as follows: $\mathrm{COM}(t_1, \ldots, t_n)$ is t_1 if $n = 1$, and $c(t_1, \ldots, t_n)$ otherwise. The symbol c is called* compound symbol. *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 ten weak dependency pairs.

11:	$0 \leqslant^\sharp y \to \mathsf{c}_1$	16:	$\mathsf{gcd}^\sharp(0, y) \to y$
12:	$\mathsf{s}(x) \leqslant^\sharp 0 \to \mathsf{c}_2$	17:	$\mathsf{gcd}^\sharp(\mathsf{s}(x), 0) \to x$
13:	$\mathsf{s}(x) \leqslant^\sharp \mathsf{s}(y) \to x \leqslant^\sharp y$	18:	$\mathsf{gcd}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{if}_{\mathsf{gcd}}{}^\sharp(y \leqslant x, \mathsf{s}(x), \mathsf{s}(y))$
14:	$\mathsf{s}(x) -^\sharp 0 \to x$	19:	$\mathsf{if}_{\mathsf{gcd}}{}^\sharp(\mathsf{true}, \mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{gcd}^\sharp(x - y, \mathsf{s}(y))$
15:	$\mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y$	20:	$\mathsf{if}_{\mathsf{gcd}}{}^\sharp(\mathsf{false}, \mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{gcd}^\sharp(y - x, \mathsf{s}(x))$

Definition 5. *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 $\mathsf{WIDP}(\mathcal{R})$.*

Definitions 3 and 5 should be compared to the definition of "standard" dependency pairs.

Definition 6 ([5]). *The set $\mathsf{DP}(\mathcal{R})$ of (standard)* dependency pairs *of a TRS $\mathcal{R}$ is defined as $\{l^\sharp \to u^\sharp \mid l \to r \in \mathcal{R}, u \trianglelefteq r, \mathrm{root}(u) \in \mathcal{D}\}$.*

Example 7 (continued from Example 1). As already mentioned in the Introduction, the TRS $\mathcal{R}$ admits 8 (standard) dependency pairs. Notice that the sets $\mathsf{DP}(\mathcal{R})$ and $\mathsf{WDP}(\mathcal{R})$ are incomparable. For example $0 \leqslant^\sharp y \to \mathsf{c}_1 \in \mathsf{WDP}(\mathcal{R}) \setminus \mathsf{DP}(\mathcal{R})$, while $\mathsf{gcd}^\sharp(\mathsf{s}(x), \mathsf{s}(y)) \to y \leqslant^\sharp x \in \mathsf{DP}(\mathcal{R}) \setminus \mathsf{WDP}(\mathcal{R})$.

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

Proposition 8 ([1]). *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 $\overset{i}{\to}_{\mathcal{R}}$ depending on whether $\mathcal{P} = \mathsf{WDP}(\mathcal{R})$ or $\mathcal{P} = \mathsf{WIDP}(\mathcal{R})$.*

We recall the notion of *relative rewriting* [11]. 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 $\mathcal{S}$. (Note that $\to_{\mathcal{R}/\mathcal{S}} = \to_{\mathcal{R}}$, if $\mathcal{S} = \varnothing$.) Let $\mathcal{A}$ denote a strongly linear interpretation.

Proposition 9 ([1]). *Let $\mathcal{R}$ and $\mathcal{S}$ be TRSs, and $\mathcal{A}$ an SLI compatible with $\mathcal{S}$. There exist constants K and L, depending only on $\mathcal{R}$ and $\mathcal{A}$, such that $\mathsf{dl}(t, \to_{\mathcal{R}\cup\mathcal{S}}) \leqslant K \cdot \mathsf{dl}(t, \to_{\mathcal{R}/\mathcal{S}}) + L \cdot |t|$ for all terminating terms t on $\mathcal{R} \cup \mathcal{S}$.*

We need some further definitions. Let $\mathcal{R}$ be a TRS, let $\mathcal{P}$ a set of weak or weak innermost 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 $\mathcal{R}$.[5]

We write $\mathcal{T}_{\mathsf{b}}^{\sharp}$ for $\{t^{\sharp} \mid t \in \mathcal{T}_{\mathsf{b}}\}$. The set $\mathcal{T}_{\mathsf{c}}^{\sharp}$ is inductively defined as follows (i) $\mathcal{T}^{\sharp} \cup \mathcal{T} \subseteq \mathcal{T}_{\mathsf{c}}^{\sharp}$, where $\mathcal{T}^{\sharp} = \{t^{\sharp} \mid t \in \mathcal{T}\}$. And (ii) $c(t_1,\dots,t_n) \in \mathcal{T}_{\mathsf{c}}^{\sharp}$, whenever $t_1,\dots,t_n \in \mathcal{T}_{\mathsf{c}}^{\sharp}$ and c a compound symbol. A proper order $\succ$ on $\mathcal{T}_{\mathsf{c}}^{\sharp}$ is called *safe* if $c(s_1,\dots,s_i,\dots,s_n) \succ c(s_1,\dots,t,\dots,s_n)$ for all n-ary compound symbols c and all terms $s_1,\dots,s_n,t$ with $s_i \succ t$. A reduction pair $(\gtrsim,\succ)$ is called *collapsible* for a TRS $\mathcal{R}$ if $\succ$ is collapsible for $\mathcal{R}$. It is called *safe* if the well-founded order $\succ$ is safe. 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 $\succ$. It is easy to see that if $(\mathcal{A},>)$ is a safe algebra then $(\geqslant_{\mathcal{A}},>_{\mathcal{A}})$ is a safe reduction pair.

Proposition 10 ([1]). *Let $\mathcal{R}$ be a TRS, let $\mathcal{A}$ an SLI, let $\mathcal{P}$ be the set of weak or weak innermost 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 $t \in \mathcal{T}_{\mathsf{b}}$, there exist constants K and L (depending only on $\mathcal{R}$ and $\mathcal{A}$) such that $\mathsf{dl}(t,\to) \leqslant K \cdot \mathsf{G}(t^{\sharp},\succ) + L \cdot |t|$. Here $\to$ denotes $\to_{\mathcal{R}}$ or $\xrightarrow{i}_{\mathcal{R}}$ depending on whether $\mathcal{P} = \mathsf{WDP}(\mathcal{R})$ or $\mathcal{P} = \mathsf{WIDP}(\mathcal{R})$.*

Suppose the assertions of the proposition are met and there exists a polynomial p such that $\mathsf{G}(t^{\sharp},\succ) \leqslant p(|t|)$ holds. Then, as an easy corollary to Proposition 10, we observe that the runtime complexity induced by $\mathcal{R}$ is majorised by p.

