Automation of Polynomial Path Orders

master thesis in computer science

by

Martin Avanzini

submitted to the Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck

in partial fulfillment of the requirements for the degree of Master of Science

supervisor: Dr. Georg Moser, Institute of Computer Science

Innsbruck, 25 February 2009

Master Thesis

Automation of Polynomial Path Orders

Martin Avanzini
Martin.Avanzini@student.uibk.ac.at

25 February 2009

Supervisor: Dr. Georg Moser

Abstract

The automated complexity analysis of rewrite systems is a challenging topic that has recently gained much attention. As one practical application, techniques developed for the analysis of rewrite systems can in principle be employed for the automated analysis of functional programs. In this thesis, we present the *polynomial path order* $>_{\mathsf{pop}*}$, a syntactic restriction of the well known multiset path order. The order is carefully crafted to induce polynomial bounds on the *runtime-complexity* of compatible rewrite systems. Semantic labeling and the dependency pair method are two prominent transformation techniques developed in the context of termination analysis. Both significantly strengthen the power of basic termination techniques. We show that suitable adaptations as proposed in the literature can be employed together with $>_{\mathsf{pop}*}$ for a complexity analysis. This severely widens the applicability of the polynomial path order. Furthermore, we give an efficient automation that works by a reduction to the Boolean satisfiability problem (SAT). Experimental results confirm the feasibility of our approach.

Acknowledgments

Foremost, special thanks to my supervisor Georg Moser for his valuable feedback, dedicated guidance and ongoing support over the last years. I must have interrupted him thousands of times from important work, and he always pretended he did not mind. Thank you Georg.

Great thanks to Nao Hirokawa who teached me not to be afraid of proofs, a fact reflected in this thesis. I want to thank Andreas Schnabl for the many fruitful conversations we had. Many thanks to Martin Korp, Christian Sternagl and Harald Zankl providing their excellent term rewriting libraries, technical assistance included. In particular, thank you Christian for helping me solve many of the LaTeX related problems.

I want to thank my family for their ongoing support, and for providing an environment that allowed me to focus on my studies. Without their help, the thesis presented here would not exist. Last but not least, I want to thank my girlfriend Daniela who had to dispense so many things over the last couple of years.

Contents

1 Introduction

Term rewriting is a branch of theoretical computer science which combines elements of logic, automated theorem proving and declarative programming. Term rewriting is akin to equational logic, where a term rewrite system $\mathcal{R}$ (or TRS for short) is conceivable as a set of directed equations. The rewrite relation $\to_{\mathcal{R}}$ as induced by the TRS $\mathcal{R}$ is defined in such a way that $s \to_{\mathcal{R}} t$ holds when t can be obtained from s by applying one of the equations from $\mathcal{R}$ from left to right. Rewriting constitutes a Turing-complete model of computation. One desired property of a TRS $\mathcal{R}$ is *termination*, that is $\to_{\mathcal{R}}$ does not give rise to infinite reductions. Consequently, termination has been studied quite early, and powerful methods for proving termination have been established over the years. Early research concentrated mainly on *direct termination techniques*, for instance the use of *reduction orders* like the *multiset path order* (MPO for short) [20] or *polynomial interpretations* [38]. A reduction order $\succ$ is a well-founded order on terms such that the inclusion $\mathcal{R} \subseteq \succ$ entails $\to_{\mathcal{R}} \subseteq \succ$, and thus termination of $\mathcal{R}$. In recent years, the attention shifted towards transformation techniques. The *dependency pair method* [5] and *semantic labeling* [58] are particular popular instances of transformation techniques. Both methods significantly increase the possibility to verify termination automatically.

Once we have established termination of a given rewrite system $\mathcal{R}$, it seems natural to direct the attention to the analysis of the *complexity* of $\mathcal{R}$. The *complexity analysis* in the context of rewriting is a challenging but compelling task with many applications. Below we render two applications we address in this thesis:

(i) The characterization of complexity classes by means of *rewriting characterizations* has attained a lot of attention [11, 17, 46]. Results established for complexity analysis of rewrite systems give rise to alternative characterizations (cf. for instance [4, 14]), simplified proofs and the analysis of the *implicit computational complexity of functions* (cf. for instance [16, 33, 49], or [13, 40, 42] in the context of rewriting).

(ii) A first order functional program P can often be understood as orthogonal constructor rewrite system $\mathcal{R}_{\mathsf{P}}$ [10, 41]. We highlight this correspondence on a small ML-program, defining the append function `@` on lists:

```
let (@) xs ys =
  match xs with
    | []     -> ys
    | x::xs -> x :: (xs @ ys)
```

The computation of the append function is encoded by the rewrite system

$\mathcal{R}_{\texttt{@}}$ consisting of the two rewrite rules

$$\begin{aligned} \texttt{[]} \mathbin{\texttt{@}} ys &\to ys \\ (x \mathbin{\texttt{::}} xs) \mathbin{\texttt{@}} ys &\to x \mathbin{\texttt{::}} (xs \mathbin{\texttt{@}} ys) \end{aligned}$$

in a very natural way. The evaluation of an expression e just amounts to the reduction of the corresponding term under $\mathcal{R}_{\texttt{@}}$. In other words, $\mathcal{R}_{\texttt{@}}$ *computes* the function `@` as defined in the ML-program.

As evaluation steps of such functional programs just correspond to rewrite steps of the respective rewrite system, the complexity analysis of the latter gives rise to an (automated) complexity analysis of the former.

In rewriting, the complexity of a rewrite system $\mathcal{R}$ is usually measured by the maximal length of derivations. Let the relation $\to$ refer to some (rewrite) relation on terms. The relation $\to$ could for instance be the rewrite relation $\to_{\mathcal{R}}$ as induced by the TRS $\mathcal{R}$, or the *innermost rewrite relation* $\xrightarrow{\text{i}}_{\mathcal{R}}$. The latter denotes the restriction of $\to_{\mathcal{R}}$ where innermost terms need to be reduced first, also referred to as *eager evaluation* in the functional programming community. With the *derivation length* $\mathrm{dl}(t, \to)$ of a term t with respect to $\to$ we refer to the length of the longest possible derivation starting from t. Let T denote a set of terms, the so called *start terms.* The *runtime-complexity function* rc based on T and $\to$ is defined as

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

for a reasonable size-measure $|\cdot|$ on terms. Then $\mathrm{rc}(n, T, \to)$ gives the length of the longest $\to$-chain starting from a term in T of size up to n.

The derivational complexity of a rewrite system $\mathcal{R}$ is given by the *derivational complexity function* $\mathrm{dc}_{\mathcal{R}}$ defined by

$$\mathrm{dc}_{\mathcal{R}}(n) = \mathrm{rc}(n, \mathcal{T}, \to_{\mathcal{R}})$$

where $\mathcal{T}$ refers to the set of all possible terms. A program proposed by Hofbauer and Lautemann [32] is to measure the power of termination methods by the induced *derivational complexity.* For example, termination proofs by multiset path orders induce primitive recursive upper bounds on $\mathrm{dc}_{\mathcal{R}}$ [31]. Similar results have been established for other reduction orders [32, 39, 55]. To the contrary, little is known about state-of-the-art termination techniques such as the dependency pair method or semantic labeling in their general setting.

Viewed from the opposite direction, a proof of termination inherits vital information about the complexity of the considered rewrite system. For instance, the inclusion $\mathcal{R} \subseteq >_{\mathsf{mpo}}$ certifies a primitive recursive derivational complexity function $\mathrm{dc}_{\mathcal{R}}$. Moreover such an analysis is easy to automate. However, the reported results on induced derivational complexity by standard techniques are of a rather theoretical value. For the complexity analysis of a functional program as highlighted above, one is usually interested in lower complexity bounds that assess *feasible* computability. Even the smallest bound given in [32, 39, 55], namely doubly-exponential as induced by polynomial interpretations, cannot be considered as computationally feasible. One recent exception is presented in [26]. Here linear derivational complexity can be verified by the use of automata techniques.

For the study of lower bounds, the derivational complexity function $\mathrm{dc}_{\mathcal{R}}$ is not well suited. In particular, this holds if a rewrite system $\mathcal{R}_f$ is conceived (as above) as a description of an algorithm, specifying the evaluation of a functions f. In this thesis, we follow [7, 29] and adopt the *innermost runtime-complexity function* $\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}$ as complexity measure for rewrite systems $\mathcal{R}$. It is defined as

$$\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}(n) = \mathrm{rc}(n, \mathcal{T}_{\mathsf{b}}, \xrightarrow{\mathsf{i}}_{\mathcal{R}})$$

where $\mathcal{T}_{\mathsf{b}}$ refers to the set of all *constructor-based* terms and $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ to the innermost rewrite relation as induced by $\mathcal{R}$. Here $\mathcal{T}_{\mathsf{b}}$ contain those terms that correspond to a function-call on values, that is terms of the form $f(v_1, \dots, v_n)$ where f denotes a *defined function symbol* and the terms v_i are *values*. The definition of the (innermost) runtime-complexity function dispels two obstacles:

(i) The derivational complexity function does not discriminate between different start terms, whereas this is natural for the runtime-complexity analysis of functions. We highlight the difference in the following TRS $\mathcal{R}_{\mathsf{double}}$ defined by the rules

$$\mathsf{double}(0) \to 0 \qquad\qquad \mathsf{double}(\mathsf{s}(n)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(n))) \ .$$

The above rewrite system *computes* the function $\mathsf{double}(n) = 2n$, that is the term[1] $\mathsf{double}(\mathsf{s}^n(0))$ reduces to $\mathsf{s}^{2n}(0)$, where naturals are represented as tally numbers. Although the computed function double is clearly feasible—even computable in linearly many steps—the derivational complexity of $\mathcal{R}_{\mathsf{double}}$ is exponential. On the other hand, for the runtime-complexity analysis of $\mathcal{R}_{\mathsf{double}}$, only terms of the shape $\mathsf{double}(\mathsf{s}^n(0))$ corresponding to the function call $\mathsf{double}(n)$ with $n \in \mathbb{N}$ are considered. The runtime-complexity of $\mathcal{R}_{\mathsf{double}}$ is linear as desired.

(ii) Unrestricted rewriting may be too powerful for the study of lower complexity classes, even if we restrict the analysis to constructor-based terms. This has already been observed by Beckmann and Weiermann [11]. In particular, an unwise reduction strategy may give rise to unneeded duplication of reducible terms (i.e. computation). Eager evaluation, that is rewriting under the innermost rewrite relation $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$, naturally circumvents duplication of computation.

In this thesis, we present the *polynomial path order* (or POP* for short) first introduced in [7] and give fruitful extensions. In essence, the polynomial path order is a carefully crafted miniaturization of MPO, incorporating the schemes of *predicative recursion* [12]. Predicative recursion was introduced by Bellantoni and Cook to break the strength of primitive recursion in their recursion-theoretic characterization of the polytime computable functions (**FP** for short).

[1]Here we employ the convention $f^n(t)$ to denote the term $f(f(\dots f(t) \dots))$ with n occurrences of f.

In the definition of the polynomial path order, the same idea is employed to break the strength of MPO that induces primitive recursive derivational complexity. The combination of MPO with predicative recursion gives rise to a completely automatic runtime-complexity analysis of rewrite systems: we show that *compatibility* of a rewrite system $\mathcal{R}$ with a polynomial path order $>_{\mathsf{pop}*}$ certifies a polynomially bounded *innermost runtime-complexity* of $\mathcal{R}$. More precisely, the inclusion $\mathcal{R} \subseteq >_{\mathsf{pop}*}$ entails that the innermost runtime-complexity function $\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}$ is bounded by a polynomial. This result is supplemented by an alternative characterization of the functions computable in polynomial time (**FP** for short). Opposed to the classical recursion-theoretic characterization [18], our characterization is entirely free of resource bounds: the polytime-computable functions can be understood as exactly those functions computed by certain syntactically restricted rewrite systems compatible with polynomial path orders $>_{\mathsf{pop}*}$. The polynomial path order is also of practical interest. As one application, such an analysis can in principle be employed for an automated complexity analysis of (eager) first-order functional programs.

The polynomial path order is a purely syntactic method. That is, it operates by a purely structural analysis of the input. This is in strong contrast to the majority of existing methods, for instance restricted forms of polynomial interpretations (employed for example in [13]) or *matrix interpretations* [22]. Intuitively, here one searches for an interpretation of the symbols appearing in $\mathcal{R}$ that certify feasible bounds on the length of derivations. The efficient implementation of such semantic techniques is always a major concern, whereas polynomial path orders are *extremely fast*. This is reflected in our experimental results.

The polynomial path order is strongly based on the *path order for **FP*** introduced by Arai and Moser in [4]. A central motivation for this research is the observation that the direct application of the latter order is only successful on a handful of (very simple) rewrite systems. The path order for **FP** gains only power if additional transformations are performed. Unfortunately, such powerful transformations are difficult to find automatically. In Chapter 3 we introduce the reader to the path order for **FP** in a slightly generalized setting. This order provides us with a suitable foundation for the reasoning carried out later on.

The polynomial path order is a restriction of MPO. Thus its power cannot exceed the power of MPO. As briefly mentioned, transformation techniques have significantly increased the possibility to automatically prove termination. This gives rise to the question whether we can combine the polynomial path order with transformation techniques employed for the termination analysis. We answer this question positively in Chapter 6, where we investigate a combination of the polynomial path order with semantic labeling and a variant of the dependency pair method [29, 30].

Although semantic labeling preserves the length of derivation, only partial results on derivational complexities are reported [44]. The main problem is that the transformed system $\mathcal{R}_{\mathsf{lab}}$ may admit an infinite signature. For this case, the results mentioned above on polynomial runtime-complexities induced by

POP* fails (as do most of the results mentioned above on derivational complexities induced by reduction orders). For finite systems $\mathcal{R}_{\mathsf{lab}}$ the situation changes. In this thesis, we employ a restricted form of semantic labeling, dubbed *finite semantic labeling.* This variant asserts that $\mathcal{R}_{\mathsf{lab}}$ is finite. By finite semantic labeling we significantly increase the strength of the polynomial path order. This is confirmed by experimental evidence presented in this thesis.

In recent efforts [29], Hirokawa and Moser provide an adaption of the dependency pair method as proposed in [5] suitable for a runtime-complexity analysis. This method enables the use of several powerful techniques such as *argument filterings* [37], *usable rules* and *reduction pairs.* Moreover, in [30] the framework has been extended by *dependency graphs.* For the integration of the polynomial path order into the dependency pair framework for complexity analysis we face several difficulties. In order to tackle these problems, and to simplify the presentation, we introduce in Chapter 5 the polynomial path order in the context of *relative rewriting* [45]. Relative rewriting is a notion generalizing rewriting, where the corresponding *relative rewrite relation* $\to_{\mathcal{R}/\mathcal{S}}$ is formed by two rewrite systems $\mathcal{R}$ and $\mathcal{S}$. We show that a pair $(\gtrsim_{\mathsf{pop}*}, >_{\mathsf{pop}*})$ provides us with the means to give polynomial bounds on the runtime-complexity with respect to the relative rewrite relation. Moreover, we present a generalization of the polynomial path order $>^{\pi}_{\mathsf{pop}*}$ by incorporating argument filtering. In particular, the pair $(\gtrsim^{\pi}_{\mathsf{pop}*}, >^{\pi}_{\mathsf{pop}*})$ certifies polynomial bounds (in the size of the start term) on the runtime-complexity with respect to the relative rewrite relation $\to_{\mathcal{R}/\mathcal{S}}$, provided all steps due to $\mathcal{R}$ happen at the root position. We show in Chapter 6 that the latter result allows the use of the polynomial path order together with argument filterings in the variant of the dependency pair method from [29]. Moreover, this approach integrates seamless to the extension of [29] to dependency graphs [30], but for brevity we concentrate solely on [29].

Beside those theoretical issues, we give an *efficient* implementation of the polynomial path order together with the above mentioned transformations. This is a challenging task, as the search space is enormous. For the integration of argument filterings with recursive path orders like MPO, standard approaches exist [19, 56]. We mainly follow these, suitable adapted for our concerns. On the other hand, although efficient techniques for the integration of (variants of) semantic labeling exist [34, 35], we cannot follow these approaches. This is due to the fact that the considered models are usually infinite.

The thesis is structured as follows: In Chapter 2 we revise central definitions and introduce the reader to the employed theory of rewriting. In Chapter 3 we present the path order for **FP** in a slightly more general setting, presenting a suitable foundation for the reasoning carried out in this thesis. In Chapter 4 we introduce the polynomial path order. Moreover, following [8] we give an alternative characterization of **FP**. In Chapter 5 we briefly explain the notion of relative rewriting. We incorporate argument filterings in the base order and we show that polynomial path orders can be employed for the analysis of relative steps. In Chapter 6 we present POP* combined with dependency pairs and semantic labeling as indicated above. In Chapter 7 we investigate the technical issues arising from the implementation of the polynomial path order. In particular, we give an automation together with semantic labeling and argument filterings. In Chapter 8 we briefly contrast the polynomial path order to related research, and we conclude in Chapter 9.

2 Preliminaries

Below, we introduce the reader to the employed terminology and recall the basic concepts of term rewriting. Although helpful, familiarity with rewriting is not assumed. For further information, [10, 52] provide good resources.