4 Dependency Graphs

In this section, we study a natural refinement of the dependency pair method, namely *dependency graphs* (see [5,7,14,8]) in the context of complexity analysis. We start with a brief motivation. Let $\mathcal{R}$ be a TRS, let $\mathcal{P}$ denote a set of weak or weak innermost dependency pairs and let $(s_i)_{i=0,\dots,n}$ denote a maximal derivation D with respect to $\mathcal{R}$ with $s_0 \in \mathcal{T}_{\mathsf{b}}$. In order to estimate the length ℓ of this derivation it suffices to estimate the length of the derivation $t_0 \to_{\mathcal{U}(\mathcal{P})\cup\mathcal{P}}^{*} t_n$, where $t_0 = s_0^{\sharp} \in \mathcal{T}_{\mathsf{b}}^{\sharp}$, cf. Proposition 8. If we suppose compatibility of $\mathcal{U}(\mathcal{P})$ with a strongly linear interpretation $\mathcal{A}$, we may estimate the derivation length ℓ by finding *one* (safe and collapsible) reduction pair $(\gtrsim,\succ)$ such that $\mathcal{U}(\mathcal{P}) \subseteq \gtrsim$ and $\mathcal{P} \subseteq \succ$ holds, cf. Proposition 10. On the other hand in termination analysis—as already mentioned in the Introduction—it suffices to guarantee that for any cycle $\mathcal{C}$ in the dependency graph $\mathsf{DG}(\mathcal{R})$, there are no $\mathcal{C}$-minimal rewrite sequences, cf. [7]. Hence, we strive to extend this idea to complexity analysis.

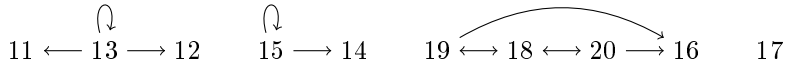
4.1 From Cycle Analysis to Path Detection

Let us recall the definition of a dependency graph and extend it suitably to weak and weak innermost dependency pairs.

[5] Note that most reduction orders are collapsible. E.g. if $\mathcal{A}$ is a polynomial interpretation then $>_{\mathcal{A}}$ is collapsible, as one may take any α and set $\mathsf{G}(t,>_{\mathcal{A}}) := [\alpha]_{\mathcal{A}}(t)$.

Definition 11. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$ and let $\mathcal{P}$ be the set of weak, weak innermost, or (standard) dependency pairs. The nodes of the* weak dependency graph $\mathsf{WDG}(\mathcal{R})$*,* weak innermost dependency graph $\mathsf{WIDG}(\mathcal{R})$*, or* dependency graph $\mathsf{DG}(\mathcal{R})$ *are the elements of $\mathcal{P}$ and there is an arrow from $s \to t$ to $u \to v$ if and only if there exist a context C and substitutions $\sigma, \tau \colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $t\sigma \to^* C[u\tau]$, where $\to$ denotes $\to_{\mathcal{R}}$ or $\xrightarrow{i}_{\mathcal{R}}$ depending on whether $\mathcal{P} = \mathsf{WDP}(\mathcal{R})$, $\mathcal{P} = \mathsf{DP}(\mathcal{R})$ or $\mathcal{P} = \mathsf{WIDP}(\mathcal{R})$, respectively.*

Example 12 (continued from Example 4). The weak dependency graph $\mathsf{WDG}(\mathcal{R})$ has the following form.



We recall a theorem on the dependency graph refinement in conjunction with usable rules and innermost rewriting (see [7], but also [15]). Similar results hold in the context of full rewriting, see [9,8].

Theorem 13 ([7]). *A TRS $\mathcal{R}$ is innermost terminating if for every maximal cycle $\mathcal{C}$ in the dependency graph $\mathsf{DG}(\mathcal{R})$ there exists a reduction pair $(\gtrsim, \succ)$ such that $\mathcal{U}(\mathcal{C}) \subseteq \gtrsim$ and $\mathcal{C} \subseteq \succ$.*

The following example shows that we cannot directly employ Theorem 13 in the realm of complexity analysis. Even though in this setting we can restrict our attention to a specific strategy: innermost rewriting.

Example 14. Consider the TRS $\mathcal{R}_{\mathsf{exp}}$

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

$\mathsf{DP}(\mathcal{R}_{\mathsf{exp}})$ consists of three pairs: $1\colon \mathsf{exp}^\sharp(\mathsf{r}(x)) \to \mathsf{d}^\sharp(\mathsf{exp}(x))$, $2\colon \mathsf{exp}^\sharp(\mathsf{r}(x)) \to \mathsf{exp}^\sharp(x)$, and $3\colon \mathsf{d}^\sharp(\mathsf{s}(x)) \to \mathsf{d}^\sharp(x)$. Hence the dependency graph $\mathsf{DG}(\mathcal{R}_{\mathsf{exp}})$ contains two maximal cycles: $\{2\}$ and $\{3\}$. It is easy to see how to define two reduction pairs $(\geqslant_{\mathcal{A}}, >_{\mathcal{A}})$ and $(\geqslant_{\mathcal{B}}, >_{\mathcal{B}})$ such that the conditions of the theorem are fulfilled. For that it suffices to define interpretations $\mathcal{A}$ and $\mathcal{B}$, respectively. Because one can find suitable linear restricted ones for $\mathcal{A}$ and $\mathcal{B}$, compatibility with these interpretations apparently induces *linear* runtime complexity of $\mathcal{R}_{\mathsf{exp}}$, cf. [3,1] (even for full rewriting). However, we must not conclude linear innermost runtime complexity for $\mathcal{R}_{\mathsf{exp}}$ in this setting, as $\mathcal{R}_{\mathsf{exp}}$ formalises the exponentiation function and setting $t_n = \mathsf{exp}(\mathsf{r}^n(\mathsf{0}))$ we obtain $\mathsf{dl}(t_n, \xrightarrow{i}_{\mathcal{R}}) \geqslant 2^n$ for each $n \geqslant 0$. Thus the innermost runtime complexity of $\mathcal{R}_{\mathsf{exp}}$ is exponential.

Note that the problem exemplified by Example 14 cannot be circumvented by replacing the dependency graph employed in Theorem 13 with weak (innermost) dependency graphs. Furthermore, observe that while Proposition 8 allows us to replace in Example 14 the innermost rewrite relation $\xrightarrow{i}_{\mathcal{R}}$ by the (sometimes simpler) rewrite relation $\xrightarrow{i}_{\mathcal{U}(\mathsf{DP}(\mathcal{R}))\cup\mathsf{DP}(\mathcal{R})}$, this is of no help: The exponential

length of $t_n^\sharp$ in Example 14 with respect to $\mathcal{U}(\mathsf{DP}(\mathcal{R})) \cup \mathsf{DP}(\mathcal{R})$ is *not* due to the cycles $\{2\}$ or $\{3\}$, but achieved through the non-cyclic pair 1 and its usable rules. These observations are cast into Definition 15, below.