2.1 Orders

Definition 2.1. A *proper order* is an irreflexive and transitive binary relation. A *preorder* is a reflexive and transitive binary relation. An *equivalence relation* is reflexive, symmetric and transitive. A binary relation R is *well-founded* if there exists no infinite chain $a_0, a_1, \ldots$ with $a_i \; R \; a_{i+1}$ for all $i \in \mathbb{N}$. Moreover, we say that R is well-founded on a set A if there exists no such infinite chain with $a_0 \in A$. R is *finitely branching* if for all elements a, the set $\{b \mid a \; R \; b\}$ is finite.

For a binary relation R, we write R^+ to denote the transitive, $R^=$ for the reflexive and R^* for the transitive and reflexive closure of R. Every preorder $\succsim$ induces a proper order $\succ$, namely $a \succ b$ if and only if $a \succsim b$ and not $b \succsim a$, and an equivalence relation $\sim$, namely $a \sim b$ if and only if $a \succsim b$ and $b \succsim a$. For a preorder $\succsim$, we usually denote the induced equivalence relation by $\sim$ and induced proper order by $\succ$. Obviously, $\succsim \; = \; \succ \cup \sim$ holds for these.

Definition 2.2. Let A be a set and $\sim$ an equivalence relation on A. The *quotient of A modulo $\sim$* is defined as the set $A/{\sim} = \{[a] \mid a \in A\}$ where $[a]$ is the *$\sim$-equivalence class of* a, i.e. $[a] = \{b \in A \mid b \sim a\}$. Suppose $\succsim \; = \; \succ \cup \sim$ is a binary relation over A satisfying $\sim \cdot \succ \cdot \sim \; \subseteq \; \succsim$ where $\sim$ is an equivalence relation. The *extension of $\succ$ to $A/{\sim}$* is defined as the binary relation $\sqsupset$ with $[a] \sqsupset [b]$ if and only if $a \succ b$.

A *multiset* M is a collection in which elements are allowed to occur more than once. In our context M is always finite.

Definition 2.3. Let A be a set. A *multiset* M is a function from A into $\mathbb{N}$ such that $M(a) = 0$ for all but finitely many elements $a \in A$, where $M(a)$ denotes how often a occurs in M. The set of all multisets over A is denoted by $\mathcal{M}(A)$.

We sometimes write $[a_1, \ldots, a_n]$ for the multiset M with elements $a_1, \ldots, a_n$, for instance we write $[1, 1, 2]$ for the multiset containing 1 twice and 2 once. The usual set operations are extended to multisets as follows: we write $a \in M(a)$ if $M(a) > 0$ and denote the *empty multiset* M, i.e. $M(a) = 0$ for all $a \in A$, by $\varnothing$. For $M_1, M_2 \in \mathcal{M}(A)$ the *sum* $M_1 \uplus M_2$ is defined as the multiset M such that $M(a) = M_1(a) + M_2(a)$ for all $a \in A$. Likewise the *difference* $M_1 \backslash M_2$ is given by M such that $M(a) = M_1(a) - M_2(a)$ if $M_1(a) \geqslant M_2(a)$ and $M(a) = 0$

otherwise. The inclusion $M_1 \subseteq M_2$ holds if for some multiset M, $M_1 \uplus M = M_2$. We write $\supseteq$ for the converse of $\subseteq$ and define equality ($=$) on multisets as the relation $\supseteq \cap \subseteq$.

Definition 2.4. Let $\succ$ be a binary relation on a set A. The *multiset extension* $\succ_{\mathsf{mul}}$ of $\succ$ is the binary relation on $\mathcal{M}(A)$ such that $M_1 \succ_{\mathsf{mul}} M_2$ if there exists multisets $X, Y \in \mathcal{M}(A)$ satisfying

(i) $M_2 = (M_1 \backslash X) \uplus Y$,

(ii) $\varnothing \neq X \subseteq M_1$ and

(iii) for all $y \in Y$ there exists an element $x \in X$ such that $x \succ y$.

In particular, if $\succ$ is a proper order over A then $\succ_{\mathsf{mul}}$ is a proper order over $\mathcal{M}(A)$ and moreover $\succ_{\mathsf{mul}}$ is well-founded on $\mathcal{M}(A)$ if and only if $\succ$ is well-founded on A. We follow [23] and generalize the above definition of multiset extension to a binary relation by identifying equivalent elements:

Definition 2.5. Let $\sim$ denote an equivalence relation and let $\succsim \, = \, \succ \cup \sim$ be a binary relation over a set A such that $\sim \cdot \succ \cdot \sim \; \subseteq \; \succsim$. Suppose $\sqsupset$ denotes the extension of $\succ$ to the quotient $A/{\sim}$ of the $\sim$-equivalence classes. Define for multiset $M \in \mathcal{M}(A)$ the corresponding multiset $M^{\sim} \in \mathcal{M}(A/{\sim})$ such that $M^{\sim}([a]) = \Sigma_{b \in [a]} M(b)$ for all $a \in A$. The *strict multiset extension* $\succ_{\mathsf{mul}}$ of $\succsim$ is defined such that $M_1 \succ_{\mathsf{mul}} M_2$ if and only if $M_1^{\sim} \sqsupset_{\mathsf{mul}} M_2^{\sim}$. Furthermore, we define equivalence by $M_1 \sim_{\mathsf{mul}} M_2$ if and only if $M_1^{\sim} = M_2^{\sim}$. We define $\succsim_{\mathsf{mul}} = \succ_{\mathsf{mul}} \cup \sim_{\mathsf{mul}}$ and call $\succsim_{\mathsf{mul}}$ the *weak multiset extension of* $\succsim$.

Notice that when $\succsim$ is a preorder on A then the strict multiset extension $\succ_{\mathsf{mul}}$ of $\succsim$ is a proper order and the weak multiset extension $\succsim_{\mathsf{mul}}$ a preorder on $\mathcal{M}(A)$. Moreover, $\succ_{\mathsf{mul}}$ is well-founded if and only if $\succ$ is well-founded.

2.2 Term Rewriting

Throughout the thesis, with $\mathcal{V}$ we denote a countably infinite set of *variables.*

Definition 2.6. A *(variadic) signature* $\mathcal{F}$ is a set of *function symbols* such that $\mathcal{F} \cap \mathcal{V} = \varnothing$, where each $f \in \mathcal{F}$ is associated with one or more natural numbers, the *arities* of f. For the case when $\mathcal{F}$ associates more than one arity to a symbol $f \in \mathcal{F}$, we call f *variadic.*

For a symbol $f \in \mathcal{F}$ and associated arity n, we say that f is n-ary. We call 0-ary function symbols *constants.* For readability, we write c instead of $c()$ for constants. We always use $\mathcal{F}$ to denote a signature and if not mentioned otherwise then $\mathcal{F}$ is finite and non-variadic, i.e. contains no variadic symbols. If not stated otherwise, $f, g, h, \ldots$ denote elements from $\mathcal{F}$. From function symbols and variables, we build *terms* as follows.

Definition 2.7. The set of *terms* over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$ and is inductively defined by

(i) $\mathcal{V} \subseteq \mathcal{T}(\mathcal{F},\mathcal{V})$, and

(ii) $f(t_1,\dots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{V})$ for all n-ary function symbols $f \in \mathcal{F}$ and terms $t_1,\dots,t_n \in \mathcal{T}(\mathcal{F},\mathcal{V})$.

If not mentioned otherwise, we use $s,t,u,v,\dots$, possibly extended by subscripts, to denote elements from $\mathcal{T}(\mathcal{F},\mathcal{V})$ and abbreviate $\mathcal{T}(\mathcal{F},\mathcal{V})$ by $\mathcal{T}$. We denote the set of all *ground terms* $\mathcal{T}(\mathcal{F},\varnothing)$ by $\mathcal{T}(\mathcal{F})$.

By $\mathsf{Var}(t) \subseteq \mathcal{V}$ we denote the set of all variables appearing in t, likewise by $\mathsf{Fun}(t) \subseteq \mathcal{F}$ we denote the set of all function symbols appearing in t. We denote by $\mathrm{root}(t)$ the *root symbol* of t, that is $\mathrm{root}(f(t_1,\dots,t_n)) = f$ and $\mathrm{root}(t) = t$ else. In the upcoming definition, we define the different size-measures for terms as employed in this thesis.

Definition 2.8. The *size* $|t|$ of a term t is recursively given by

$$|t| = \begin{cases} 1 + \Sigma_{i=1}^{n}|t_i| & \text{if } t = f(t_1,\dots,t_n) \\ 1 & \text{otherwise.} \end{cases}$$

The *width* of a term t is recursively given by

$$\mathrm{width}(t) = \begin{cases} \max\{n, \mathrm{width}(t_1),\dots,\mathrm{width}(t_n)\} & \text{if } t = f(t_1,\dots,t_n) \text{ and } n > 0 \\ 1 & \text{otherwise.} \end{cases}$$

The *depth* of a term t is recursively defined by

$$\mathrm{depth}(t) = \begin{cases} 1 + \max\{\mathrm{depth}(t_1),\dots,\mathrm{depth}(t_n)\} & \text{if } t = f(t_1,\dots,t_n) \text{ and } n > 0 \\ 0 & \text{otherwise.} \end{cases}$$

The *Buchholz-norm* $\|t\|$ of term t is recursively defined by

$$\|t\| = \begin{cases} 1 + \max\{n, \|t_1\|,\dots,\|t_n\|\} & \text{if } t = f(t_1,\dots,t_n) \\ 1 & \text{otherwise.} \end{cases}$$

A *position* is a finite sequence of positive natural numbers. The positions of a term t are collected in $\mathcal{P}\mathrm{os}(t)$ as follows.

Definition 2.9. The set $\mathcal{P}\mathrm{os}(t)$ of all positions in a term t is inductively defined as follows:

$$\mathcal{P}\mathrm{os}(t) = \begin{cases} \{\varepsilon\} & \text{if } t \in \mathcal{V}, \\ \{\varepsilon\} \ \cup \ \{ip \mid 1 \leq i \leq n \ \text{ and } p \in \mathcal{P}\mathrm{os}(t_i)\} & \text{if } t = f(t_1,\dots,t_n)\ . \end{cases}$$

Here the root position is denoted by the empty sequence ε and pq denotes the concatenation of positions p and q.

Definition 2.10. Given a term t, the *subterm* of t at position $p \in \mathcal{P}\mathrm{os}(t)$ is denoted by $t|_p$ and defined as

- $t|_\varepsilon = t$, and

- $t|_{ip} = t_i|_p$ if $t = f(t_1, \ldots, t_n)$.

We write $s \trianglelefteq t$ if $s = t|_p$ for some position $p \in \mathcal{P}\mathrm{os}(t)$ and call s a *subterm* of t. For the case when $s \neq t$ we write $s \lhd t$ and call s a *proper subterm* of t. We denote by $\trianglerighteq$ the converse of $\trianglelefteq$ and by $\rhd$ the converse of $\lhd$ respectively.

A *substitution* σ is a mapping from $\mathcal{V}$ into $\mathcal{T}(\mathcal{F}, \mathcal{V})$ and we write $t\sigma$ for the term extension of the mapping in the following sense.

Definition 2.11. Let σ be a mapping from $\mathcal{V}$ into $\mathcal{T}(\mathcal{F}, \mathcal{V})$, called a *substitution*. Let t be a term. We extend σ to terms by

- $t\sigma = \sigma(t)$ if $t \in \mathcal{V}$, and
- $t\sigma = f(t_1\sigma, \ldots, t_n\sigma)$ if $t = f(t_1, \ldots, t_n)$.

A bijective substitution from $\mathcal{V}$ to $\mathcal{V}$ is called a *renaming*. Below, we always use σ to denote a substitution.

Definition 2.12. A binary relation R on terms is *closed under substitutions* if whenever $s\ R\ t$ holds then $s\sigma\ R\ t\sigma$ holds for any substitution σ.

Consider a fresh constant symbol $\square$, named *hole*. Terms that contain holes are called *contexts*.

Definition 2.13. We call an element $C \in \mathcal{T}(\mathcal{F} \cup \{\square\}, \mathcal{V})$ a *context*. Let C be a n-hole context and let $p_1, \ldots, p_n$ denote all positions in $\mathcal{P}\mathrm{os}(C)$ such that $C|_{p_i} = \square$, sorted in lexicographic order. For terms $t_1, \ldots, t_n$, we denote by $C[t_1, \ldots, t_n]$ the term obtained by replacing the holes at position p_i with t_i. Let $\mathcal{X}$ be a set of symbols. We write $C\langle t_1, \ldots, t_n\rangle_{\mathcal{X}}$ to denote $C[t_1, \ldots, t_n]$, whenever $\mathrm{root}(t_i) \in \mathcal{X}$ and C is a n-hole context containing no symbols from $\mathcal{X}$ (observe that here C may be degenerated, that is C does not contain a hole).

The context $\square$ is called the *empty context*.

Definition 2.14. Let R be a binary relation on terms. R is *closed under contexts* if whenever $s\ R\ t$ holds then $C[s]\ R\ C[t]$ holds for any context C with exactly one hole. If in addition R is closed under substitutions, then R is called a *rewrite relation*.

A proper order R on terms that is also a rewrite relation is called a *rewrite order*. When R is well-founded in addition, then R is called a *reduction order*. Reduction orders are of particular interest for the termination analysis of term rewrite systems. We briefly introduce two prominent reduction orders below, but before we recall term rewrite systems and their properties.

Definition 2.15. A *rewrite rule* is a pair (ℓ, r) of terms such that $\ell, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, $\ell \notin \mathcal{V}$ and $\mathsf{Var}(r) \subseteq \mathsf{Var}(\ell)$. For a rewrite rule (ℓ, r), we write $\ell \to r$ instead.

Definition 2.16. A *term rewrite system* (*TRS* for short) is a tuple $(\mathcal{F}, \mathcal{R})$ where $\mathcal{F}$ is a signature and $\mathcal{R}$ is a set of rewrite rules over $\mathcal{T}(\mathcal{F}, \mathcal{V})$.

We usually abbreviate $(\mathcal{F}, \mathcal{R})$ by $\mathcal{R}$ when $\mathcal{F}$ is clear from context. If not stated otherwise, we use $\mathcal{R}$, $\mathcal{S}$, $\mathcal{Q}$ and $\mathcal{P}$ to denote rewrite systems. The smallest extension of $\mathcal{R}$ that is a rewrite relation is denoted by $\rightarrow_{\mathcal{R}}$.

Definition 2.17. Let $\mathcal{R}$ be a TRS. The *rewrite relation* $\rightarrow_{\mathcal{R}}$ is the binary relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $s \rightarrow_{\mathcal{R}} t$ if and only if there exists a one-hole context C, substitution σ and position p such that $C|_p = \Box$, $s = C[\ell\sigma]$ and $t = C[r\sigma]$ for some rule $\ell \rightarrow r \in \mathcal{R}$.

For a *step* $s \rightarrow_{\mathcal{R}} t$ we say that s *rewrites to* t in one step and call t a *reduct* of s. In order to indicate the rewrite position p of a step $s \rightarrow_{\mathcal{R}} t$ we write $s \rightarrow^{p}_{\mathcal{R}} t$. We call a step $s \rightarrow^{\varepsilon}_{\mathcal{R}} t$ a *top-step*. The term s is a *normal form* of $\mathcal{R}$ if there exists no term t such that $s \rightarrow_{\mathcal{R}} t$ holds. We collect all normal forms of $\mathcal{R}$ in the set $\mathsf{NF}(\mathcal{R})$. When $s \rightarrow^{*}_{\mathcal{R}} t$ and $t \in \mathsf{NF}(\mathcal{R})$ then t is called a normal form of s and we write $s \rightarrow^{!}_{\mathcal{R}} t$ in this case. The *innermost rewrite relation* $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ is the restriction of $\rightarrow_{\mathcal{R}}$ where innermost terms have to be reduced first:

Definition 2.18. Let $\mathcal{R}$ be a TRS. The *innermost rewrite relation* $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ is defined such that $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$ if and only if $s = C[f(\ell_1\sigma, \ldots, \ell_n\sigma)]$, $t = C[r\sigma]$ and $\ell_i\sigma \in \mathsf{NF}(\mathcal{R})$ for all $i \in \{1, \ldots, n\}$. Here C denotes a one-hole context, σ a substitution and $f(\ell_1, \ldots, \ell_n) \rightarrow r \in \mathcal{R}$.

Definition 2.19. A TRS $\mathcal{R}$ is *terminating* if $\rightarrow_{\mathcal{R}}$ is well-founded.

We say that a rewrite order $\succ$ is *$\mathcal{R}$-compatible* (and vice versa) if the inclusion $\mathcal{R} \subseteq \succ$ holds. Notice that compatibility of $\mathcal{R}$ with a reduction order asserts termination of $\mathcal{R}$.

Definition 2.20. Two terms s and t are *joinable* if there exists a term u such that $s \rightarrow^{*}_{\mathcal{R}} u$ and $t \rightarrow^{*}_{\mathcal{R}} u$. A TRS $\mathcal{R}$ is *confluent* if for each term s each pair of reducts t_1 and t_2 is joinable.