A graph is called *strongly connected* if any node is connected with every other node by a path. A *strongly connected component* (*SCC* for short) is a maximal strongly connected subgraph.[6]

Definition 15. *Let $\mathcal{G}$ be a graph, let $\equiv$ denote the equivalence relation induced by SCCs, and let $\mathcal{P}$ be a SCC in $\mathcal{G}$. The set of all source nodes in $\mathcal{G}/_\equiv$ is denoted by* Src. *Let $l \to r$ be a dependency pair in G, let $\mathcal{K} \in \mathcal{G}/_\equiv$ and let $\mathcal{C}$ denote the SCC represented by $\mathcal{K}$. Then we write $l \to r \in \mathcal{K}$ if $l \to r \in \mathcal{C}$.*

Example 16 (Continued from Example 12). There are 8 SCCs in $\mathsf{WDG}(\mathcal{R})$, almost all except $\{18, 19, 20\}$ being trivial. Hence the graph $\mathsf{WDG}(\mathcal{R})/_\equiv$ has the following form and $\mathsf{Src} = \{\{13\}, \{15\}, \{17\}, \{18, 19, 20\}\}$.

$$11 \longleftarrow 13 \longrightarrow 12 \qquad 15 \longrightarrow 14 \qquad \{18{,}19{,}20\} \longrightarrow 16 \qquad 17$$

4.2 Refinement Based on Path Detection

We re-consider the motivating derivation D:

$$t_0 \to_{\mathcal{U}(\mathcal{P})\cup\mathcal{P}} t_1 \to_{\mathcal{U}(\mathcal{P})\cup\mathcal{P}} \cdots \to_{\mathcal{U}(\mathcal{P})\cup\mathcal{P}} t_n \,, \tag{1}$$

where $t_0 \in \mathcal{T}_\mathsf{b}^\sharp$. To simplify the exposition, we set $\mathcal{P} = \mathsf{WDP}(\mathcal{R})$ and $\mathcal{G} = \mathsf{WDG}(\mathcal{R})$. Momentarily we assume that all compound symbol are of arity 0, as is for instance the case in Example 4. Above we asserted that there exists an SLI $\mathcal{A}$ such that $\mathcal{U}(\mathcal{P}) \subseteq >_\mathcal{A}$. Hence Proposition 9 is applicable. Thus, to estimate the length of the derivation (1) it suffices to consider the following relative rewriting derivation:

$$t_0 \to_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t_1 \to_{\mathcal{P}/\mathcal{U}(\mathcal{P})} \cdots \to_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t_n \,. \tag{2}$$

Exploiting the given assumptions, it is not difficult to see that derivation (2) is representable as follows:

$$t_0 \to^{\ell_1}_{\mathcal{P}_1/\mathcal{U}(\mathcal{P}_1)} t_{\ell_1} \to^{\ell_2}_{\mathcal{P}_2/\mathcal{U}(\mathcal{P}_1)\cup\mathcal{U}(\mathcal{P}_2)} \cdots \to^{\ell_m}_{\mathcal{P}_m/\mathcal{U}(\mathcal{P}_1)\cup\cdots\cup\mathcal{U}(\mathcal{P}_m)} t_n \tag{3}$$

where, $(\mathcal{P}_1, \ldots, \mathcal{P}_m)$ is a *path* in $\mathcal{G}/_\equiv$ with $\mathcal{P}_1 \in \mathsf{Src}$. Since the length ℓ of the pictured $\to_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$-rewrite sequence equals $\ell_1 + \cdots + \ell_m$, this suggests that we can estimate each ℓ_j ($j \in \{1, \ldots, m\}$) independently. We assume the existence of a family of SLIs $\mathcal{B}_j$ ($j \in \{1, \ldots, m\}$) such that $\mathcal{U}(\mathcal{P}_1) \cup \cdots \cup \mathcal{U}(\mathcal{P}_j) \subseteq \geqslant_{\mathcal{B}_j}$ and $\mathcal{P}_j \subseteq >_{\mathcal{B}_j}$ holds for every i. From this we can conclude $\ell_j = \mathsf{O}(|t_{\ell_j}|)$ for all $j \in \{1, \ldots, m\}$. The next step is to estimate each ℓ_j by a function (preferable a polynomial) in $|t_0|$. As each of the WMAs $\mathcal{B}_j$ is assumed to be strongly linear, we can even conclude $[\alpha_0]_{\mathcal{B}_j}(t_{\ell_j}) = \Omega(|t_{\ell_j}|)$.(Here α_0 denotes the assignment

[6] Note that in the literature SCCs are sometimes defined as *maximal cycles*. This alternative definition is of limited use in our context.

mapping any variable to 0.) In sum, we obtain for each $j \in \{1, \dots, m\}$, the existence of a constant c_j such that $|t_{\ell_j}| \leqslant c_j \cdot |t_0|$ and thus there exists a linear polynomial $p(x)$ such that $\ell_j \leqslant p(|t_0|)$. However, some care is necessary in assessing this observation: Note that the given argument cannot be used to deduce *polynomial* runtime complexity, if we weaken the assumption that the algebras $\mathcal{B}_j$ are strongly linear only slightly. Hence, we replace the direct application of Proposition 9 as follows.

Lemma 17. *$n \leqslant \mathsf{dl}(s, \to_{\mathcal{R}_2/\mathcal{S}_1 \cup \mathcal{S}_2})$ whenever $s \to_{\mathcal{S}_1}{}^* \cdot \to_{\mathcal{R}_2/\mathcal{S}_2}{}^n u$.*

Proof. Straightforward. □

We lift the assumption that all compound symbols are of arity at most 0. Perhaps surprisingly this generalisation complicates the matter considerably. First a *maximal* derivation need no longer be of the form given in (3) which is exemplified by Example 18 below.

Example 18. Consider the TRS $\mathcal{R} = \{\mathsf{f}(\mathsf{0}) \to \mathsf{leaf}, \mathsf{f}(\mathsf{s}(x)) \to \mathsf{branch}(\mathsf{f}(x), \mathsf{f}(x))\}$ The set $\mathsf{WDP}(\mathcal{R})$ consists of the two weak dependency pairs: $1\colon \mathsf{f}^\sharp(\mathsf{0}) \to \mathsf{c}_1$ and $2\colon \mathsf{f}^\sharp(\mathsf{s}(x)) \to \mathsf{c}_2(\mathsf{f}^\sharp(x), \mathsf{f}^\sharp(x))$. Hence the weak dependency graph $\mathsf{WDG}(\mathcal{R})$ contains 2 SCCs: $\{2\}$ and $\{1\}$. Clearly $\mathsf{Src} = \{\{2\}\}$. Let $t_n = \mathsf{f}^\sharp(\mathsf{s}^n(\mathsf{0}))$. Consider the following sequence:

$$\begin{aligned} t_2 &\to_{\{2\}}{}^2 \mathsf{c}_2(\mathsf{c}_2(t_0, t_0), t_1) \to_{\{1\}} \mathsf{c}_2(\mathsf{c}_2(\mathsf{c}_1, t_0), t_1) \\ &\to_{\{2\}} \mathsf{c}_2(\mathsf{c}_2(\mathsf{c}_1, t_0), \mathsf{c}_2(t_0, t_0)) \to_{\{1\}}{}^3 \mathsf{c}_2(\mathsf{c}_2(\mathsf{c}_1, \mathsf{c}_1), \mathsf{c}_2(\mathsf{c}_1, \mathsf{c}_1)) \,. \end{aligned}$$

This derivation does not have the form (3), because it is based on the sequence $(\{2\}, \{1\}, \{2\}, \{1\})$, which is not a path in $\mathsf{WDG}(\mathcal{R})/_\equiv$.

Notice that the derivation in Example 18 can be reordered (without affecting its length) such that the derivation becomes based on a path. Still, not every derivation can be abstracted to a path. Consider a maximal (with respect to subset inclusion) component of $\mathsf{WDG}(\mathcal{R})/_\equiv$. Clearly this component forms a directed acyclic graph $\mathcal{G}$, and without loss of generality we can conceive $\mathcal{G}$ as a tree T with root in Src. Suppose further that T is not degenerated to a branch. Then a given derivation may only be abstractable by different paths in T, as exemplified by Example 19.

Example 19. Consider the TRS $\mathcal{R} = \{\mathsf{f} \to \mathsf{c}(\mathsf{g}, \mathsf{h}), \mathsf{g} \to \mathsf{a}, \mathsf{h} \to \mathsf{a}\}$. Thus $\mathsf{WDP}(\mathcal{R})$ consists of three dependency pairs: $1\colon \mathsf{f}^\sharp \to \mathsf{c}_1(\mathsf{g}^\sharp, \mathsf{h}^\sharp)$, $2\colon \mathsf{g}^\sharp \to \mathsf{c}_2$, and $3\colon \mathsf{h}^\sharp \to \mathsf{c}_3$. Let $\mathcal{P} := \mathsf{WDP}(\mathcal{R})$, then Then clearly $\mathcal{P} = \mathsf{WDG}(\mathcal{R}) = \mathsf{WDG}(\mathcal{R})/_\equiv$. Consider the following derivation

$$\mathsf{f}^\sharp \to_{\mathcal{P}} \mathsf{c}_1(\mathsf{g}^\sharp, \mathsf{h}^\sharp) \to_{\mathcal{P}} \mathsf{c}_1(\mathsf{c}_2, \mathsf{h}^\sharp) \to_{\mathcal{P}} \mathsf{c}_1(\mathsf{c}_2, \mathsf{c}_3) \,.$$

This derivation is composed from the paths $(\{1\}, \{2\})$ and $(\{1\}, \{3\})$.

Fortunately, we can circumvent these obstacles. Let $\mathcal{P}$ denote the set of weak or weak innermost dependency pairs of a TRS $\mathcal{R}$. We make the following easy observation.

Lemma 20. *Let $\mathcal{G}$ denote a weak or weak innermost dependency graph. Let $\mathcal{C} \subseteq \mathcal{G}$ and let $D\colon s \to^*_{\mathcal{C}/\mathcal{U}(\mathcal{P})} t$ denote a derivation based on $\mathcal{C}$ with $s \in \mathcal{T}^\sharp_\mathsf{c}$. Then D has the following form: $s = s_0 \to_{\mathcal{C}/\mathcal{U}(\mathcal{P})} s_1 \to_{\mathcal{C}/\mathcal{U}(\mathcal{P})} \cdots \to_{\mathcal{C}/\mathcal{U}(\mathcal{P})} s_n = t$ where each $s_i \in \mathcal{T}^\sharp_\mathsf{c}$.*

Proof. It is easy to see that D has the presented form and that for each $i \in \{0, \ldots, n\}$ there exists a context C such that $s_i = C[u^\sharp_1, \ldots, u^\sharp_r]$ and C consists of compound symbols only. This establishes the lemma. □

Motivated by Example 18 we observe that a weak (innermost) dependency pair containing an m-ary ($m > 1$) compound symbol can only induces m *independent* derivations. Hence, we can reorder derivations to achieve the structure of derivation (3). This is formally proven via the next two lemmas.

Lemma 21. *Let $\mathcal{G}$ denote a weak or weak innermost dependency graph and let $\mathcal{K}$ and $\mathcal{L}$ denote two different nodes in $\mathcal{G}/_\equiv$ such that there is no edge from $\mathcal{K}$ to $\mathcal{L}$. Let $s_0 \in \mathcal{T}^\sharp_\mathsf{c}$ and suppose the existence of a derivation D of the following form: $s_0 \to^n_{\mathcal{K}/\mathcal{U}(\mathcal{P})} s_n \to^*_{\mathcal{U}(\mathcal{P})} t_0 \to^m_{\mathcal{L}/\mathcal{U}(\mathcal{P})} t_m$. Then there exists a derivation D' which has the form $t'_0 \to^m_{\mathcal{L}/\mathcal{U}(\mathcal{P})} t'_m \to^*_{\mathcal{U}(\mathcal{P})} s'_0 \to^n_{\mathcal{K}/\mathcal{U}(\mathcal{P})} s'_n$ with $t'_0 \in \mathcal{T}^\sharp_\mathsf{c}$.*

Proof. Consider the following two dependency pairs: $1\colon u^\sharp_k \to \mathrm{COM}(v^\sharp_{k1}, \ldots, v^\sharp_{kr})$ and $2\colon u^\sharp_l \to \mathrm{COM}(v^\sharp_{l1}, \ldots, u^\sharp_{lr})$. Here the dependency pair 1 belongs to $\mathcal{K}$ and denotes the last dependency pair employed in D before the path leaves $\mathcal{K}$ into $\mathcal{L}$, while 2 denotes the first pair in $\mathcal{L}$. The assumption that there is no edge connecting $\mathcal{K}$ and $\mathcal{L}$ can be reformulated as follows:

(†) No context C and no substitutions $\sigma, \tau\colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ exist such that $\mathrm{COM}(v^\sharp_{k1}\sigma, \ldots, v^\sharp_{kr}\sigma) \to^*_{\mathcal{U}(\mathcal{P})} C[u^\sharp_l\tau]$ holds.