Every terminating and confluent system admits the *unique normal form* property, that is $s \rightarrow^{!}_{\mathcal{R}} t_1$ and $s \rightarrow^{!}_{\mathcal{R}} t_2$ implies $t_1 = t_2$. A rewrite system $\mathcal{R}$ is *left-linear* if for each rule $\ell \rightarrow r \in \mathcal{R}$ each variable appears at most once in ℓ. If $\ell \rightarrow r$ is a rewrite rule and σ a renaming then the rewrite rule $\ell\sigma \rightarrow r\sigma$ is called a *variant* of $\ell \rightarrow r$.

Definition 2.21. A rewrite system is *orthogonal* if it is left-linear and does not give rise to overlaps in the following sense: An *overlap* of a TRS $\mathcal{R}$ is a triple $\langle \ell_1 \rightarrow r_1, p, \ell_2 \rightarrow r_2 \rangle$ satisfying

(i) $\ell_1 \rightarrow r_1$ and $\ell_2 \rightarrow r_2$ are variants of rules from $\mathcal{R}$ without common variables,

(ii) $\ell_1|_p \notin \mathcal{V}$,

(iii) $\ell_1|_p$ and ℓ_2 are unifiable, and

(iv) if $p = \varepsilon$ then $\ell_1 \rightarrow r_1$ and $\ell_2 \rightarrow r_2$ are not variants.

It is well-known that each orthogonal system is confluent [10]. In the following, let $|t|_a$ denote the number of appearances of the symbol a in t.

Definition 2.22. A TRS $\mathcal{R}$ is called *duplicating* if some rule $\ell \to r \in \mathcal{R}$ is *duplicating*, that is there exists a variable $x \in \mathsf{Var}(\ell)$ such that $|r|_x > |\ell|_x$

We assume a partitioning of $\mathcal{F}$ into two disjoint sets, denoted by $\mathcal{D}$ and $\mathcal{C}$. A function symbol $f \in \mathcal{D}$ is called a *defined symbol*, whereas elements from $\mathcal{C}$ are called *constructors*.

Definition 2.23. Let $\mathcal{D} \uplus \mathcal{C}$ be a partitioning of $\mathcal{F}$ into defined symbols and constructors. A term t is called *constructor-based* if $t = f(t_1, \ldots, t_n)$, $f \in \mathcal{D}$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C}, \mathcal{V})$.

We denote the set of all constructor-based terms by $\mathcal{T}_{\mathsf{b}}(\mathcal{F}, \mathcal{V})$. For brevity, we write $\mathcal{T}_{\mathsf{b}}$ for $\mathcal{T}_{\mathsf{b}}(\mathcal{F}, \mathcal{V})$ when the signature can be inferred from context. Likewise we abbreviate the set $\mathcal{T}(\mathcal{C}, \mathcal{V})$ by Val and sometimes call elements from Val *values*.

Definition 2.24. The rewrite system $(\mathcal{F}, \mathcal{R})$ induces a partitioning on $\mathcal{F}$ into defined symbols $\mathcal{D}$ and constructors $\mathcal{C}$ as follows: the set $\mathcal{D}$ is obtained by collecting all root symbols of left-hand sides in $\mathcal{R}$, that is

$$\mathcal{D} = \{f \mid f(\ell_1, \ldots, \ell_n) \to r \in \mathcal{R}\} \; .$$

Furthermore, $\mathcal{C}$ is given by $\mathcal{F} \setminus \mathcal{D}$. The TRS $(\mathcal{F}, \mathcal{R})$ is called a *constructor TRS* if every left-hand side in $\mathcal{R}$ is constructor-based, i.e. $\ell \in \mathcal{T}_{\mathsf{b}}$ for all $\ell \to r \in \mathcal{R}$ with respect to the partitioning $\mathcal{D} \uplus \mathcal{C}$.

Definition 2.25. Let $\mathcal{R}$ be a TRS. A defined function symbol is *completely defined* if it does not occur in any normal form, i.e. $f \notin \mathsf{Fun}(t)$ for all $t \in \mathsf{NF}(\mathcal{R})$. A TRS is *completely defined* if each defined symbol $f \in \mathcal{D}$ is completely defined.

Observe that for a completely defined system $\mathcal{R}$, normal forms and constructor terms coincide, i.e. $\mathsf{NF}(\mathcal{R}) = \mathsf{Val}$.

Definition 2.26. Let $\to$ be a finitely branching and well-founded binary relation. The *derivation length* of a terminating term t with respect to $\to$ is defined as

$$\mathrm{dl}(t, \to) = \max\{n \mid \exists s.\; t \to^n s\} \; .$$

Let T be a set of terms over a finite signature $\mathcal{F}$, and assume $\to$ is well-founded on T. The *runtime-complexity function* with respect to $\to$ and T is defined as follows:

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

In particular we are interested in the runtime-complexity with respect to $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ on the set $\mathcal{T}_{\mathsf{b}}$ of all constructor-based terms.

Definition 2.27. The *innermost runtime-complexity function* of a rewrite system $\mathcal{R}$ is defined as $\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}(n) = \mathrm{rc}(n, \mathcal{T}_{\mathsf{b}}, \xrightarrow{\mathsf{i}}_{\mathcal{R}})$.

We say the runtime-complexity of $\mathcal{R}$ is *linear*, *quadratic*, or *polynomial* if $\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}(n)$ is bounded linearly, quadratically, or polynomially in n, respectively.

Definition 2.28. An *$\mathcal{F}$-algebra* $\mathcal{A}$ is a non-empty set A—the so called *carrier* of $\mathcal{A}$—equipped with operations $f_{\mathcal{A}} : A^n \to A$ for every n-ary symbol $f \in \mathcal{F}$. A mapping α from $\mathcal{V}$ into A is called an *assignment* for $\mathcal{A}$. The *interpretation* of a term t under an assignment α is inductively defined as follows:

$$[\alpha]_{\mathcal{A}}(t) = \begin{cases} \alpha(t) & \text{if } t \in \mathcal{V} \\ f_{\mathcal{A}}([\alpha]_{\mathcal{A}}(t_1), \ldots, [\alpha]_{\mathcal{A}}(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \; . \end{cases}$$

Definition 2.29. A *monotone $\mathcal{F}$-algebra* is a pair $(\mathcal{A}, \succ)$ where $\mathcal{A}$ is an $\mathcal{F}$-algebra, $\succ$ a well-founded proper order on the carrier A of $\mathcal{A}$ and all operations $f_{\mathcal{A}}$ are *strictly monotone* in all arguments, i.e.

$$f(a_1, \ldots, a_i, \ldots, a_n) \succ f(a_1, \ldots, b, \ldots, a_n)$$

for all $a_1, \ldots, a_n \in A$, $i \in \{1, \ldots, n\}$ and b with $a_i \succ b$. We call $(\mathcal{A}, \succ)$ *well-founded* if $\succ$ is well-founded.

$\mathcal{F}$-algebras give rise to reduction orders in a natural way:

Definition 2.30. Let $(\mathcal{A}, \succ)$ be a monotone and well-founded $\mathcal{F}$-algebra. We write $\succ_{\mathcal{A}}$ for the reduction order defined by $s \succ_{\mathcal{A}} t$ if and only if $[\alpha]_{\mathcal{A}}(s) \succ [\alpha]_{\mathcal{A}}(t)$ for all assignments α.

As a special instance, an $\mathcal{F}$-algebras $\mathcal{A}$ is called a *polynomial interpretations* if (i) the carrier is $\mathbb{N}$ equipped with the usual order $>$ on naturals, and (ii) all operations $f_{\mathcal{A}}$ are monotone polynomials. Clearly, polynomial interpretations are monotone and well-founded $\mathcal{F}$-algebra. We frequently make use the following restricted form of polynomial interpretations:

Definition 2.31. We call a polynomial $p(x_1, \ldots, x_n) = \Sigma_{i=1}^{n} x_i + c$ with $c \in \mathbb{N}$ a *weight functions.* A *strongly linear interpretation* $\mathcal{A}$ is a polynomial interpretation where all interpretation functions $f_{\mathcal{A}}$ are weight functions.

Other instances of reduction orders are so called *recursive path orders.* These are induced by *precedences.*

Definition 2.32. A *strict precedence* $\succ$ is a proper order, a *quasi-precedence* $\succsim$ a preorder on a (possible variadic or infinite) signature $\mathcal{F}$. A well-founded precedence $\succ$ (quasi-precedence $\succsim$) induces a *rank* $\mathrm{rank}(f)$ on $f \in \mathcal{F}$ as follows:

$$\mathrm{rank}(f) = 1 + \max\{\mathrm{rank}(g) \mid g \in \mathcal{F} \text{ and } f \succ g\} \; .$$

Here we employ the convention that the maximum of the empty set is 0, and call a quasi-precedence $\succsim$ well-founded when the induced proper order $\succ$ is well founded.

Let $\succsim$ be a quasi-precedence. We lift equivalence $\sim$ on function symbols as induced by $\succsim$ to terms as in the following definition.

Definition 2.33. Let $\succsim$ be a quasi-precedence. We say that two terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ are *equivalent* with respect to $\succsim$, written by $s \sim t$, if either

(i) $s = t$, or

(ii) $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_n)$, $f \sim g$, and there exists a permutation π such that $s_i \sim t_{\pi(i)}$ holds for all $i \in \{1, \ldots, n\}$.

An instance of a recursive path order is the *multiset path order* (*MPO* for short). We render the definition of a multiset path order $>_{\mathsf{mpo}}$ as follows.

Definition 2.34. The *multiset path order* $>_{\mathsf{mpo}}$ induced by the precedence $\succsim$ is inductively defined by $s >_{\mathsf{mpo}} t$ for $s = f(s_1, \ldots, s_n)$ if one of the following alternatives hold:

(i) there exists $i \in \{1, \ldots, n\}$ such that $s_i >_{\mathsf{mpo}} t$ or $s_i \sim t$,

(ii) $t = g(t_1, \ldots, t_m)$, $f \succ g$ and $s >_{\mathsf{mpo}} t_j$ for all $j \in \{1, \ldots, m\}$, or

(iii) $t = g(t_1, \ldots, t_m)$, $f \sim g$ and $\{s_1, \ldots, s_n\} >^{\mathsf{mul}}_{\mathsf{mpo}} \{t_1, \ldots, t_m\}$.

Here $>^{\mathsf{mul}}_{\mathsf{mpo}}$ denotes the strict multiset extension of $>_{\mathsf{mpo}} \cup \sim$.

It is a well known fact that for well-founded precedences $\succsim$, the order $>_{\mathsf{mpo}}$ is a reduction order. Thus compatibility of $\mathcal{R}$ with an instance $>_{\mathsf{mpo}}$ certifies termination of $\mathcal{R}$.

Definition 2.35. A TRS $\mathcal{R}$ is called *precedence terminating* if there exists a well-founded precedence $\succsim$ such that $\mathrm{root}(\ell) \succ f$ for every rewrite rule $\ell \to r \in \mathcal{R}$ and every function symbol $f \in \mathsf{Fun}(r)$.

Observe that every precedence terminating system can be shown terminating with multiset path orders.

Finally, we sometimes refer to *many-sorted rewrite systems*. For that, let S be a finite set representing the set of *types* or *sorts*. An *S-sorted set A* is a family of sets $\{A_s \mid s \in S\}$ such that all sets A_s are pairwise disjoint. We write $\mathcal{V}_S$ for a S-sorted set of variables with $\mathcal{V} = \bigcup_{s \in S} \mathcal{V}_s$. A *$S$-sorted signature* $\mathcal{F}_S$ is like a (non-variadic) signature, but the *arity* of $f \in \mathcal{F}_S$ is defined by $\mathrm{arity}(f) = (s_1, \ldots, s_n)$ for $s_1, \ldots, s_n \in S$. Additionally, each symbol $f \in \mathcal{F}_S$ is associated with a sort $s \in S$, called the *type of f* and denoted by $\mathrm{type}(f)$.

Definition 2.36. The *S-sorted set of terms* $\mathcal{T}(\mathcal{F}, \mathcal{V})_S$ consists of the sets $\mathcal{T}(\mathcal{F}, \mathcal{V})_s$ for $s \in S$, where $\mathcal{T}(\mathcal{F}, \mathcal{V})_s$ is inductively defined by

(i) $\mathcal{V}_s \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})_s$, and

(ii) $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})_s$ for all function symbols $f \in \mathcal{F}_S$ with $\mathrm{arity}(f) = (s_1, \ldots, s_n)$, $\mathrm{type}(f) = s$ and terms $t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})_{s_i}$ for $i \in \{1, \ldots, n\}$.

Definition 2.37. A S-sorted term rewrite system $\mathcal{R}$ is a TRS such that for $\ell \to r \in \mathcal{R}$, it holds that $\ell, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})_s$ for some sort $s \in S$.

3 The Polynomial Path Order on Sequences

Arai and Moser proposed in [4] the *path order for the polytime computable functions* (path order for **FP** for short). The order is carefully crafted to induce polynomial bounds on the runtime-complexity. More precise, let $\blacktriangleright$ denote an instance of the path order for **FP**, and assume the inclusion $\mathcal{R} \subseteq \blacktriangleright$ holds. Then for a defined symbol f there exists a polynomial p such that for every sequence

$$f(v_1, \ldots, v_n) = t_0 \rightarrow_{\mathcal{R}} t_1 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t_\ell = r$$

and values $v_1, \ldots, v_n$ it follows that $\ell \leqslant p(|f(v_1, \ldots, v_n)|)$. Moreover, the definition of the path order for **FP** gives rise to a new characterization of the polynomial time computable functions **FP**. In [4] it has been shown that every function from **FP** is *computable* by a syntactically restricted form of TRSs that is compatible with an instance of the path order for **FP**, and vice versa every such TRS computes a function from **FP**.

The order is centered around the idea of *predicative recursion*. As already observed by Cobham in 1964, unrestricted recursion is in general to strong for **FP**. In his characterization of the polytime-computable functions [18], Cobham circumvents the problem by weakening the recursion scheme suitably. This is done by demanding a polynomial size-bound on recursively computed results. Almost thirty years later, Bellantoni and Cook provided a new recursion-characterization of the polytime computable functions [12]. In particular, their definition does not rely on any externally imposed resource bounds. In order to break the strength of the recursion on notation scheme, they differentiate between *safe* and *normal* argument positions. To highlight this separation, we write $f(\vec{x};\vec{y})$ instead of $f(\vec{x},\vec{y})$ where $\vec{x}$ appear at normal and $\vec{y}$ at safe argument positions of f. A new function f is defined by *predicative recursion on notation* via the equations

$$f(0,\vec{x};\vec{y}) = g(\vec{x};\vec{y})$$
$$f(zi,\vec{x};\vec{y}) = h_i(z,\vec{x};\vec{y},f(z,\vec{x};\vec{y}))$$

for $i \in \{0,1\}$ and previously defined functions g, h_0 and h_1. Observe that recursion is performed on a normal argument position, whereas the recursively computed result is substituted into a safe argument position. In particular this implies that the stepping functions h_i cannot perform recursion on the recursively computed $f(z,\vec{x};\vec{y})$. The separation of safe and normal argument positions is enforced by the *predicative composition scheme*, where a function f is defined by composition via the equation

$$f(\vec{x};\vec{y}) = h(\vec{r}(\vec{x};);\vec{s}(\vec{x};\vec{y})) .$$

As shown in [12], starting from a suitable set of initial functions, predicative recursion and composition generate exactly those functions computable in polynomial time.

The separation of safe and normal arguments is implicitly employed in the definition of the path orders for **FP**. Roughly, a call $f(x_1,\dots,x_n;y_1,\dots,y_m)$ is represented by a *sequence* of the form $(f(x_1,\dots,x_n)\ y_1 \cdots y_m)$ where the safe arguments y_i are singled out. Naturally, if one wants to show compatibility of a TRS $\mathcal{R}$ with path orders for **FP**, then the rewrite system $\mathcal{R}$ either needs to follow this representation or be appropriately transformed. Unfortunately, such (automatic) transformations are difficult to find. This led to the definition of the polynomial path order $>_{\mathsf{pop}*}$.

Compatibility of $>_{\mathsf{pop}*}$ with a TRS $\mathcal{R}$ gives vital information that can be used for an *embedding* of $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ into the path order for **FP**. The results established in [4] can then be employed for estimating the derivation length of a term t with respect to $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$. Below, we present a slight generalization of the path order for **FP** suitable for our concerns, duped *polynomial path order* $\blacktriangleright$ *on sequences.* We show that the crucial main theorem from [4] carries over to this generalization. This was already observed in [7, 8] for $\blacktriangleright$ as induced by strict precedences. Although the extension to quasi-precedences is indeed not difficult, for the sake of completeness we reformulate an adaption to quasi-precedences below.