To prove the lemma, we proceed by induction on n. It suffices to consider the step case $n > 1$. By assumption the last rewrite step in the subderivation $D_0\colon s_0 \to^n_{\mathcal{K}/\mathcal{U}(\mathcal{P})} s_n$ employs dependency pair 1. Let $p \in \mathcal{P}\mathsf{os}(s_n)$ denote the position of the reduct $\mathrm{COM}(v^\sharp_{k1}\tau, \ldots, v^\sharp_{kr}\tau)$ in s_n. By assumption there exists a derivation $s_n \to^*_{\mathcal{U}(\mathcal{P})} t_0$. Let $q \in \mathcal{P}\mathsf{os}(s_n)$ denote the position of the redex in s_n that is contracted as first step in this reduction. Without loss of generality we can assume that both positions are parallel to each other. Otherwise one of the following cases applies. Either $p < q$ or $p \geqslant q$. But clearly the first case contradicts the assumption (†). Hence, assume the second. But this is also impossible. Lemma 20 yields that $s_n|_q \in \mathcal{T}^\sharp_\mathsf{c}$, which contradicts that q is redex with respect to $\mathcal{U}(\mathcal{P})$. Repeating this argument we see that position p has exactly one descendant in t_0. A similar argument shows that all redex positions in the subderivation $D_1\colon t_0 \to^m_{\mathcal{L}/\mathcal{U}(\mathcal{P})} t_m$ are parallel to (descendants of) p. Hence, we can move the last rewrite step $s_{n-1} \to_\mathcal{K} s_n$ in the derivation D_0 after the derivation D_1. Note that in each of the terms $(t_i)_{i=1,\ldots,m}$ the position p exists and denotes the term $\mathrm{COM}(v^\sharp_{k1}\tau, \ldots, v^\sharp_{kr}\tau)$. Hence, the replacement of $\mathrm{COM}(v^\sharp_{k1}\tau, \ldots, v^\sharp_{kr}\tau)$ everywhere by $u^\sharp_k\sigma$ does not affect the validity of the rewrite sequence. Furthermore the set $\mathcal{T}^\sharp_\mathsf{c}$ is closed under this operation. Now, induction hypothesis becomes applicable to derive the existence of the sought derivation D'. □

Let $\mathcal{G}$ denote a weak or weak innermost dependency graph and let $D\colon s \to^{\ell} t$ denote a derivation, such that $s \in \mathcal{T}^{\sharp}_{\mathsf{b}}$. Here $\to$ denotes either $\to_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$ or $\xrightarrow{\mathrm{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$. We say that D is *based on* $(\mathcal{P}_1, \dots, \mathcal{P}_m)$ in $\mathcal{G}/_{\equiv}$ if D is of form

$$s \xrightarrow{(\mathrm{i})}{}^{\ell_1}_{\mathcal{P}_1/\mathcal{U}(\mathcal{P})} \cdots \xrightarrow{(\mathrm{i})}{}^{\ell_m}_{\mathcal{P}_m/\mathcal{U}(\mathcal{P})} t \;,$$

with $\ell_1, \dots, \ell_m \geqslant 0$. We arrive at the main lemma of this section.

Lemma 22. *Let $\mathcal{P}$ denote a set of weak or weak innermost dependency pairs, let $s \in \mathcal{T}^{\sharp}_{\mathsf{b}}$ and let $D\colon s \to^{\ell} t$ denote a maximal derivation, where $\to$ denotes $\to_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$ or $\xrightarrow{\mathrm{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$ respectively. Suppose that D is based on $(\mathcal{P}_1, \dots, \mathcal{P}_m)$ and $\mathcal{P}_1 \in \mathsf{Src}$. Then there exists a derivation $D'\colon s \to^{\ell} t$ based on $(\mathcal{P}'_1, \dots, \mathcal{P}'_{m'})$, with $\mathcal{P}'_1 \in \mathsf{Src}$ such that all $\mathcal{P}'_i$ $(i \in \{1, \dots, m'\})$ are pairwise distinct.*

Proof. Without loss of generality, we restrict our attention to weak dependency pairs. To prove the lemma, we consider a sequence $(\mathcal{P}_1, \dots, \mathcal{P}_m)$, where there exist indices i, j and k with $i < j < k$ and $\mathcal{P}_i = \mathcal{P}_k$. By induction on $j - i$ we show that this path is transformable into a sequence $(\mathcal{P}'_1, \dots, \mathcal{P}'_{m'})$ of the required form. It suffices to prove the step case. Moreover, we can assume without loss of generality that $k = j + 1$. Consider the two dependency pairs: $1\colon l^{\sharp}_j \to \mathrm{COM}(u^{\sharp}_{j1}, \dots, u^{\sharp}_{jr})$ and $2\colon l^{\sharp}_k \to \mathrm{COM}(u^{\sharp}_{k1}, \dots, u^{\sharp}_{kr})$. Dependency pair 1 belongs to $\mathcal{P}_j$ and denotes the last dependency pair employed in D before the sequence leaves $\mathcal{P}_j$ into $\mathcal{P}_k$, while 2 denotes the first pair in $\mathcal{P}_k$. We consider two cases:

1. Assume there exist a context C and substitutions $\sigma, \tau\colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that the following holds: $\mathrm{COM}(u^{\sharp}_{j1}\sigma, \dots, u^{\sharp}_{jr}\sigma) \to^{*} C[l^{\sharp}_k\tau]$. Thus by definition of weak dependency graphs the node in $\mathsf{WDG}(\mathcal{R})$ representing dependency pair 1 is connected to the node representing dependency pair 2. In particular every node in the SCCs represented by $\mathcal{P}_i = \mathcal{P}_k$ is connected to every node in the SCC represented by $\mathcal{P}_j$. This implies that $\mathcal{P}_i = \mathcal{P}_j = \mathcal{P}_k$ contradicting the assumption.
2. Otherwise, there is no edge between $\mathcal{P}_j$ and $\mathcal{P}_k$ in the graph $\mathcal{G}/_{\equiv}$ and by the assumptions on $(\mathcal{P}_1, \dots, \mathcal{P}_\ell)$ we find a derivation of the following form: $D_0\colon s_{j_1} \to^{p}_{\mathcal{P}_j/\mathcal{U}(\mathcal{P})} s_{j_p} \to^{*}_{\mathcal{U}(\mathcal{P})} s_{k_1} \to^{q}_{\mathcal{P}_k/\mathcal{U}(\mathcal{P})} s_{k_q}$. Due to Lemma 21 there exists a derivation $D_1\colon s'_{k_1} \to^{q}_{\mathcal{P}_k/\mathcal{U}(\mathcal{P})} s'_{k_q} \to^{*}_{\mathcal{U}(\mathcal{P})} s'_{j_1} \to^{p}_{\mathcal{P}_j/\mathcal{U}(\mathcal{P})} s'_{j_p}$ so that the number of (weak) dependency pair steps is unchanged. The sequence $(\mathcal{P}_1, \dots, \mathcal{P}_j, \mathcal{P}_k, \dots, \mathcal{P}_m)$ is reorderable into $(\mathcal{P}_1, \dots, \mathcal{P}_k, \mathcal{P}_j, \dots, \mathcal{P}_m)$ without affecting the length ℓ of the $\to_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$-rewrite sequence. By assumption $k = j + 1$, hence induction hypothesis becomes applicable and we conclude the existence of a path $(\mathcal{P}'_1, \dots, \mathcal{P}'_{m'})$ fulfilling the assertions of the lemma. □

Finally, we arrive at the main contribution of this paper.

Theorem 23. *Let $\mathcal{R}$ be a TRS, let $\mathcal{P}$ be the set of weak or weak innermost dependency pairs, let A denotes the maximum arity of compound symbols and let K denotes the number of SCCs in the weak (innermost) dependency graph $\mathcal{G}$. Suppose $t \in \mathcal{T}^{\sharp}_{\mathsf{b}}$ is (innermost) terminating and define*

$$L(t) := \max\{\mathsf{dl}(t, \to_{\mathcal{P}_m/\mathcal{S}}) \mid (\mathcal{P}_1, \dots, \mathcal{P}_m) \text{ is a path in } \mathcal{G}/_{\equiv} \text{ such that } \mathcal{P}_1 \in \mathsf{Src}\} \;,$$

where $\mathcal{S} = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_{m-1} \cup \mathcal{U}(\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_m)$. *Then* $\mathsf{dl}(t, \xrightarrow{(\mathrm{i})}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}) \leqslant A^K \cdot K \cdot L(t)$.

Proof. Let $(\mathcal{P}_1, \ldots, \mathcal{P}_m)$ be a path in $\mathcal{P}/_{\equiv}$ such that $\mathcal{P}_1 \in \mathsf{Src}$ and let $D\colon t \to^{\ell} u$, denote a maximal derivation based on this path. (Here $\to$ denotes $\to_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$ or $\xrightarrow{\mathrm{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$.) Lemma 22 yields that D has the following form:

$$t = t_0 \to^{\ell_1}_{\mathcal{P}_1/\mathcal{U}(\mathcal{P}_1)} t_{\ell_1} \to^{\ell_2}_{\mathcal{P}_2/\mathcal{U}(\mathcal{P}_1)\cup\mathcal{U}(\mathcal{P}_2)} \cdots \to^{\ell_m}_{\mathcal{P}_m/\mathcal{U}(\mathcal{P}_1)\cup\cdots\cup\mathcal{U}(\mathcal{P}_m)} t_n = u\,, \quad (4)$$

where $t_0 \in \mathcal{T}^{\sharp}_{\mathsf{b}}$ and $t_i \in \mathcal{T}^{\sharp}_{\mathsf{c}}$ for all $i \geqslant 1$. It suffices to estimate ℓ_j for all $j = 1, \ldots, m$ suitably. Let j be arbitrary, but fixed. Consider the subderivation D' of (4) where m is replaced by j. Clearly D' is contained in the following derivation:

$$t \to^{*}_{\mathcal{P}_1\cup\cdots\cup\mathcal{P}_{i-1}\cup\mathcal{U}(\mathcal{P}_1)\cup\cdots\cup\mathcal{U}(\mathcal{P}_{j-1})} \cdot \to^{\ell_j}_{\mathcal{P}_j/\mathcal{U}(\mathcal{P}_1)\cup\cdots\cup\mathcal{U}(\mathcal{P}_j)} t_{\ell_j}$$

Hence Lemma 17 is applicable, thus $\ell_j \leqslant \mathsf{dl}(t, \to_{\mathcal{P}_j/\mathcal{P}_1\cup\cdots\cup\mathcal{P}_{j-1}\cup\mathcal{U}(\mathcal{P}_1)\cup\cdots\cup\mathcal{U}(\mathcal{P}_j)})$. As $\mathcal{U}(\mathcal{P}_1) \cup \cdots \cup \mathcal{U}(\mathcal{P}_j) \subseteq \mathcal{U}(\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_j)$ we conclude $\ell_j \leqslant L(t)$ and obtain $\ell = \ell_1 + \ell_2 + \cdots + \ell_m \leqslant K \cdot L(t)$.

Above we argued that any connected component in $\mathcal{P}/_{\equiv}$ is a tree. Clearly the number of nodes in this tree is less than $\frac{A^K-1}{A-1}$ and an arbitrary derivation can at most be based on $\frac{A^K-1}{A-1}$-many different paths. As the length of a derivation D based on a specific path can be estimated by $K \cdot L(t)$, we conclude that the length of an arbitrary derivation is less than $\frac{A^K-1}{A-1} \cdot K \cdot L(t) \leqslant A^K \cdot K \cdot L(t)$. This completes the proof of the theorem. □

Theorem 23 together with Proposition 9 form a suitable analog of Theorem 13: Let $\mathcal{P}$ be the set of weak or weak innermost dependency pairs. Suppose for every path $(\mathcal{P}_1, \ldots, \mathcal{P}_m)$ in $\mathcal{P}$ there exist an SLI $\mathcal{A}_m$ compatible with the usable rules of $\bigcup_{i=1}^{m} \mathcal{P}_i$. Assume the existence of a safe and G-collapsible reduction pairs $(\gtrsim_m, \succ_m)$ such that $\mathcal{U}(\bigcup_{i=1}^{m} \mathcal{P}_i) \cup \bigcup_{i=1}^{m-1} \mathcal{P}_i$ is compatible with $\gtrsim_m$ and $\mathcal{P}_m$ compatible with $\succ_m$. Then for any $t \in \mathcal{T}_{\mathsf{b}}$ the derivation height $\mathsf{dl}(t, \xrightarrow{(\mathrm{i})})$ with respect to (innermost) rewriting is linearly bounded in $\mathsf{G}(t^{\sharp}, \succ_m)$ and $|t|$.