Let $\mathcal{F}$ be a signature and $\circ \notin \mathcal{F}$ be a variadic function symbol. We extend the signature $\mathcal{F}$ by $\circ$ and write $\mathcal{S}\text{eq}(\mathcal{F},\mathcal{V}) = \mathcal{T}(\mathcal{F}\cup\{\circ\},\mathcal{V})$. We refer to elements of $\mathcal{S}\text{eq}(\mathcal{F},\mathcal{V})$ as *sequences.* For brevity, we may write $\mathcal{S}\text{eq}$ for $\mathcal{S}\text{eq}(\mathcal{F},\mathcal{V})$. Instead of $\circ(s_1,\dots,s_n)$ we write $(s_1\cdots s_n)$ and we write $\varnothing$ for the empty sequence $()$. We define concatenation of sequences $a,b \in \mathcal{S}\text{eq}$ by $\varnothing \mathbin{@} b = b$, $a \mathbin{@} \varnothing = a$ and $(a_1\cdots a_n) \mathbin{@} (b_1\cdots b_m) = (a_1\cdots a_n\ b_1\cdots b_m)$ otherwise. In the following we write $\succsim$ for a quasi-precedence on $\mathcal{F}$.

The polynomial path order $\blacktriangleright$ on sequences is a miniaturization of the multiset path order. The definition of $\blacktriangleright$ is based on an auxiliary order $\gtrdot$. This separation is necessary to break the strength of MPO. We render the orders $\blacktriangleright$ and $\gtrdot$ in the following two definitions. Recall that we use $\succ$ and $\sim$ for the proper and equivalence relation as induced by the quasi-precedence $\succsim$.

Definition 3.1. We define $\gtrdot$ induced by the precedence $\succsim$ inductively by $s \gtrdot t$ for $s = f(s_1,\dots,s_n)$ or $s = (s_1\cdots s_n)$ if one of the following alternatives holds:

(1) $s_i \gtrsim t$ for some $i \in \{1,\dots,n\}$, or

(2) $s = f(s_1,\dots,s_n)$, $t = (t_1\cdots t_m)$ or $t = g(t_1,\dots,t_m)$ with $f \succ g$ and $s \gtrdot t_j$ for all $j \in \{1,\dots,m\}$, or

(3) $s = (s_1\cdots s_n)$, $t = (t_1\cdots t_m)$, and $[s_1,\dots,s_n] \gtrdot_{\mathsf{mul}} [t_1,\dots,t_m]$.

Here $\gtrsim$ refers to $\gtrdot \cup \sim$ and $\gtrdot_{\mathsf{mul}}$ denotes the strict multiset extension of $\gtrsim$.

Definition 3.2. We define the *polynomial path order* $\blacktriangleright$ *on sequences* induced by the precedence $\succsim$ inductively by $s \blacktriangleright t$ for $s = f(s_1,\dots,s_n)$ or $s = (s_1\cdots s_n)$ if one of the following alternatives holds:

(1) $s \gtrdot t$,

(2) $s_i \mathrel{\blacktriangleright\!\!\!\!\sim} t$ for some $i \in \{1, \ldots, n\}$,

(3) $s = f(s_1, \ldots, s_n)$, $t = (t_1 \cdots t_m)$, and the following two properties hold:
   - $s \blacktriangleright t_{j_0}$ for some $j_0 \in \{1, \ldots, m\}$, and
   - $s \gtrdot t_j$ for all $j \neq j_0$, or

(4) $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, $f \sim g$ and $(s_1 \cdots s_n) \blacktriangleright (t_1 \cdots t_m)$,

(5) $s = (s_1 \cdots s_n)$, $t = (t_1 \cdots t_m)$, and $[s_1, \ldots, s_n] \blacktriangleright_{\mathsf{mul}} [t_1, \ldots, t_m]$.

Again we write $\mathrel{\blacktriangleright\!\!\!\!\sim}$ for $\blacktriangleright \cup \sim$. Furthermore, $\blacktriangleright_{\mathsf{mul}}$ denotes the strict multiset extension of $\mathrel{\blacktriangleright\!\!\!\!\sim}$.

Buchholz was the first to observe that finite term rewrite systems compatible with recursive path orders $\succ$ are even compatible to *finite approximations* of $\succ$ [15]. This observation carries over to polynomial path orders over sequences. We introduce those approximations $\gtrdot^l_k$ and $\blacktriangleright^l_k$ for $\gtrdot$ and $\blacktriangleright$ respectively below. These are just suitable restrictions of $\gtrdot$ and $\blacktriangleright$, where the parameters k and l control the growth of width and height in descents with respect to the corresponding order.

Definition 3.3. For $k, l \geqslant 1$, we define $\gtrdot^l_k$ induced by the precedence $\succsim$ inductively by $s \gtrdot^l_k t$ for $s = f(s_1, \ldots, s_n)$ or $s = (s_1 \cdots s_n)$ if one of the following alternatives holds:

(1) $s_i \mathrel{\gtrsim}^l_k t$ for some $i \in \{1, \ldots, n\}$,

(2) $s = f(s_1, \ldots, s_n)$ if $t = g(t_1, \ldots, t_m)$ with $f \succ g$ or $t = (t_1 \cdots t_m)$,
   - $s \gtrdot^{l-1}_k t_j$ for all $j \in \{1, \ldots, m\}$,
   - and $m < k + \mathrm{width}(s)$, or

(3) $s = (s_1 \cdots s_n)$, $t = (t_1 \cdots t_m)$ and the following properties hold:
   - $[t_1, \ldots, t_m] = N_1 \uplus \cdots \uplus N_n$, and
   - there exists $i \in \{1, \ldots, n\}$ such that $[s_i] \not\gtrdot_{\mathsf{mul}} N_i$, and
   - for all $i \in \{1, \ldots, n\}$ such that $[s_i] \not\gtrdot_{\mathsf{mul}} N_i$ we have $s_i \gtrdot^l_k r$ for all $r \in N_i$, and
   - $m < k + \mathrm{width}(s)$.

Definition 3.4. For $k, l \geqslant 1$, we define the *approximation of the polynomial path order $\blacktriangleright^l_k$ on sequences* induced by the precedence $\succsim$ inductively as follows: $s \blacktriangleright^l_k t$ for $s = f(s_1, \ldots, s_n)$ or $s = (s_1 \cdots s_n)$ if one of the following alternatives holds:

(1) $s \gtrdot^l_k t$,

(2) $s_i \mathrel{\blacktriangleright\!\!\!\!\sim}^l_k t$ for some $i \in \{1, \ldots, n\}$,

(3) $s = f(s_1, \ldots, s_n)$, $t = (t_1 \cdots t_m)$, and the following properties hold:
   - $s \blacktriangleright^{l-1}_k t_{j_0}$ for some $j_0 \in \{1, \ldots, m\}$,

- $s >_k^{l-1} t_j$ for all $j \neq j_0$, and
- $m < k + \text{width}(s)$, or

(4) $s = f(s_1, \dots, s_n)$, $t = g(t_1, \dots, t_m)$, $f \sim g$ and $(s_1 \cdots s_n) \blacktriangleright_k^l (t_1 \cdots t_m)$, or

(5) $s = (s_1 \cdots s_n)$, $t = (t_1 \cdots t_m)$ and the following properties hold:

- $[t_1, \dots, t_m] = N_1 \uplus \cdots \uplus N_n$, and
- there exists $i \in \{1, \dots, n\}$ such that $[s_i] \not\succ_{\mathsf{mul}} N_i$, and
- for all $i \in \{1, \dots, n\}$ such that $[s_i] \not\succ_{\mathsf{mul}} N_i$ we have $s_i \blacktriangleright_k^l r$ for all $r \in N_i$, and
- $m < k + \text{width}(s)$.

In the following we write $\blacktriangleright_k$ to abbreviate $\blacktriangleright_k^k$ and likewise $>_k$ for $>_k^k$. As a direct consequence of the definition, it follows that $\blacktriangleright = \bigcup_{k \in \mathbb{N}} \blacktriangleright_k$. We start with some simple observations.

Lemma 3.5.

(i) $\blacktriangleright_k$ *is closed under context, that is if* $s \blacktriangleright_k t$ *then* $C[s] \blacktriangleright_k C[t]$ *with one-hole context* C *over the signature* $\mathcal{F} \cup \{\circ\}$.

(ii) $\blacktriangleright_k \subseteq \blacktriangleright_l$ *for* $k \leqslant l$.

(iii) The orders $\sim$ *and* $\blacktriangleright_k$ *are compatible, that is the inclusions* $\sim \cdot \blacktriangleright_k \subseteq \blacktriangleright_k$ *and* $\blacktriangleright_k \cdot \sim \subseteq \blacktriangleright_k$ *hold.*

Proof. The first observations follows by a standard inductive argument, and is essentially due to the clauses (4) and (5) from the definition of $\blacktriangleright_k$. The inclusion $\blacktriangleright_k \subseteq \blacktriangleright_l$ for $k \leqslant l$ follows directly from the definition. Finally, it is not difficult to see that an instance $\blacktriangleright_k$ can neither detect permutation of arguments nor can distinguish equally ranked function symbols $f \sim g$. Thus the last observation is immediate. □

Observe that an approximation $\blacktriangleright_k$ is not transitive due to the restrictions imposed on depth and width of terms. This is essential, as it controls the growth of sizes in $\blacktriangleright_k$-descending sequences: Whenever $s \blacktriangleright_k t$, then there exists a (uniform) constant c such that $\|t\| \leqslant \|s\| + c$ and thus, if we have a $\blacktriangleright_k$-descending sequence $s = t_0 \blacktriangleright_k t_1 \blacktriangleright_k \cdots \blacktriangleright_k t_\ell$ we conclude that $\|t_n\| \leqslant cn + \|s\|$ for $0 \leqslant n \leqslant \ell$. More precise, the following Lemma holds:

Lemma 3.6. *If* $s \blacktriangleright_k^l t$, *then* $\text{depth}(t) \leqslant \text{depth}(s) + l$, $\text{width}(t) < k + \text{width}(s)$ *and* $\|t\| \leqslant \|s\| + k + l$.

Proof. Observe that $\text{depth}(t) > \text{depth}(s)$ is only possible via the application of clause (3) from Definition 3.4 or clause (2) from Definition 3.3 respectively. It is easy to see that the number of applications of these clauses is controlled by l, yielding the desired result. By similar reasoning, we derive $\text{width}(t) < k + \text{width}(s)$, which follows due to the restrictions on clauses (3) and (5) from Definition 3.4 and clauses (2) and (3) from Definition 3.3 respectively. Finally, as $\|t\| \leqslant \text{depth}(t) + \text{width}(t)$ we conclude the last claim. □

Definition 3.7. We define

$$\mathsf{G}_k(t) = \max\{\ell \in \mathbb{N} \mid t = t_0 \blacktriangleright_k t_1 \blacktriangleright_k \cdots \blacktriangleright_k t_\ell\} \ .$$

Moreover, for $f \in \mathcal{F}$ we define

$$\mathsf{F}_{k,p}(m) = \max\{\mathsf{G}_k(f(t_1, \ldots, t_n)) \mid \mathrm{rank}(f) = p \wedge \sum_i \mathsf{G}_k(t_i) + n \leqslant m\} \ .$$

Lemma 3.8. $\mathsf{G}_k((s_1 \cdots s_n)) = n + \sum_{i=1}^n \mathsf{G}_k(s_i)$.

Proof. The assertion follows by course of value induction on the length of the sequence $(s_1 \cdots s_n)$. The base case $s = \varnothing$ follows trivially from minimality of $\varnothing$. For the inductive step, suppose $s = (s_1 \cdots s_n) \mathbin{@} (s_{n+1})$ and assume a maximal descent

$$s = u_0 \blacktriangleright_k u_1 \blacktriangleright_k \cdots \blacktriangleright_k u_\ell \ ,$$

i.e. $\mathsf{G}_k(s) = \ell$. The only non-pathological case is when $u_0 \blacktriangleright_k u_1$ follows due to clause (5) from Definition 3.4 (or clause (3) from Definition 3.3). In this case it suffices to realize that $\mathsf{G}_k(s) = \mathsf{G}_k((s_1 \cdots s_n)) + \mathsf{G}_k((s_{n+1}))$ and the claim follows directly from the induction hypothesis on $(s_1 \cdots s_n)$ and (s_{n+1}). □

Lemma 3.9. $\mathsf{G}_k((c_1 \cdots c_n)) = \sum_{i=1}^n \mathrm{rank}(c_i)$ *for constants* c_i, $i \in \{1, \ldots, n\}$.

Proof. From Lemma 3.8 we infer $\mathsf{G}_k(s) = n + \sum_{i=1}^n \mathsf{G}_k(c_i)$. Thus the Lemma follows with a proof of $\mathsf{G}_k(c) = \mathrm{rank}(c) - 1$ for an arbitrary constant c, which we show by induction on $\mathrm{rank}(c)$: First, assume $\mathrm{rank}(c) = 1$. Thus $c = \varnothing$ and by minimality of $\varnothing$ it follows that $\mathsf{G}_k(\varnothing) = 0$. For the inductive step, suppose by induction hypothesis $\mathsf{G}_k(c_j) = \mathrm{rank}(c_j) - 1$ for all c_j with $c \succ c_j$ in the precedence. It is easy to see that

$$\{t \mid c \blacktriangleright_k t\} = \{c_j \mid c \succ c_j\} \ ,$$

and so $\mathsf{G}_k(c) = \max\{1 + \mathsf{G}_k(c_j) \mid c \succ c_j\}$. By applying the induction hypothesis on the right-hand side, we see that

$$\mathsf{G}_k(c) = \max\{\mathrm{rank}(c_j) \mid c \succ c_j\} = \mathrm{rank}(c) - 1 \ .$$

□

The next theorem constitutes the main result of this chapter. In particular, it states that under the assumption that $\mathsf{G}_k(t_i)$ is polynomially bounded for $i \in \{1, \ldots, n\}$, then $\mathsf{G}_k(f(t_1, \ldots, t_n))$ is polynomially bounded.

Theorem 3.10. *For all* $k, p \in \mathbb{N}$, *let* $d_{k,p}$ *be recursively given by* $d_{k,0} = k + 1$ *and* $d_{k,p+1} = (d_{k,p})^k + 1$. *Then for all* $k, p \in \mathbb{N}$ *there exists a constant* c *such that*

$$F_{k,p}(m) \leqslant c(m + 2)^{d_{k,p}}$$

for all $m \in \mathbb{N}$. *The constant* c *depends only on* k *and* p.

Proof. Let $s = f(s_1, \dots, s_n)$ and assume $\mathrm{rank}(f) = p$ and $\sum_i \mathsf{G}_k(s_i) + n \leqslant m$. We proceed by main induction on p and side-induction on m. To show the theorem, we show that for all t with $s \blacktriangleright_k t$ there exists a constant c (depending only on k and p) such that $\mathsf{G}_k(t) < c(m+2)^{d_{k,p}}$ holds. From the assertion, the theorem follows easily by definition of $\mathsf{F}_{k,p}$:

$$\begin{aligned}
\mathsf{F}_{k,p}(m) &= \max\{\mathsf{G}_k(f(s_1, \dots, s_n)) \mid \mathrm{rank}(f) = p \wedge \sum_i \mathsf{G}_k(s_i) + n \leqslant m\} \\
&= 1 + \max\{\mathsf{G}_k(t) \mid f(s_1, \dots, s_n) \blacktriangleright_k t \\
&\qquad\qquad \wedge \mathrm{rank}(f) = p \wedge \sum_i \mathsf{G}_k(s_i) + n \leqslant m\} \\
&\leqslant c(m+2)^{d_{k,p}} \ .
\end{aligned}$$

Now we show $\mathsf{G}_k(t) < c(m+2)^{d_{k,p}}$ holds for all $m \geqslant 0$.

Base Case. First we show that $s \gtrdot_k^l r$ implies that $\mathsf{G}_k(r) \leqslant k^l(m+2)^l$ by induction on l. If $l = 1$, then $s \gtrdot_k^l r$ implies that r is a subterm of s, i.e., there exists $i \in \{1, \dots, n\}$ such that $s_i \gtrsim_k r$. It follows that $\mathsf{G}_k(r) \leqslant m \leqslant k(m+2)$. For the above, observe $\mathsf{G}_k(r) = \mathsf{G}_k(s_i)$ for $s_i \sim r$ is an easy consequence of $\sim \cdot \blacktriangleright_k \subseteq \blacktriangleright_k$ (cf. Lemma 3.5). Next, suppose $l > 1$ and $s \gtrdot_k^l r = (r_1 \cdots r_\ell)$. By induction hypothesis on l, we have $\mathsf{G}_k(r_i) \leqslant k^{l-1}(m+2)^{l-1}$ for all $i \in \{1, \dots, \ell\}$. Observe that $\ell < k + \mathrm{width}(s) \leqslant k(m+2)$ hold by definition and the fact that $\mathrm{width}(s) \leqslant m$. Using Lemma 3.8 we conclude

$$\begin{aligned}
\mathsf{G}_k(r) &\leqslant \ell + \ell k^{l-1}(m+2)^{l-1} \\
&= \ell(k^{l-1}(m+2)^{l-1} + 1) \\
&< (k(m+2) - 1)(k^{l-1}(m+2)^{l-1} + 1) \\
&\leqslant k^l(m+2)^l \ .
\end{aligned}$$

We continue the proof of the base case and set $c = k^k$. By side-induction hypothesis on m we conclude that $\mathsf{F}_{k,p}(m_1) \leqslant c(m_1+2)^{d_{k,p}}$ holds for all $m_1 < m$. Let $l \leqslant k$ and let $s \blacktriangleright_k^l r$. We show $\mathsf{G}_k(r) \leqslant c(m+1)^{k+1} + k^l(m+2)^l$ with induction on $s \blacktriangleright_k^l r$.

1. Suppose $s \gtrdot_k^l r$ holds. From the preparatory observations, we conclude $\mathsf{G}_k(r) \leqslant k^l(m+2)^l \leqslant c(m+1)^{k+1} + k^l(m+2)^l$.

2. Suppose there exists $i \in \{1, \dots, n\}$, such that $s_i \mathrel{\underset{\sim}{\blacktriangleright}}_k r$. Then $\mathsf{G}_k(r) \leqslant \mathsf{G}_k(s_i) \leqslant m \leqslant c(m+1)^{k+1} + k^l(m+2)^l$, where we additionally employ $\sim \cdot \blacktriangleright_k \subseteq \blacktriangleright_k$.

3. Suppose $r = (r_1 \cdots r_\ell)$ and there exists i_0 such that $s \blacktriangleright_k^{l-1} r_{i_0}$ and $s \gtrdot_k^{l-1} t_i$ for all $i \neq i_0$. Observe that $l-1 > 1$ by definition. From the preparatory step we see
$$\sum_{i \neq i_0} \mathsf{G}_k(r_i) \leqslant (\ell - 1)k^{l-1}(m+2)^{l-1} \ .$$
By induction hypothesis we have
$$\mathsf{G}_k(r_{i_0}) < c(m+1)^{k+1} + k^{l-1}(m+2)^{l-1} \ .$$

In total, using again $\ell < k(m+2)$, we obtain with the help of Lemma 3.8

$$\begin{aligned}
\mathsf{G}_k(r) &\leqslant \ell + (\ell-1)k^{l-1}(m+2)^{l-1} + c(m+1)^{k+1} + k^{l-1}(m+2)^{l-1} \\
&\leqslant (k(m+2)-1) + (k(m+2)-2)k^{l-1}(m+2)^{l-1} \\
&\qquad + c(m+1)^{k+1} + k^{l-1}(m+2)^{l-1} \\
&= (k(m+2)-1) + k^{l}(m+2)^{l} - k^{l-1}(m+2)^{l-1} + c(m+1)^{k+1} \\
&\leqslant c(m+1)^{k+1} + k^{l}(m+2)^{l} \ ,
\end{aligned}$$

where the last inequality follows from $l-1 \geqslant 1$.

4. Suppose $r = f(r_1, \ldots, r_\ell)$ and $(s_1 \cdots s_n) \blacktriangleright_k^l (r_1 \cdots r_\ell)$. Hence it follows that $\sum_j \mathsf{G}_k(r_j) < \sum_i \mathsf{G}_k(s_i) \leqslant m$. By assumption, we conclude

$$\mathsf{G}_k(f(r_1, \ldots, r_\ell)) < c(m+1)^{k+1} \leqslant c(m+1)^{k+1} + k^{l}(m+2)^{l} \ .$$

Finally, we obtain $\mathsf{G}_k(t) \leqslant c(m+1)^{k+1} + c(m+2)^{k} < c(m+2)^{k+1}$.

Step Case. Let $\operatorname{rank}(f) = p+1$. By induction hypothesis on p we can assume $F_{k,p}(m) \leqslant c(m+2)^{d_{k,p}}$. By side-induction hypothesis on m we conclude that $\mathsf{F}_{k,p+1}(m_1) \leqslant c(m_1+2)^{d_{k,p}}$ holds for all $m_1 < m$.

We set $d = d_{k,p}$, $d' = d_{k,p+1} = d^k + 1$ and furthermore $c' = c^{\sum_{i=0}^{k-2} d^i} k^{\sum_{i=0}^{k-1} d^i}$. Let $g^{(1)}(n) = n$, $g^{(l+1)}(n) = c[k(n+2)g^{(l)}(n)]^d$. We consider the following claim.

Claim. If $s = f(s_1, \ldots, s_n)$, $\sum_i \mathsf{G}_k(s_i) + n \leqslant m$ and $s >_k^l r$ then

$$\mathsf{G}_k(r) \leqslant g^{(l)}(m+2) \ .$$

Proof of Claim. By induction on l. Suppose $l = 1$, then $s >_k^1 r$ implies the existence of $i \in \{1, \ldots, n\}$ such that $s_i \gtrsim_k r$. Hence $\mathsf{G}_k(r) \leqslant \mathsf{G}_k(s_i) \leqslant m \leqslant g^{(1)}(m+2)$. Suppose $l > 1$, hence $s >_k^l r$ either implies that $s_i \gtrsim_k^l r$ for some $i \in \{1, \ldots, n\}$ or $r = g(r_1, \ldots, r_\ell)$ and $s >_k^{l-1} r_j$ for all $1 \leqslant j \leqslant \ell$. In the first case, the assertion follows as above. For the second case, we obtain from $F_{k,p}(m) \leqslant c(m+2)^d$ and induction hypothesis on l

$$\mathsf{G}_k(t) \leqslant c[\sum_i \mathsf{G}_k(r_i) + 2]^d \leqslant c[k(m+2)g^{(l-1)}(m+2)]^d = g^{(l)}(m+2) \ ,$$

where again we employ $\ell < k(m+2)$. A standard induction yields

$$g^{(l)}(m+2) \leqslant c^{\sum_{i=0}^{l-2} d^i} k^{\sum_{i=1}^{l-1} d^i} (m+2)^{\sum_{i=1}^{l} d^i}$$

as desired. □

We show that $s \blacktriangleright_k^l r$ implies $\mathsf{G}_k(r) \leqslant c'(m+1)^{d'} + c'(m+2)^{d^l}$ by induction on $s \blacktriangleright_k^l r$. We restrict our attention to the following two sub-cases.

1. Suppose $r = (r_1 \cdots r_\ell)$ and there exists j_0 such that $s \blacktriangleright_k^{l-1} r_{j_0}$ and $s \gtrdot_k^{l-1} r_i$ for all $i \in \{1, \dots, \ell\}$. By the claim and Lemma 3.8, we have

$$\begin{aligned}\sum_{i \neq j_0} \mathsf{G}_k(r_i) &\leqslant (\ell - 1) g^{(l-1)}(m+2) \\ &\leqslant k(m+2) c^{\sum_{i=0}^{l-3} d^i} k^{\sum_{i=1}^{l-2} d^i} m^{\sum_{i=1}^{l-1} d^i} \\ &\leqslant c'(m+2)^{\sum_{i=1}^{l-1} d^i} \\ &\leqslant c'(m+2)^{d^l - 1} .\end{aligned}$$

By induction hypothesis, we see $\mathsf{G}_k(r_{j_0}) \leqslant c'(m+1)^{d'} + c'(m+2)^{d^{l-1}}$. Hence

$$\begin{aligned}\mathsf{G}_k(r) &\leqslant c'(m+1)^{d'} + c'(m+2)^{d^{l-1}} + c'(m+2)^{d^l - 1} \\ &\leqslant c'(m+1)^{d'} + c'(m+2)^{d^l}\end{aligned}$$

2. Suppose $r = f(r_1, \dots, r_\ell)$ and furthermore $(s_1 \cdots s_n) \blacktriangleright_k^l (r_1 \cdots r_\ell)$. Hence $\sum_j \mathsf{G}_k(r_j) + \ell < \sum_i \mathsf{G}_k(s_i) + n \leqslant m$. By this assumption, we conclude that $\mathsf{G}_k(f(r_1, \dots, r_\ell)) \leqslant c'(m+1)^{d'}$ and thus $\mathsf{G}_k(r) \leqslant c'(m+1)^{d'} + c'(m+2)^{d^l}$.

Finally, we obtain $\mathsf{G}_k(t) \leqslant c'(m+1)^{d'} + c'(m+2)^{d^k} < c'(m+2)^{d'}$. $\square$

4 The Polynomial Path Order POP*

In this chapter we present the *polynomial path order* $>_{\mathsf{pop}*}$. Similar to the polynomial path order $\blacktriangleright$ on sequences, the idea of predicative recursion is employed to break the strength of MPO. Unlike $\blacktriangleright$, the schemes of predicative recursion are enforced explicitly. For this, we reflect the separation of safe and normal argument positions in the definition of *safe mappings* defined below. In the sequel we fix a finite and non-variadic signature $\mathcal{F}$, and denote by $\mathcal{R}$ a constructor TRS over the signature $\mathcal{F}$. Furthermore, we write $\mathcal{D}$ for the defined symbols, and $\mathcal{C}$ for the constructors of $\mathcal{R}$ respectively.

Definition 4.1. A *safe mapping* safe is a function $\mathsf{safe} : \mathcal{F} \to 2^{\mathbb{N}}$ that associates with every n-ary function symbol f the set of *safe argument positions.* If $f \in \mathcal{D}$ then $\mathsf{safe}(f) \subseteq \{1, \dots, n\}$, for $f \in \mathcal{C}$ we fix that all argument positions of f are safe, that is $\mathsf{safe}(f) = \{1, \dots, n\}$. The argument positions not included in $\mathsf{safe}(f)$ are called *normal* and denoted by $\mathsf{nrm}(f)$, i.e. $\mathsf{safe}(f) \uplus \mathsf{nrm}(f) = \{1, \dots, n\}$.

We always denote by safe a safe-mapping, and by nrm the normal part of safe. We extend the mapping safe to terms by $\mathsf{safe}(f(t_1, \dots, t_n)) = \{t_{i_1}, \dots, t_{i_p}\}$ where $\mathsf{safe}(f) = \{i_1, \dots, i_p\}$, and $\mathsf{safe}(t) = \varnothing$ if $t \in \mathcal{V}$. In a similar spirit, we define $\mathsf{nrm}(f(t_1, \dots, t_n)) = \{t_{j_1}, \dots, t_{j_q}\}$ with $\mathsf{nrm}(f) = \{j_1, \dots, j_q\}$ and $\mathsf{nrm}(t) = \varnothing$ otherwise. Furthermore, the polynomial path order relies on the distinction between function symbols that give rise to reductions and symbols that encode values. We need to be careful not to break those separations in the definition of equivalence. This is made precise in the following two definitions.

Definition 4.2. We say that a quasi precedence $\succsim$ is *admissible* for the polynomial path order if the following conditions are satisfied:

(i) all constructors are minimal in the precedence, that is if $f \succ g$ with $g \in \mathcal{D}$ then $f \in \mathcal{D}$, and

(ii) the equivalence part $\sim$ of $\succsim$ respects the separation of $\mathcal{F}$ into defined and constructor symbols, that is if $f \sim g$ then $f \in \mathcal{D}$ if and only if $g \in \mathcal{D}$.

In the remaining, we denote by $\succsim$ a quasi-precedence that satisfies conditions (i) and (ii) of Definition 4.2. We simply call $\succsim$ a precedence. Based on $\succsim$, we define equivalence on terms under a safe mapping safe.

Definition 4.3. We say that two terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ are *equivalent* with respect to $\succsim$ and safe, written as $s \overset{\mathsf{s}}{\sim} t$, if either

(i) $s = t$, or

(ii) $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_n)$, $f \sim g$, and there exists a permutation π such that $s_i \overset{\mathsf{s}}{\sim} t_{\pi(i)}$ and $i \in \mathsf{safe}(f)$ if and only if $\pi(i) \in \mathsf{safe}(g)$ holds for $i \in \{1, \ldots, n\}$.

Observe that $\overset{\mathsf{s}}{\sim} \subseteq \sim$ for the standard definition of equivalence $\sim$ on terms (cf. Definition 2.33). Similar to the polynomial path order $\blacktriangleright$ on sequences, the polynomial path order $>_{\mathsf{pop}*}$ is an extension of an auxiliary order, below denoted by $>_{\mathsf{pop}}$. We render both orders in the next two definitions.

Definition 4.4. We define the auxiliary order $>_{\mathsf{pop}}$ induced by the precedence $\succsim$ and safe mapping safe inductively by $s >^{\pi}_{\mathsf{pop}} t$ for $s = f(s_1, \ldots, s_n)$ if one of the following alternatives hold:

(1) $s_i \gtrsim_{\mathsf{pop}} t$ for some $i \in \{1, \ldots, n\}$, and if $f \in \mathcal{D}$ then $i \in \mathsf{nrm}(f)$, or

(2) $t = g(t_1, \ldots, t_m)$, $f \in \mathcal{D}$, $f \succ g$ and for all $j \in \{1, \ldots, m\}$, $s >^{\pi}_{\mathsf{pop}} t_j$.

Here $\gtrsim_{\mathsf{pop}}$ denotes $>_{\mathsf{pop}} \cup \overset{\mathsf{s}}{\sim}$.

Definition 4.5. We define the *polynomial path order* $>_{\mathsf{pop}*}$ induced by the precedence $\succsim$ and safe mapping safe inductively by $s >_{\mathsf{pop}*} t$ for $s = f(s_1, \ldots, s_n)$ if either $s >_{\mathsf{pop}} t$ or one of the following alternatives hold:

(1) $s_i \gtrsim_{\mathsf{pop}*} t$ for some $i \in \{1, \ldots, n\}$, or

(2) $t = g(t_1, \ldots, t_m)$, $f \in \mathcal{D}$, $f \succ g$ and
   - $s >_{\mathsf{pop}*} t_{j_0}$ for some $j_0 \in \mathsf{safe}(g)$, and
   - for all $j \neq j_0$ either $s >_{\mathsf{pop}} t_j$, or $s \rhd t_j$ and $j \in \mathsf{safe}(g)$, or

(3) $t = g(t_1, \ldots, t_m)$, $f \in \mathcal{D}$, $f \sim g$ and both $\mathsf{nrm}(s) >^{\mathsf{mul}}_{\mathsf{pop}*} \mathsf{nrm}(t)$ as well as $\mathsf{safe}(s) \gtrsim^{\mathsf{mul}}_{\mathsf{pop}*} \mathsf{safe}(t)$ holds.

Here $\gtrsim_{\mathsf{pop}*}$ refers to $>_{\mathsf{pop}*} \cup \overset{\mathsf{s}}{\sim}$. Observe that $\overset{\mathsf{s}}{\sim} \cdot >_{\mathsf{pop}*} \cdot \overset{\mathsf{s}}{\sim} \subseteq >_{\mathsf{pop}*}$ and hence $>^{\mathsf{mul}}_{\mathsf{pop}*}$ and $\gtrsim^{\mathsf{mul}}_{\mathsf{pop}*}$ denoting the strict and weak multiset extension of $\gtrsim_{\mathsf{pop}*}$ respectively are well-defined.

In clause (2) above, we allow $s \rhd t_j$ for some safe argument position $j \in \mathsf{safe}(g)$. This is a mainly technical point that strengthens the polynomial path order suitably for the completeness property shown below (notice that $\rhd \subseteq >_{\mathsf{pop}}$ does not hold in general). Furthermore, notice that due to the restricted definition of clause (3), $>_{\mathsf{pop}*}$ is not a reduction order. Although it is closed under substitutions, it lacks closure under contexts. Still, compatibility of $\mathcal{R}$ with $>_{\mathsf{pop}*}$ asserts termination of $\mathcal{R}$, as the polynomially path order is just a tamed version of MPO. Hence the following theorem is immediate.

Theorem 4.6. *Let $\mathcal{R}$ be a TRS compatibly with $>_{\mathsf{pop}*}$, i.e. $\mathcal{R} \subseteq >_{\mathsf{pop}*}$ holds. Then $\mathcal{R}$ is terminating.*

We demonstrate the application of the polynomial path order on three small examples below.

Example 4.7. Consider the well-known constructor TRS $\mathcal{R}_{\mathsf{mult}}$ defined by the following rewrite rules:

$$\begin{aligned} \mathsf{add}(0, y) &\to y & \mathsf{mult}(0, y) &\to 0 \\ \mathsf{add}(\mathsf{s}(x), y) &\to \mathsf{s}(\mathsf{add}(x, y)) & \mathsf{mult}(\mathsf{s}(x), y) &\to \mathsf{add}(y, \mathsf{mult}(x, y))\ . \end{aligned}$$

The above rewrite system encodes addition (add) and multiplication (mult) over tally numbers. It is easy to see that $\mathcal{R}_{\mathsf{mult}}$ is compatible with $>_{\mathsf{mpo}}$ as induced by the strict precedence $\mathsf{mult} \succ \mathsf{add} \succ \mathsf{s} \succ 0$. Furthermore, the recursive definitions of addition and multiplication as encoded in $\mathcal{R}_{\mathsf{mult}}$ obey the predicative recursion and composition schemes: declare the first argument of addition and both arguments of multiplication as normal, and declare the second argument position of addition as safe. Both functions perform recursion on a normal argument, and furthermore the recursively computed results $\mathsf{add}(x, y)$ and respectively $\mathsf{mult}(x, y)$ are substituted into safe argument positions. Based on this observation, we define $\mathsf{safe}(\mathsf{add}) = \{2\}$, $\mathsf{safe}(\mathsf{mult}) = \varnothing$ and $\mathsf{safe}(\mathsf{s}) = \{1\}$. Then it can be verified that $\mathcal{R}_{\mathsf{mult}}$ is compatible with $>_{\mathsf{pop*}}$ as induced by $\succ$ and safe. We demonstrate this on the last rule and show

$$\mathsf{mult}(\mathsf{s}(;x), y;) >_{\mathsf{pop*}} \mathsf{add}(y; \mathsf{mult}(x, y;))\ . \tag{i}$$

Above we employ the usual notation for the separation of normal (to the left) and safe arguments (to the right). As $\mathsf{mult} \succ \mathsf{add}$ in the precedence it suffices to verify $\mathsf{mult}(\mathsf{s}(;x), y;) >_{\mathsf{pop}} y$ and $\mathsf{mult}(\mathsf{s}(;x), y;) >_{\mathsf{pop*}} \mathsf{mult}(x, y;)$. The former follows directly by one application of clause (1) from $>_{\mathsf{pop}}$, and the latter follows from $[\mathsf{s}(;x), y] >^{\mathsf{mul}}_{\mathsf{pop*}} [x, y]$ and $[] \gtrsim^{\mathsf{mul}}_{\mathsf{pop*}} []$ where we employ clause (3) from the definition of $>_{\mathsf{pop*}}$. We conclude (i).