Corollary 24. *Let $\mathcal{R}$ be a TRS, let $\mathcal{P}$ be the set of weak (innermost) dependency pairs, and let $\mathcal{G}$ denote the weak (innermost) dependency graph. Suppose for every path $(\mathcal{P}_1, \ldots, \mathcal{P}_m)$ in $\mathcal{G}/_{\equiv}$ there exist an SLI $\mathcal{A}_m$ and linear (quadratic) restricted interpretations $\mathcal{B}_m$ such that $(\geqslant_{\mathcal{B}_m}, >_{\mathcal{B}_m})$ forms a safe reduction pair with (i) $\mathcal{U}(\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_m) \subseteq >_{\mathcal{A}_m}$ (ii) $\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_{m-1} \cup \mathcal{U}(\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_m) \subseteq \geqslant_{\mathcal{B}_m}$, and (iii) $\mathcal{P}_m \subseteq >_{\mathcal{B}_m}$. Then the runtime complexity of a TRS $\mathcal{R}$ is linear or quadratic, respectively.*

Proof. Observe that the assumptions imply that any basic term $t \in \mathcal{T}_{\mathsf{b}}$ is terminating with respect to $\mathcal{R}$: Any infinite derivation with respect to $\mathcal{R}$ starting in t can be translated into an infinite derivation with respect to $\mathcal{U}(\mathcal{R}) \cup \mathcal{P}$ (see [1, Lemma 16]). Moreover, as the number of paths in $\mathcal{G}/_{\equiv}$ is finite, there exists a component $\mathcal{P}_i$ that represents an infinite rewrite sequence. This is a contradiction. Without loss of generality, we assume $\mathcal{P} = \mathsf{WDP}(\mathcal{P})$ and $\mathcal{G} = \mathsf{WDG}(\mathcal{P})$.

Notice that the reduction pair $(\geqslant_{\mathcal{B}_m}, >_{\mathcal{B}_m})$ is safe and collapsible. Hence for all m, the length of any $\to_{\mathcal{P}_m/\mathcal{S}}$-rewrite sequence is less than $p_m(|t|)$, where p_m denotes a linear (or quadratic) polynomial, depending on $|t|$ only. (Here $\mathcal{S} = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_{m-1} \cup \mathcal{U}(\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_m)$.) In analogy to the operator L, we define $M(t) := \max\{\mathsf{dl}(t, \to_{\mathcal{P}_m \cup \mathcal{S}}) \mid (\mathcal{P}_1, \ldots, \mathcal{P}_m) \text{ is a path in } \mathcal{G}/_{\equiv} \text{ such that } \mathcal{P}_1 \in \mathsf{Src}\}$. An application of Proposition 9 yields $M(t) = \mathsf{O}(p_m(|t|))$. Following the pattern of the proof of the Theorem, we establish the existence of a polynomial p such that $\mathsf{dl}(t, \to_{\mathcal{P} \cup \mathcal{U}(\mathcal{P})}) \leqslant p(|t|)$ holds for any basic term t. Finally, the corollary follows by an application of Proposition 8. □

As mentioned above, in the dependency graph refinement for termination analysis it suffices to guarantee for each cycle $\mathcal{C}$ that there exists no $\mathcal{C}$-minimal rewrite sequences. For that one only needs to find a reduction pair $(\gtrsim, \succ)$ such that $\mathcal{R} \subseteq \gtrsim$, $\mathcal{C} \subseteq \gtrsim$ and $\mathcal{C} \cap \succ \neq \varnothing$. Thus, considering Theorem 23 it is tempting to think that it should suffice to replace strongly connected components by cycles and the stronger conditions should apply. However this intuition is deceiving as shown by the next example.

Example 25. Consider the TRS $\mathcal{R}$ of $\mathsf{f}(\mathsf{s}(x), \mathsf{0}) \to \mathsf{f}(x, \mathsf{s}(x))$ and $\mathsf{f}(x, \mathsf{s}(y)) \to \mathsf{f}(x, y)$. $\mathsf{WDP}(\mathcal{R})$ consists of $1\colon \mathsf{f}^\sharp(\mathsf{s}(x), \mathsf{0}) \to \mathsf{f}^\sharp(x, \mathsf{s}(x))$ and $2\colon \mathsf{f}^\sharp(x, \mathsf{s}(y)) \to \mathsf{f}^\sharp(x, y)$, and the weak dependency graph $\mathsf{WDG}(\mathcal{R})$ contains two cycles $\{1, 2\}$ and $\{2\}$. There are two linear restricted interpretations $\mathcal{A}$ and $\mathcal{B}$ such that $\{1, 2\} \subseteq \geqslant_{\mathcal{A}} \cup >_{\mathcal{A}}$, $\{1\} \subseteq >_{\mathcal{A}}$, and $\{1\} \subseteq >_{\mathcal{B}}$. Here, however, we must not conclude linear runtime complexity, because the runtime complexity of $\mathcal{R}$ is at least quadratic.

5 Conclusion

In this section we provide (experimental) evidence on the applicability of the technique for complexity analysis established in this paper. We briefly consider the efficient implementation of the techniques provided by Theorem 23 and Corollary 24. Firstly, in order to approximate (weak) dependency graphs, we adapted (innermost) dependency graph estimations using the functions TCAP (ICAP) [14]. Secondly, note that a graph including n nodes may contain an exponential number of paths. However, to apply Corollary 24 it is sufficient to handle only paths in the following set. Notice that this set contains at most n^2 paths.

$$\{(\mathcal{P}_1, \ldots, \mathcal{P}_k) \mid (\mathcal{P}_1, \ldots, \mathcal{P}_m) \text{ is a maximal path and } k \leqslant m\}\,,$$

Example 26 (continued from Example 16). For $\mathsf{WDG}(\mathcal{R})/_{\equiv}$ the above set consists of 8 paths: $(\{13\})$, $(\{13\}, \{11\})$, $(\{13\}, \{12\})$, $(\{15\})$, $(\{15\}, \{14\})$, $(\{17\})$, $(\{18, 19, 20\})$, and $(\{18, 19, 20\}, \{16\})$. In the following we only consider the last three paths, since all other paths are similarly handled.

- Consider $(\{17\})$. Note $\mathcal{U}(\{17\}) = \varnothing$. By taking an arbitrary SLI $\mathcal{A}$ and the linear restricted interpretation $\mathcal{B}$ with $\mathsf{gcd}^\sharp_{\mathcal{B}}(x, y) = x$ and $\mathsf{s}_{\mathcal{B}}(x) = x + 1$, we have $\varnothing \subseteq >_{\mathcal{A}}$, $\varnothing \subseteq \geqslant_{\mathcal{B}}$, and $\{17\} \subseteq >_{\mathcal{B}}$.