Notice how $>_{\mathsf{pop*}}$ naturally enforces the predicative recursion scheme: remember that the recursively computed result needs to be substituted into a safe argument position of the stepping function. To the contrary, in the example above assume that the recursively computed result $\mathsf{mult}(x, y;)$ is substituted into a normal argument position of add. As $>_{\mathsf{pop*}}$ can only be applied on a safe argument position, we need to certify $\mathsf{mult}(s(;x), y;) >_{\mathsf{pop}} \mathsf{mult}(x, y;)$ for that case. Clearly this is prohibited by definition of $>_{\mathsf{pop}}$. On the other hand, predicative recursion dictates that each recursively defined function performs recursion on a normal argument position. Again, this is asserted by the orientation with $>_{\mathsf{pop*}}$. To the contrary, suppose mult performs recursion on a safe argument y. Then the orientation of $\mathsf{mult}(x; s(;y)) >_{\mathsf{pop*}} \mathsf{mult}(x; y)$ fails as $[x] >^{\mathsf{mul}}_{\mathsf{pop*}} [x]$ cannot be shown.

Example 4.8. Consider the constructor TRS $\mathcal{R}_{\mathsf{exp}}$ defined as

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

This system is compatible with $>_{\mathsf{mpo}}$ induced by the precedence $\mathsf{exp} \succ \mathsf{double} \succ \mathsf{s} \succ 0$. It is not difficult to see that the runtime-complexity of $\mathcal{R}_{\mathsf{exp}}$ is exponential. Compatibility with a polynomial path order cannot be shown, due to the restrictions imposed by the safe mapping.

Notice that the encoded exponentiation function cannot be defined under the regime of predicative recursion. To see why the orientation fails, consider the last rewrite rule of $\mathcal{R}_{\mathsf{exp}}$, that is the rewrite rule defining the recursion step of exp. According to clause (2) from the definition of $>_{\mathsf{pop}*}$, the recursively computed result $\mathsf{exp}(x)$ needs to be put into a safe argument position of the stepping function double, i.e. $\mathsf{safe}(\mathsf{double}) = \{1\}$. However, in order to orient the rewrite rules defining double one needs to mark the argument position of the recursion argument as normal, that is $\mathsf{safe}(\mathsf{double}) = \varnothing$. A uniform safe mapping safe cannot be found.

Example 4.9. Consider the constructor TRS $\mathcal{R}_{\mathsf{bits}}$ taken from [24] and defined through the following six rewrite rules:

$$
\begin{aligned}
\mathsf{half}(0) &\to 0 & \mathsf{bits}(0) &\to 0 \\
\mathsf{half}(\mathsf{s}(0)) &\to 0 & \mathsf{bits}(\mathsf{s}(0)) &\to \mathsf{s}(0) \\
\mathsf{half}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{half}(x)) & \mathsf{bits}(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{s}(\mathsf{bits}(\mathsf{s}(\mathsf{half}(x)))) \ .
\end{aligned}
$$

Here $\mathsf{half}(x)$ computes $\lfloor \frac{x}{2} \rfloor$ and $\mathsf{bits}(x)$ computes $\lceil \log(x) \rceil$ where x is a natural number given in successor representation. Again, this system does not follow the predicative recursion schema, even thought the encoded functions are polytime computable (and the above rewrite system defines an algorithm with at most quadratic many evaluation steps).

Although the system is compatible with $>_{\mathsf{mpo}}$, again the inclusion $\mathcal{R}_{\mathsf{bits}} \subseteq >_{\mathsf{pop}*}$ does not hold for any order $>_{\mathsf{pop}*}$. By solely relying on the concept of predicative recursion, the polynomial path order cannot detect that the argument to bits in the critical rule $\mathsf{bits}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{bits}(\mathsf{s}(\mathsf{half}(x))))$ does not increase the recursive argument, and thus does not give rise to an exponential blowup.

The above observed restriction of predicative recursion is a well known problem, and treated extensively in Caseiro's PhD thesis [16]. Caseiro introduces various more general schemata that certifying polytime computability. Unfortunately, these are mostly of semantical nature and hard to automate. The above highlighted problem is also addressed by Hofmann in [33]. In the system proposed, recursion on recursively computed result is allowed, at the expense that all involved functions are *non-size increasing* and moreover, all expressions are typable under a *linear type system*. Both conditions present severe restrictions, for instance they prohibit the natural definition of multiplication. Even though the polynomial path order is not applicable to the system $\mathcal{R}_{\mathsf{bits}}$ directly, in Chapter 6 we demonstrate that by simple mechanic transformations the polynomial path order is indeed capable of handling $\mathcal{R}_{\mathsf{bits}}$. Transformations allow us to at least partially break the limitations of predicative recursion.

Below we present the Main Theorem from this chapter. It states that the polynomial path order induces polynomial bounds on the runtime-complexity.

Theorem 4.10. *Let $\mathcal{R}$ be a constructor TRS compatible with $>_{\mathsf{pop}*}$, i.e $\mathcal{R} \subseteq >_{\mathsf{pop}*}$ holds. There exists a polynomial $p \colon \mathbb{N} \to \mathbb{N}$ such that*

$$\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}(n) \leqslant p(n)$$

for all $n \in \mathbb{N}$. The polynomial p depends only on the cardinality of $\mathcal{F}$ and the sizes of the right-hand sides in $\mathcal{R}$.

We highlight that the above theorem is restricted to constructor TRSs. Although this is essential, it seems possible to slightly adapt the definition of POP* such that Theorem 4.10 can be reformulated for arbitrary TRSs. This is subject to future research. A proof of the theorem is delayed until the next chapter, where we show a more general result (cf. Theorem 5.34). Instead, we relate the polynomial path order to classical complexity analysis and establish an important completeness property (cf. also the technical report [8]). In particular, we give yet another characterization of the polytime computable functions. Following [13], we provide a semantics for completely defined orthogonal TRSs. For Σ an alphabet, let Σ^* denotes the set of words over Σ. Let $\ulcorner\cdot\urcorner\colon \Sigma^* \to \mathsf{Val}$ denote an encoding function that represents words over Σ as ground values. We call an encoding $\ulcorner\cdot\urcorner$ *reasonable* if it is bijective and there exists a constant c such that $|u| \leqslant |\ulcorner u \urcorner| \leqslant c \cdot |u|$ for every $u \in \Sigma^*$.

Definition 4.11. Let $\ulcorner\cdot\urcorner$ be an encoding function and let $\mathcal{R}$ denote a completely defined, orthogonal TRS. An n-ary function $f\colon (\Sigma^*)^n \to \Sigma^*$ is *computable* by $\mathcal{R}$ if there exists a defined function symbol f such that for all $w_1, \dots, w_n, v \in \Sigma^*$

$$\mathsf{f}(\ulcorner w_1 \urcorner, \dots, \ulcorner w_n \urcorner) \to^! \ulcorner v \urcorner \iff f(w_1, \dots, w_n) = v \; .$$

On the other hand we say that $\mathcal{R}$ *computes* f, if the function $f\colon (\Sigma^*)^n \to \Sigma^*$ is defined by the above equation.

In [11] Beckmann and Weiermann provide a *term rewriting characterization* R_B of Bellantoni and Cook's class of predicative recursive functions. A term rewriting characterization is a very powerful but quite simple concept: assume a recursion-theoretic characterization of a class of functions $\mathcal{C}$. From this it is often straight forward to define a schema of rewrite rules $\mathcal{R}_\mathcal{C}$ such that every function $f \in \mathcal{C}$ is computed by some restriction $\mathcal{R}_f \subseteq \mathcal{R}_\mathcal{C}$ and vice versa. In all non-pathological cases the TRS $\mathcal{R}_f$ is a terminating and confluent constructor TRS, and may be conceived as an algorithm computing the function f under the resource bounds admitted by $\mathcal{C}$. This way, theory developed in the context of rewriting is applicable for the analysis of the class $\mathcal{C}$ (cf. for instance [17], where elegant proofs of non-trivial closure properties of the primitive recursive functions are given, but cf. also [11]). We render the schema R_B for the predicative recursive functions in the upcoming definition.

Definition 4.12. For $k, l \in \mathbb{N}$ we define $\mathcal{B}^{k,l}$ as the least set of function symbols with k normal and l safe argument positions such that:

(i) $0 \in \mathcal{B}^{0,0}$, $\mathrm{S}_0, \mathrm{S}_1, \mathrm{P} \in \mathcal{B}^{0,1}$ and $\mathrm{C} \in \mathcal{B}^{0,3}$, and

(ii) $\mathrm{U}^{k,l}_r, \mathrm{O}^{k,l} \in \mathcal{B}^{k,l}$ if $k + l > 0$ and $r \in \{1, \dots, k + l\}$, and

(iii) $\mathrm{SUB}^{k,l}_{m,n}[f, \vec{g}, \vec{h}] \in \mathcal{B}^{k,l}$ for $f \in \mathcal{B}^{m,n}$, $\vec{g} \in \mathcal{B}^{k,0}$ and $\vec{h} \in \mathcal{B}^{k,l}$, and

(iv) $\mathrm{PREC}^{k,l}[g, h_0, h_1] \in \mathcal{B}^{k+1,l}$ for $g \in \mathcal{B}^{k,l}$ and $h_0, h_1 \in \mathcal{B}^{k+1,l+1}$.

The *predicative signature* $\mathcal{B}$ is given by $\bigcup_{k,l\in\mathbb{N}} \mathcal{B}^{k,l}$. Below, for brevity we abbreviate $x_1,\dots,x_k$ by $\vec{x}$ and $y_1,\dots,y_l$ by $\vec{y}$. Furthermore, we write the sequences $g_1(\vec{x};),\dots,g_m(\vec{x};)$ as $\vec{g}(\vec{x};)$ and $h_1(\vec{x};\vec{y}),\dots,h_n(\vec{x};\vec{y})$ as $\vec{h}(\vec{x};\vec{y})$. The schema of TRSs R_B over the signature $\mathcal{B}$ is given below, where indices k,l,m,n range over naturals numbers and $i \in \{0,1\}$.

$$\begin{aligned}
\mathrm{O}^{k,l}(\vec{x};\vec{y}) &\to 0 \\
\mathrm{U}^{k,l}_r(\vec{x};\vec{y}) &\to x_r && \text{for } 1 \leqslant r \leqslant k \\
\mathrm{U}^{k,l}_r(\vec{x};\vec{y}) &\to y_{r-k} && \text{for } k < r \leqslant l+k \\
\mathrm{P}(;0) &\to 0 \\
\mathrm{P}(;\mathrm{S}_i(;x)) &\to x \\
\mathrm{C}(;0,y_1,y_2) &\to y_1 \\
\mathrm{C}(;\mathrm{S}_i(;x),y_1,y_2) &\to y_2 \\
\mathrm{SUB}^{k,l}_{m,n}[f,\vec{g},\vec{h}](\vec{x};\vec{y}) &\to f(\vec{g}(\vec{x};);\vec{h}(\vec{x};\vec{y})) \\
\mathrm{PREC}^{k+1,l}[g,h_0,h_1](0,\vec{x};\vec{y}) &\to g(\vec{x};\vec{y}) \\
\mathrm{PREC}^{k+1,l}[g,h_0,h_1](\mathrm{S}_i(;z),\vec{x};\vec{y}) &\to h_i(z,\vec{x};\vec{y},\mathrm{PREC}^{k+1,l}[g,h_0,h_1](z,\vec{x};\vec{y}))\ .
\end{aligned}$$

The following proposition follows directly from the definition of R_B.

Proposition 4.13 ([11])**.** *Let $f \in$ **FP**. There exists a finite restriction $\mathcal{R}_f \subseteq R_B$ such that $\mathcal{R}_f$ computes f.*

Clearly R_f is in any case a terminating and confluent constructor TRS. Termination can be verified by recursive path orders, and from orthogonality, confluence is asserted. In particular, this implies that the computation of f through the rewrite system R_f is independent on the employed strategy, a fact we use in a moment.

Remark. It is important to remark that the schema of TRSs R_B rendered above is dubbed *unfeasible* in [11]. This is due to the case that this schema admits an exponential lower bound on the derivation length and therefore is not (directly) suitable as a term-rewriting characterization of the predicative recursive functions. This exponential lower-bound is only correct if we consider full rewriting. Together with an innermost strategy, or more generally the assertions that safe arguments are evaluated in a *call by value* fashion, the system becomes *feasible*, cf. [11].

Definition 4.14 ([40])**.** A *simple* signature $\mathcal{F}$ is a sorted signature such that each sort has a finite *rank* r in the following sense: the sort s has rank r if for every constructor c with $\mathrm{arity}(c) = (s_1,\dots,s_n)$ and $\mathrm{type}(c) = s$, the rank of each sort s_i is less than the rank of s, except for at most one sort which can be of rank r.

Simple signatures allow the definition of enumerated datatypes and inductive datatypes like words and lists but prohibit for instance the definition of tree structures. In particular the scheme R_B from Definition 4.12 is based on a simple signature.

We conclude this chapter with a final theorem. Following [8], we present an alternative characterization of the functions computable in **FP**.

Theorem 4.15 ([8]). *Each finite, orthogonal and constructor TRS based on a simple signature that is compatible with $>_{\mathsf{pop}*}$ is computable in polynomial time and vice versa each polynomial computable function is computable by a finite, orthogonal and constructor TRS compatible with $>_{\mathsf{pop}*}$ that is based on a simple signature.*

Proof. We consider the first half of the assertion. Suppose $\mathcal{R}$ denotes a finite, orthogonal and constructor TRS compatible with an instance $>_{\mathsf{pop}*}$. We single out one of the defined symbols $\mathsf{f} \in \mathcal{D}$ and consider the corresponding function $f\colon (\Sigma^*)^n \to \Sigma^*$ computed by $\mathcal{R}$. From the inclusion $\mathcal{R} \subseteq >_{\mathsf{pop}*}$ we see by Theorem 4.6 that $\mathcal{R}$ is terminating. Moreover, from orthogonality (and hence confluence) of $\mathcal{R}$, normal forms are unique and thus the function f is well-defined. Suppose $\mathsf{f}(\ulcorner w_1 \urcorner, \dots, \ulcorner w_n \urcorner) \to^!_{\mathcal{R}} \ulcorner v \urcorner$ for words $w_1, \dots, w_n, v$. In particular, from confluence we see that

$$\mathsf{f}(\ulcorner w_1 \urcorner, \dots, \ulcorner w_n \urcorner) \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_1 \xrightarrow{\mathsf{i}}_{\mathcal{R}} \cdots \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_\ell = \ulcorner v \urcorner \ .$$

It is folklore, that there exists a polytime algorithm performing one rewrite step. From the inclusion $\mathcal{R} \subseteq >_{\mathsf{pop}*}$, by Theorem 4.10 we see that ℓ is bounded by a polynomial in the sum of the sizes of $\ulcorner w_1 \urcorner, \dots, \ulcorner w_n \urcorner$. As $\ulcorner \cdot \urcorner$ is reasonable, we conclude the existence of polynomial p such that $\ell \leqslant p(\sum_i |w_i|)$. To conclude the existence of a polytime algorithm for f it suffices to bound the size of terms t_i for $1 \leqslant i \leqslant \ell$ polynomially in $\sum_i |w_i|$. This is a consequence of the following claim, where again we employ that the encoding $\ulcorner \cdot \urcorner$ is reasonable.

Claim. Let $\mathcal{R}$ be a constructor TRS based on a simple signature that is compatible with $>_{\mathsf{pop}*}$. Let $t \in \mathcal{T}_{\mathsf{b}}$. There exists a polynomial q such that when $t \xrightarrow{\mathsf{i}}^*_{\mathcal{R}} s$ then $|s| \leqslant q(|t|)$.

Proof of Claim. We establish this claim in Lemma 5.37. □

Now assume on the other hand that $f \in \mathbf{FP}$ and let $\mathcal{R}_f$ be the finite rewrite system computing f as given from Proposition 4.13. The assertion follows, if we define a suitable instance $>_{\mathsf{pop}*}$ such that $\mathcal{R}_f \subseteq >_{\mathsf{pop}*}$. As $\mathcal{R}_f$ is based on the predicative signature (cf. Definition 4.12) the definition of the safe mappings is immediate. It remains to define a suitable (finite) precedence. We define a mapping lh from the signature of $\mathcal{B}$ into the natural numbers as follows. Let $\mathrm{lh}(g)$, $g \in \mathcal{B}$ be defined as follows: $\mathrm{lh}(g) = 1$ for $g \in \{0, \mathrm{S}_0, \mathrm{S}_1, \mathrm{P}, \mathrm{U}^{k,l}_r, \mathrm{O}^{k,l}, \mathrm{C}\}$ with $k, l \in \mathbb{N}$. Furthermore, define

$$\begin{aligned} \mathrm{lh}(\mathrm{SUB}^{k,l}_{m,n}[f, \vec{g}, \vec{h}]) &= 1 + \mathrm{lh}(f) + \Sigma^m_{i=1} \mathrm{lh}(g_i) + \Sigma^n_{i=1} \mathrm{lh}(h_i), \text{ and} \\ \mathrm{lh}(\mathrm{PREC}^{k,l}[g, h_0, h_1]) &= 1 + \mathrm{lh}(g) + \mathrm{lh}(h_0) + \mathrm{lh}(h_1) \ . \end{aligned}$$

We define the strict precedence $\succ$ such that for $f, g \in \mathcal{B}$, if $\mathrm{lh}(f) \succ \mathrm{lh}(g)$ and f, g appear in $\mathcal{R}_f$ then $f \succ g$. As $\mathcal{R}_f$ is finite, it is easy to see that $\succ$ is finite. Moreover, the only constructor symbols $\mathrm{S}_0, \mathrm{S}_1$ and 0 are minimal in the precedence $\succ$. Thus $\succ$ is admissible (cf. Definition 4.2). This concludes the definition of $>_{\mathsf{pop*}}$.

Unfortunately $\mathcal{R}_f$ is in general not compatible with $>_{\mathsf{pop*}}$ as we cannot orient the rules governing predicative composition, that is the rules defining $\mathrm{SUB}^{k,l}_{m,n}[f, \vec{g}, \vec{h}]$. In particular, the orientation fails as $\mathrm{SUB}^{k,l}_{m,n}[f, \vec{g}, \vec{h}](\vec{x}; \vec{y}) >_{\mathsf{pop}} h_i(\vec{x}; \vec{y})$ does not hold for any i. However, we can transform $\mathcal{R}_f$ into a finite, orthogonal, constructor TRS $\mathcal{R}'_f$ by rewriting the rule for predicative composition to

$$\begin{aligned}
\mathrm{SUB}^{k,l}_{m,n}[f, \vec{g}, \vec{h}](\vec{x}; \vec{y}) &\to f_1(\vec{g}(\vec{x};), \vec{x}; h_1(\vec{x}; \vec{y}), \vec{y}) \\
f_1(\vec{u}, \vec{x}; v_1, \vec{y}) &\to f_2(\vec{u}, \vec{x}; v_1, h_2(\vec{x}; \vec{y}), \vec{y}) \\
&\vdots \\
f_{n-1}(\vec{u}, \vec{x}; \vec{v}, \vec{y}) &\to f(\vec{u}; \vec{v}, h_n(\vec{x}; \vec{y})) \ .
\end{aligned}$$

Here $\vec{u} = u_1, \ldots, u_m$, $\vec{v} = v_1, \ldots, u_{n-1}$ are variables and $f_1, \ldots, f_{n-1}$ are auxiliary function symbols for which we adapt the precedence such that $\mathrm{SUB}^{k,l}_{m,n} \succ f_1 \succ \cdots \succ f_{n-1} \succ f$. It is easy to see that the modified system $\mathcal{R}'_f$ simulates $\mathcal{R}_f$, that is $\mathcal{R}'_f$ computes the same functions as $\mathcal{R}_f$. Moreover $\mathcal{R}'_f \subseteq >_{\mathsf{pop*}}$ can be verified, and we conclude the claim. $\square$

5 The Polynomial Path Order and Relative Rewriting

In this chapter, we introduce the polynomial path order in the context of *relative rewriting*. We proof the Main Theorem of Chapter 4 in this generalized setting. In turn, this provides all technical details necessary for Chapter 6, where we apply the polynomial path order in the dependency pair setting for complexity analysis as proposed in [29, 30].

Relative rewriting is a notion that generalizes rewriting. Below, we briefly recall basic concepts. For further information, relative rewriting has been extensively studied in [25], and also [45, 52] provide some resources. Let $\mathcal{R}$ and $\mathcal{S}$ be two rewrite systems over the same signature. The *relative rewrite relation* $\to_{\mathcal{R}/\mathcal{S}}$ is defined by $\to_{\mathcal{R}/\mathcal{S}} = \to_{\mathcal{S}}^{*} \cdot \to_{\mathcal{R}} \cdot \to_{\mathcal{S}}^{*}$, that is, $\to_{\mathcal{R}/\mathcal{S}}$ corresponds to $\to_{\mathcal{R}\cup\mathcal{S}}^{+}$ with exactly one step due to $\mathcal{R}$. In a similar spirit, we define the *relative innermost rewrite relation* $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$ as $\xrightarrow{\mathsf{i}}_{\mathcal{R}\cup\mathcal{S}}^{+}$ with exactly one innermost step due to $\mathcal{R}$. Formally, we introduce $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$ in terms of the *generalized restricted rewrite relation* $\xrightarrow{\mathcal{Q}}_{\mathcal{R}}$ (cf. [53]).

Definition 5.1. Let $\mathcal{R}$ and $\mathcal{Q}$ be two TRSs. We define the *generalized restricted rewrite relation* $\xrightarrow{\mathcal{Q}}_{\mathcal{R}}$ such that $s \xrightarrow{\mathcal{Q}}_{\mathcal{R}} t$ if and only if there exists a one-hole context C, substitution σ and rule $f(\ell_1, \ldots, \ell_n) \to r \in \mathcal{R}$ such that $s = C[f(\ell_1\sigma, \ldots, \ell_n\sigma)]$, $t = C[r\sigma]$ and $\ell_i\sigma \in \mathsf{NF}(\mathcal{Q})$ for all $i \in \{1, \ldots, n\}$.

Generalized restricted rewriting generalizes both full and innermost rewriting: we have $\to_{\mathcal{R}} = \xrightarrow{\varnothing}_{\mathcal{R}}$ and moreover $\xrightarrow{\mathsf{i}}_{\mathcal{R}} = \xrightarrow{\mathcal{R}}_{\mathcal{R}}$. Observe that whenever $\mathsf{NF}(\mathcal{Q}) \subseteq \mathsf{NF}(\mathcal{P})$ for two TRSs $\mathcal{Q}$ and $\mathcal{P}$, it follows that $\xrightarrow{\mathcal{Q}}_{\mathcal{R}} \subseteq \xrightarrow{\mathcal{P}}_{\mathcal{R}}$ holds. The notion of generalized restricted rewriting allows us to concisely define relative rewriting with respect to an innermost reduction strategy:

Definition 5.2. Let $\mathcal{R}$ and $\mathcal{S}$ be two TRSs over the same signature. We define the *relative innermost rewrite relation* $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$ by

$$\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} = \xrightarrow{\mathcal{R}\,\cup\,\mathcal{S}}{}_{\mathcal{S}}^{*} \cdot \xrightarrow{\mathcal{R}\,\cup\,\mathcal{S}}{}_{\mathcal{R}} \cdot \xrightarrow{\mathcal{R}\,\cup\,\mathcal{S}}{}_{\mathcal{S}}^{*} .$$

A common technique for the termination analysis of $\to_{\mathcal{R}/\mathcal{S}}$ is the use of a pair of orderings $(\succeq, \succ)$. Here $\succeq$ is a rewrite preorder and $\succ$ a well-founded proper and *compatible order*, that is, the inclusion $\succeq \cdot \succ \cdot \succeq \; \subseteq \; \succ$ is satisfied. When $\mathcal{S} \subseteq \; \succeq$ and $\mathcal{R} \subseteq \; \succ$ hold, then each relative steps $\to_{\mathcal{R}/\mathcal{S}}$ translates to a descent with respect to $\succ$. As $\succ$ is well-founded, infinite reductions cannot exist. Since $>_{\mathsf{pop*}}$ is compatible with $\gtrsim_{\mathsf{pop*}}$, the above observation suggests that a pair of orders $(\gtrsim_{\mathsf{pop*}}, >_{\mathsf{pop*}})$ can be employed for the complexity analysis of $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$. Below we show that this is indeed the case. In particular, we show that the inclusions $\mathcal{R} \subseteq \; >_{\mathsf{pop*}}^{\pi}$ and $\mathcal{S} \subseteq \; \gtrsim_{\mathsf{pop*}}^{\pi}$ certify a polynomial bound in n

on the runtime-complexity $\mathrm{rc}(n, \mathcal{T}_{\mathsf{b}}, \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}})$ provided that $\mathcal{R} \cup \mathcal{S}$ is a constructor TRS. Since $\xrightarrow{\mathsf{i}}_{\mathcal{R}} = \xrightarrow{\mathsf{i}}_{\mathcal{R}/\varnothing}$, this simultaneously establishes Theorem 4.10.

When we consider the polynomial path order in the dependency pair method adapted for complexity analysis [29], the relation obtained from $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$ by restricting $\mathcal{R}$ steps to top-steps is of particular interest.

Definition 5.3. Let $\mathcal{R}$ and $\mathcal{S}$ be two TRSs over the same signature. We define

$$\xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{R}/\mathcal{S}} = \xrightarrow{\mathcal{R}\cup\mathcal{S}}{}^{*}_{\mathcal{S}} \cdot \xrightarrow{\mathcal{R}\cup\mathcal{S}}{}^{\varepsilon}_{\mathcal{R}} \cdot \xrightarrow{\mathcal{R}\cup\mathcal{S}}{}^{*}_{\mathcal{S}}$$

and call a step $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$ a relative (innermost) top-step.

The relation $\xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{R}/\mathcal{S}}$ gives rise to another well known transformation technique, the so called *argument filtering transformation* [37]. It allows one to remove arguments of function symbols, or to replace a term by one of its subterms.

Definition 5.4. An *argument filtering* for a signature $\mathcal{F}$ is a mapping π that associates with every n-ary function symbol $f \in \mathcal{F}$ either an argument position $i \in \{1, \dots, n\}$ or a 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:

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

We write $\pi(\mathcal{R})$ for the system $\{\pi(\ell) \to \pi(r) \mid \ell \to r \in \mathcal{R}\}$.

We define the *polynomial path ordering* $>^{\pi}_{\mathsf{pop}*}$ *with argument filterings* such that $s >^{\pi}_{\mathsf{pop}*} t$ if and only if $\pi(s) >_{\mathsf{pop}*} \pi(t)$. Below we show that a pair of orders $(\gtrsim^{\pi}_{\mathsf{pop}*}, >^{\pi}_{\mathsf{pop}*})$ can be employed for the complexity analysis of $\xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{R}/\mathcal{S}}$. This result will be used in Chapter 6, where we show that polynomial path orders with argument filterings can be used to form a *safe reduction pair* in the dependency pair setting from [29].

In the technical report [8] we presented a complete proof of Theorem 4.10, the main result from [7]. Ideally, we would like to follow [8] below, and then extend this result to the polynomial path order $>^{\pi}_{\mathsf{pop}*}$ with argument filterings. According to the following proposition, it is easy to see that the derivation length of $\to^{\varepsilon}_{\mathcal{R}/\mathcal{S}}$ can be estimated in terms of $\to^{\varepsilon}_{\pi(\mathcal{R})/\pi(\mathcal{S})}$.

Proposition 5.5. *Let $\mathcal{R}$ be a TRS and π an argument filtering. Then*

$$s \to^{\varepsilon}_{\mathcal{R}} t \Longrightarrow \pi(s) \to^{\varepsilon}_{\pi(\mathcal{R})} \pi(t) \textit{ and } s \to_{\mathcal{R}} t \Longrightarrow \pi(s) \to^{=}_{\pi(\mathcal{R})} \pi(t)\ .$$

Observe that the inclusion $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop}*}$ just amounts to $\pi(\mathcal{R}) \subseteq >_{\mathsf{pop}*}$, and from the above proposition together with the application of Theorem 4.10 it might be tempting to think that a complexity-certificate provided by $\pi(\mathcal{R}) \subseteq >_{\mathsf{pop}*}$ is transferable to $\mathcal{R}$. Unfortunately, this cannot be easily achieved, due to the following two problems:

(i) For a term $t \in \mathcal{T}_\mathsf{b}$, the transformed term $\pi(t)$ need not be constructor-based with respect to defined symbols and constructors as induced by $\pi(\mathcal{R})$ and $\pi(\mathcal{S})$ respectively.

(ii) Although a simulation of relative top-steps in the sense of Proposition 5.5 works for full rewriting, it fails for innermost steps.

We illustrate those problems in the following example.

Example 5.6. Let $\mathcal{D} = \{\mathsf{f}, \mathsf{h}\}$ and $\mathcal{C} = \{\mathsf{s}, 0\}$. Let the TRS $(\mathcal{R}, \mathcal{D} \uplus \mathcal{C})$ be defined by the rules

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

Define the argument filtering π such that $\pi(\mathsf{h}) = 1$ and $\pi(\mathsf{f}) = \pi(\mathsf{s}) = [1]$. Then $\pi(\mathcal{R})$ is given by

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

The term $\mathsf{f}(\mathsf{s}^n(0))$ is not constructor-based with respect to defined symbols $\mathcal{D}^\pi = \{\mathsf{f}, \mathsf{s}\}$ and constructors $\mathcal{C}^\pi = \{0\}$ as induced by $\pi(\mathcal{R})$. A complexity certificate for $\mathrm{rc}(n, \mathcal{T}_\mathsf{b}(\mathcal{D}^\pi \cup \mathcal{C}^\pi, \mathcal{V}), \to_{\pi(\mathcal{R})})$ does not necessarily translate to a certificate on $\mathrm{rc}(n, \mathcal{T}_\mathsf{b}(\mathcal{D} \cup \mathcal{C}, \mathcal{V}), \to_\mathcal{R})$. Notice that the above observation holds even for full rewriting. Next, observe that the step $\mathsf{f}(\mathsf{s}(0)) \xrightarrow{\mathsf{i}}_\mathcal{R} 0$ cannot be simulated by $\pi(\mathcal{R})$:

$$\pi(\mathsf{f}(\mathsf{s}(0))) = \mathsf{f}(\mathsf{s}(0)) \xrightarrow{\mathsf{i}}_{\pi(\mathcal{R})} \mathsf{f}(0) \neq \pi(0) \ .$$

Here, the rule $\mathsf{f}(\mathsf{s}(x)) \to x$ cannot be applied as $\mathsf{s}(0) \in \mathsf{NF}(\pi(\mathcal{R}))$.

However, both problems are easily lifted. Notably, in the extension $>^\pi_{\mathsf{pop}*}$ of $>_{\mathsf{pop}*}$ to argument filterings, we keep the separation of $\mathcal{F}$ into defined symbols $\mathcal{D}$ and constructors $\mathcal{C}$ as induced by the untransformed rewrite system. Thus the set of constructor-based terms $\mathcal{T}_\mathsf{b}$ stays stable. This addresses problem (i) from above. Reasoning via the inclusion $\pi(\mathcal{R}) \subseteq >_{\mathsf{pop}*}$ roughly amounts to the analysis of $\xrightarrow{\mathsf{i}}_{\pi(\mathcal{R})} \cap >_{\mathsf{pop}*}$. To the contrary, we analyze the relation $\xrightarrow{\mathsf{i}}_\mathcal{R} \cap >^\pi_{\mathsf{pop}*}$ and consequently $\xrightarrow{\mathsf{i}}_\mathcal{R} \cap >_{\mathsf{pop}*}$ when π denotes the identity on terms. This addresses problem (ii).

Let $s = f(s_1, \ldots, s_n)$ and π be an argument filtering such that $\pi(f) = [i_1, \ldots, i_p]$. In the following, we write $\mathsf{safe}_\pi(t)$ to denotes the multiset of safe arguments positions not erased, i.e. $\mathsf{safe}_\pi(t) = \{s_i \mid i \in \mathsf{safe}(f) \cap \pi(f)\}$. In a similar spirit, we use $\mathsf{nrm}_\pi(t)$ to denote the normal and not erased arguments of t. For sake of clarity, we unfold the definition of $>^\pi_{\mathsf{pop}*}$ in the following three definitions.