Table 1. Results for Linear Runtime Complexities

		full rewriting			*innermost rewriting*		
	direct	Prop.8	Prop.10	Cor.24	Prop.8	Prop.10	Cor.24
success	139	138	119	137	143	128	147
			(161)	(179)		(170)	(189)
	15	*21*	*18*	*52*	*21*	*21*	*65*
failure	1535	1518	1560	1510	1511	1551	1499
	1789	*3185*	*152*	*1690*	*3214*	*214*	*1625*
timeout	5	23	0	32	25	0	33

- Consider $(\{18,19,20\})$. Note $\mathcal{U}(\{18,19,20\}) = \{1,\dots,5\}$. By taking the SLI $\mathcal{A}$ and the linear restricted interpretation $\mathcal{B}$ with $\mathsf{0}_\mathcal{A} = \mathsf{true}_\mathcal{A} = \mathsf{false}_\mathcal{A} = 0$, $\mathsf{s}_\mathcal{A}(x) = x+1$, $x -_\mathcal{A} y = x \leqslant_\mathcal{A} y = x+y+1$; $\mathsf{0}_\mathcal{B} = \mathsf{true}_\mathcal{B} = \mathsf{false}_\mathcal{B} = x \leqslant_\mathcal{B} y = 0$, $\mathsf{s}_\mathcal{B}(x) = x+2$, $x -_\mathcal{B} y = x$, $\mathsf{gcd}^\sharp_\mathcal{B}(x,y) = x+y+1$, and $\mathsf{if}_{\mathsf{gcd}}{}^\sharp_\mathcal{B}(x,y,z) = y+z$, we obtain $\{1,\dots,5\} \subseteq >_\mathcal{A}$, $\{1,\dots,5\} \subseteq \geqslant_\mathcal{B}$, and $\{18,19,20\} \subseteq >_\mathcal{B}$.
- Consider $(\{18,19,20\},\{16\})$. Note $\mathcal{U}(\{16\}) = \varnothing$. By taking the same $\mathcal{A}$ and also $\mathcal{B}$, we have $\{1,\dots,5\} \subseteq >_\mathcal{A}$, $\{1,\dots,5,18,19,20\} \subseteq \geqslant_\mathcal{B}$, and $\{16\} \subseteq >_\mathcal{B}$.

Thus, all path constraints are handled by linear restricted interpretations. Hence, the runtime complexity function of $\mathcal{R}$ is linear.

Moreover, to deal efficiently with polynomial interpretations, the issuing constraints are encoded in *propositional logic* in a similar spirit as in [16]. Assignments are found by employing a state-of-the-art SAT solver, in our case MiniSat[7]. Furthermore, SLIs are handled by linear programming. Based on these ideas we implemented a complexity analyser. As suitable test bed we used the rewrite systems in the Termination Problem Data Base version 4.0.[8] 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 60 seconds. In interpreting defined and dependency pair symbols, we restrict the search to polynomials in the range $\{0,1,\dots,5\}$. Table 1 (2) shows the experimental results for linear (quadratic) runtime complexities based on linear (quadratic) restricted interpretations.[9] Text written in *italics* below the number of successes or failures indicates total time (in seconds) of success cases or failure cases, respectively.[10] The columns marked "Prop. 10" and "Cor. 24" refer to the applicability of the respective results. For sake of comparison, in the parentheses we indicate the number of successes by the method of the column *or* by Proposition 8.

In concluding, we observe that the experimental data shows that the here introduced dependency graph refinement for complexity analysis extends the analytic power of the methods introduced in [1]. Notice the significant difference between those TRSs that can be handled by Propositions 8, 10 in contrast to those that can be handled either by Proposition 8 or by Corollary 24. Moreover observe the gain in power in relation to direct methods, compare also [3,4].

[7] `http://minisat.se/`.
[8] See `http://www.lri.fr/~marche/tpdb/`.
[9] For full experimental evidence see `http://www.jaist.ac.jp/~hirokawa/08b/`.
[10] Sum of numbers in each column may be less than 1679 because of stack overflow.

Table 2. Results for Quadratic Runtime Complexities

		full rewriting			*innermost rewriting*		
	direct	Prop.8	Prop.10	Cor.24	Prop.8	Prop.10	Cor.24
success	179	172	125	141	172	126	146
			(191)	(210)		(192)	(213)
	623	*732*	*295*	*616*	*725*	*278*	*787*
failure	745	699	1499	1434	699	1496	1431
	4431	*4522*	*1128*	*2883*	*4536*	*1062*	*2856*
timeout	753	807	55	104	807	57	102

References

1. Hirokawa, N., Moser, G.: Automated complexity analysis based on the dependency pair method. In: Proc. 4th IJCAR. Volume 5195 of LNCS. (2008) 364–379
2. Lescanne, P.: Termination of rewrite systems by elementary interpretations. Formal Aspects of Computing **7**(1) (1995) 77–90
3. Bonfante, G., Cichon, A., Marion, J.Y., Touzet, H.: Algorithms with polynomial interpretation termination proof. JFP **11**(1) (2001) 33–53
4. Avanzini, M., Moser, G.: Complexity analysis by rewriting. In: Proc. 9th FLOPS. Volume 4989 of LNCS. (2008) 130–146
5. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. TCS **236** (2000) 133–178
6. Arts, T., Giesl, J.: A collection of examples for termination of term rewriting using dependency pairs. Technical Report AIB-2001-09, RWTH Aachen (2001)
7. Giesl, J., Arts, T., Ohlebusch, E.: Modular termination proofs for rewriting using dependency pairs. JSC **34**(1) (2002) 21–58
8. Hirokawa, N., Middeldorp, A.: Tyrolean termination tool: Techniques and features. IC **205** (2007) 474–511
9. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and improving dependency pairs. JAR **37**(3) (2006) 155–203
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. Hofbauer, D., Lautemann, C.: Termination proofs and the length of derivations. In: Proc. 3rd RTA. Volume 355 of LNCS. (1989) 167–177
13. Contejean, E., Marché, C., Tomás, A.P., Urbain, X.: Mechanically proving termination using polynomial interpretations. JAR **34**(4) (2005) 325–363
14. Giesl, J., Thiemann, R., Schneider-Kamp, P.: Proving and disproving termination of higher-order functions. In: Proc. 5th FroCoS. Volume 3717 of LNAI. (2005) 216–231
15. Hirokawa, N., Middeldorp, A.: Automating the dependency pair method. IC **199**(1,2) (2005) 172–199
16. 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