Definition 5.7. We define the equivalence relation $s \overset{\mathsf{s}}{\sim}_\pi t$ under π inductively, if one of the following alternatives holds:

(1) $s = t$, or

(2) $s = f(u_1, \ldots, u_n)$ and $t = x \in \mathcal{V}$, or $s = x \in \mathcal{V}$ and $t = f(u_1, \ldots, u_n)$, $\pi(f) = i$ and $u_i \overset{\mathsf{s}}{\sim}_\pi x$, or

(3) $s = f(s_1, \dots, s_n)$, $t = g(t_1, \dots, t_m)$, and
   (a) $\pi(f) = i$ and $s_i \overset{\mathsf{s}}{\sim}_\pi t$, or
   (b) $\pi(f) = [i_1, \dots, i_p]$, $\pi(g) = j$ and $s \overset{\mathsf{s}}{\sim}_\pi t_j$, or
   (c) $\pi(f) = [i_1, \dots, i_p]$, $\pi(g) = [j_1, \dots, j_q]$, $f \sim g$ and the multisets $\mathsf{nrm}_\pi(s)$ and $\mathsf{nrm}_\pi(t)$, as well as $\mathsf{safe}_\pi(s)$ and $\mathsf{safe}_\pi(t)$ are equivalent under $\overset{\mathsf{s}}{\sim}_\pi$.

Definition 5.8. We define the auxiliary order $>^\pi_{\mathsf{pop}}$ induced by the precedence $\succsim$, safe mapping safe and argument filtering π inductively by $s >^\pi_{\mathsf{pop}} t$ for $s = f(s_1, \dots, s_n)$ if one of the following alternatives holds:

(1) $\pi(f) = i$ and $s_i >^\pi_{\mathsf{pop}} t$, or

(2) $\pi(f) = [i_1, \dots, i_p]$ and for some $i \in \pi(f)$, $s_i \gtrsim^\pi_{\mathsf{pop}} t$ and if $f \in \mathcal{D}$ then $i \in \mathsf{nrm}_\pi(f)$, or

(3) $\pi(f) = [i_1, \dots, i_p]$, $t = g(t_1, \dots, t_m)$ and either
   (a) $\pi(g) = j$ and $s >^\pi_{\mathsf{pop}} t_j$, or
   (b) $\pi(g) = [j_1, \dots, j_q]$, $f \in \mathcal{D}$, $f \succ g$ and for all $j \in \pi(g)$, $s >^\pi_{\mathsf{pop}} t_j$.

Here $\gtrsim^\pi_{\mathsf{pop}}$ denotes $>^\pi_{\mathsf{pop}} \cup \overset{\mathsf{s}}{\sim}_\pi$.

Definition 5.9. We define the *polynomial path order* $>^\pi_{\mathsf{pop}*}$ *with argument filterings* induced by the precedence $\succsim$, safe mapping safe and argument filtering π inductively by $s >^\pi_{\mathsf{pop}*} t$ for $s = f(s_1, \dots, s_n)$ if either $s >^\pi_{\mathsf{pop}} t$ or one of the following alternatives holds:

(1) $\pi(f) = i$ and $s_i >^\pi_{\mathsf{pop}*} t$, or

(2) $\pi(f) = [i_1, \dots, i_p]$ and for some $i \in \pi(f)$, $s_i \gtrsim^\pi_{\mathsf{pop}*} t$, or

(3) $\pi(f) = [i_1, \dots, i_p]$, $t = g(t_1, \dots, t_m)$ and either
   (a) $\pi(g) = j$ and $s >^\pi_{\mathsf{pop}*} t_j$, or
   (b) $\pi(g) = [j_1, \dots, j_q]$, $f \in \mathcal{D}$, $f \succ g$ and
      – $s >^\pi_{\mathsf{pop}*} t_{j_0}$ for some $j_0 \in \mathsf{safe}(g) \cap \pi(g)$, and
      – for all $j \neq j_0$ with $j \in \mathsf{safe}(g) \cap \pi(g)$, either $s >^\pi_{\mathsf{pop}} t_j$, or $\pi(s) \rhd \pi(t_j)$ and $j \in \mathsf{safe}(g)$, or
   (c) $\pi(g) = [j_1, \dots, j_q]$, $f \in \mathcal{D}$, $f \sim g$ and $\mathsf{nrm}_\pi(s) >^{\pi,\mathsf{mul}}_{\mathsf{pop}*} \mathsf{nrm}_\pi(t)$ as well as $\mathsf{safe}_\pi(s) \gtrsim^{\pi,\mathsf{mul}}_{\mathsf{pop}*} \mathsf{safe}_\pi(t)$ hold.

Above we write $\gtrsim^\pi_{\mathsf{pop}*}$ for $>^\pi_{\mathsf{pop}*} \cup \overset{\mathsf{s}}{\sim}_\pi$. Observe that $\overset{\mathsf{s}}{\sim}_\pi \cdot >^\pi_{\mathsf{pop}*} \cdot \overset{\mathsf{s}}{\sim}_\pi \subseteq >^\pi_{\mathsf{pop}*}$ and hence $>^{\pi,\mathsf{mul}}_{\mathsf{pop}*}$ and $\gtrsim^{\pi,\mathsf{mul}}_{\mathsf{pop}*}$, denoting the strict and weak multiset extension of $\gtrsim^\pi_{\mathsf{pop}*}$ respectively, are well-defined.

Definition 5.10. We write id for the unique argument filtering that maps every term to itself, i.e. $\mathsf{id}(s) = s$ for all terms s. Formulated otherwise, $\mathsf{id}(f) = [1, \dots, n]$ for every function symbol $f \in \mathcal{F}$ of arity n.

The following Lemma attests the intuition behind $>^\pi_{\mathsf{pop}*}$.

Lemma 5.11. *For $s, t \in \mathcal{T}$, if $s >^{\pi}_{\mathsf{pop}*} t$ then $\pi(s) >_{\mathsf{pop}*} \pi(t)$ and vice versa, $\pi(s) >_{\mathsf{pop}*} \pi(t)$ implies $s >^{\pi}_{\mathsf{pop}*} t$. In particular, $>_{\mathsf{pop}*} = >^{\mathsf{id}}_{\mathsf{pop}*}$. Furthermore, $s \overset{s}{\sim}_{\pi} t$ if and only if $\pi(s) \overset{s}{\sim} \pi(p)$.*

Proof. The proof of the lemma follows by a standard induction. □

We start with some simple observations.

Lemma 5.12. *Let $s, t \in \mathcal{T}$. For $>^{\pi}_{\mathsf{pop}*}$ it holds that:*

(i) $>^{\pi}_{\mathsf{pop}*}$ is closed under substitution, that is, if $s >^{\pi}_{\mathsf{pop}*} t$ then $s\sigma >^{\pi}_{\mathsf{pop}*} t\sigma$ for any substitution σ, and

(ii) if $s >^{\pi}_{\mathsf{pop}*} f(t_1, \ldots, t_n)$ then $s >^{\pi}_{\mathsf{pop}*} t_i$ for all $i \in \pi(f)$ or $\pi(f) = i$, and

(iii) for $\pi(s) \in \mathsf{Val}$, if $s >^{\pi}_{\mathsf{pop}*} t$ then for some position $p \neq \varepsilon$, $\pi(s)|_p \overset{s}{\sim} \pi(t)$.

Proof. All three statements can be shown by a straight forward induction on the definition of $>^{\pi}_{\mathsf{pop}*}$ or $\overset{\mathsf{s}}{\sim}$ respectively. □

Observe that Lemma 5.12 implies that from $\pi(s) \in \mathsf{Val}$ and $s >^{\pi}_{\mathsf{pop}*} t$, $\pi(t) \in \mathsf{Val}$ follows. Moreover, the argument can be lifted to substitutions from variables into values. Precisely, from $\pi(s\sigma) \in \mathsf{Val}$ and $s >^{\pi}_{\mathsf{pop}*} t$, $\pi(t\sigma) \in \mathsf{Val}$ follows. This observation is crucial for the reasoning carried out below, and will be employed several times.

In order to certify that the polynomial path order induces polynomial bounds on the innermost runtime-complexity, we embed innermost rewrite steps into a finite approximation $\blacktriangleright_k$ of the path order for **FP**. We have seen in Chapter 3 that for a suitable set of starting terms the length of $\blacktriangleright_k$-descents is controlled by a polynomial in the sizes of terms. By the embedding, this polynomial bound translates to a polynomial bound on innermost rewrite steps. The embedding becomes possible, if we represent the size of terms as well as the information on normal and safe arguments underlying the definition of $>_{\mathsf{pop}*}$ explicitly. For that, we interpret the signature $\mathcal{F}$ into the *normalized signature*, defined as follows:

$$\mathcal{F}^{\mathsf{n}} = \{f^{\mathsf{n}} \mid f \in \mathcal{F}, \mathsf{nrm}\, f = \{i_1, \ldots, i_p\}, \text{ and} \operatorname{arity}(f^{\mathsf{n}}) = p\} .$$

We represent the size of a term t as the unary representation of the Buchholz-norm $\|t\|$ of t. For that, we introduce a fresh nullary function symbol s that is minimal in the precedence $\gtrsim$ on $\mathcal{F}$. We define the mapping $\mathsf{BN} : \mathbb{N} \to \mathcal{S}\mathrm{eq}(\{\mathsf{s}\}, \varnothing)$ as $\mathsf{BN}(t) = (\mathsf{s} \ \cdots \ \mathsf{s})$ with $\|t\|$ occurrences of the constant s. Below, we also write $\gtrsim$ for the quasi-precedence $\{(f^{\mathsf{n}}, g^{\mathsf{n}}) \mid f \gtrsim g \text{ and } f, g \in \mathcal{F}\}$ isomorphic to $\gtrsim$ on $\mathcal{F}$. Moreover, we set $f^{\mathsf{n}} > \circ$ for all $f \in \mathcal{F}$ and the variadic symbol $\circ$.

Definition 5.13. A *predicative interpretation* is a pair (S, N) of mappings

$$\mathsf{S}, \mathsf{N} : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{S}\mathrm{eq}(\mathcal{F}^{\mathsf{n}} \cup \{\mathsf{s}\}, \mathcal{V})$$

defined as follows:

$$\mathsf{S}(t) = \begin{cases} \varnothing & \text{if } t \in \mathsf{Val} \\ (f^{\mathsf{n}}(\mathsf{N}(t_{j_1}), \dots, \mathsf{N}(t_{j_p}))\ \ \mathsf{S}(t_{i_1})\ \cdots\ \mathsf{S}(t_{i_q})) & \text{if } t = f(t_1, \dots, t_n). \end{cases}$$
$$\mathsf{N}(t) = (\mathsf{S}(t)) \mathbin{@} \mathsf{BN}(t)$$

Here $\mathsf{safe}(f) = \{i_1, \dots, i_q\}$ and $\mathsf{nrm}(f) = \{j_1, \dots, j_p\}$.

The Buchholz-norm of a term t is reflected in the width of $\mathsf{N}(t)$, a property that is essential for the reasoning carried out below.

Lemma 5.14. *For $t \in \mathcal{T}$,* $\mathrm{width}(\mathsf{S}(t)) \leqslant \mathrm{width}(\mathsf{N}(t)) = \|t\| + 1$.

Proof. As $\mathsf{S}(t)$ is a subterm of $\mathsf{N}(t)$, it suffices to show $\mathrm{width}(\mathsf{N}(t)) = \|t\| + 1$. We perform induction on t. For the base case, that is when $t \in \mathcal{V}$ the claim follows from $\mathrm{width}(\mathsf{N}(t)) = \mathrm{width}((\varnothing\ \ \mathsf{s})) = 2$. For the inductive step, assume $t = f(t_1, \dots, t_n)$. Then $\mathsf{N}(t) = (\mathsf{S}(t)) \mathbin{@} \mathsf{BN}(t)$. When $t \in \mathsf{Val}$, then $\mathsf{S}(t) = \varnothing$ and the claim is immediate. So assume $t \notin \mathsf{Val}$, and thus

$$\mathsf{S}(t) = (f^{\mathsf{n}}(\mathsf{N}(t_1), \dots, \mathsf{N}(t_l))\ \ \mathsf{S}(t_{l+1}) \cdots \mathsf{S}(t_n)) \ .$$

Here we assume without loss of generality $\mathsf{safe}(f) = \{l + 1, \dots, n\}$. Unfolding the definition of width and applying the induction hypothesis on $\mathsf{N}(t_i)$ for $i \in \{1, \dots, n\}$ we see $\mathrm{width}(\mathsf{S}(t)) \leqslant \|t\|$. As

$$\mathrm{width}(\mathsf{N}(t)) = \max\{\mathrm{width}(\mathsf{S}(t)), \|t\| + 1\} = \|t\| + 1 \ ,$$

we conclude the lemma. $\square$

We now lift Theorem 3.10 to predicative interpretations. In particular, the next Lemma states that the length of $\blacktriangleright_k$-descents starting from $\mathsf{N}(t)$ for constructor-based terms t is controlled by a polynomial in the size of t.

Lemma 5.15. *Let $t \in \mathcal{T}_{\mathsf{b}} \cup \mathsf{Val}$. For any k, there exist constants $c \in \mathbb{N}$ and $d \in \mathbb{N}$ such that*

$$\mathsf{G}_k(\mathsf{N}(t)) \leqslant c \cdot |t|^d \ .$$

The constants c and d depend only on the cardinality of the considered signature $\mathcal{F}$ and k.

Proof. First assume $t \in \mathsf{Val}$. Thus $\mathsf{N}(t) = (\varnothing\ \ \mathsf{s} \cdots \mathsf{s})$ with $\|t\|$ occurrences of the constant s. Since $\|t\| \leqslant |t|$ and $\mathrm{rank}(\mathsf{s}) = 2$, we conclude with the help of Lemma 3.9 the stronger claim

$$\mathsf{G}_k(\mathsf{N}(t)) = 2 \cdot \|t\| + 1 \leqslant c \cdot |t|$$

for some constant c.

Now assume $t = f(t_1, \dots, t_n) \in \mathcal{T}_{\mathsf{b}}$. Thus by the assumption $t \in \mathcal{T}_{\mathsf{b}}$

$$\mathsf{N}(t) = ((f^{\mathsf{n}}(\mathsf{N}(t_{j_1}), \dots, \mathsf{N}(t_{j_m}))\ \varnothing\ \cdots\ \varnothing)) \mathbin{@} \mathsf{BN}(t)$$

where $\mathsf{nrm}(f) = \{j_1, \ldots, j_l\}$. Let $j \in \mathsf{nrm}(f)$. As $t_j \in \mathsf{Val}$ it follows that $\mathsf{N}(t_j) = (\varnothing) @ \mathsf{BN}(t_j)$, and by Lemma 3.9 we infer $\mathsf{G}_k(\mathsf{N}(t_j)) = 2 \cdot \|t_j\| + 1$. Let $m = \sum_{j \in \mathsf{nrm}(f)} (2 \cdot \|t_j\| + 1)$. According to Theorem 3.10 there exist constants $c_1 \in \mathbb{N}$ and $d_1 \in \mathbb{N}$ with

$$\mathsf{G}_k(f^{\mathsf{n}}(\mathsf{N}(t_{j_1}), \ldots, \mathsf{N}(t_{j_m}))) \leqslant c_1 \cdot (m+2)^{d_1} \ .$$

The constants depend only on $\operatorname{rank}(f^{\mathsf{n}})$, which is bounded by the cardinality of $\mathcal{F}$. Let c_2 be such that $m + 2 \leqslant c_2 \cdot |t| + n$. The constant c_2 exists by the fact $\|t_j\| < |t|$. From this, and with the help of Lemma 3.8 we conclude

$$\begin{aligned}
\mathsf{G}_k(\mathsf{N}(t)) &= \|t\| + 1 + \mathsf{G}_k((f^{\mathsf{n}}(\mathsf{N}(t_{j_1}), \ldots, \mathsf{N}(t_{j_m}))\ \varnothing\ \cdots\ \varnothing)) \\
&\leqslant \|t\| + 1 + n + 1 + \mathsf{G}_k(f^{\mathsf{n}}(\mathsf{N}(t_{j_1}), \ldots, \mathsf{N}(t_{j_m}))) \\
&\leqslant \|t\| + 1 + n + 1 + c_1 \cdot (c_2 \cdot |t|)^{d_1} \ .
\end{aligned}$$

Thus it is easy to see that we can define constants c and d such that $\|t\| + 2 + n + c_1 \cdot (c_2 \cdot |t|)^{d_1} \leqslant c \cdot |t|^d$. The lemma follows. ☐

Assume $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop*}}$ and $\mathcal{S} \subseteq \gtrsim^{\pi}_{\mathsf{pop*}}$. Under these assumptions, a relative step $s \xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{R}/\mathcal{S}} t$ gives rise to a descent $\mathsf{N}(\pi(s)) \blacktriangleright_k \cdots \blacktriangleright_k \mathsf{N}(\pi(t))$ for some uniform natural number k. As indicated by Lemma 5.15, starting from a constructor-based term, the number of descents with respect to $\blacktriangleright_k$ is bounded polynomially in the size of the starting term. From this, a polynomial bound in n on $\mathrm{rc}(n, \mathcal{T}_{\mathsf{b}}, \xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{R}/\mathcal{S}})$ is easy to establish. If we dispense argument filterings, the same observation carries over to relative steps $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$, and thus on $\xrightarrow{\mathsf{i}}_{\mathcal{R}} = \xrightarrow{\mathsf{i}}_{\mathcal{R}/\varnothing}$. In the following, we proof these claims.

Due to compatibility of $\blacktriangleright_k$ with $\sim$, in order to assert an embedding of $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$ into $\blacktriangleright_k^{+}$, it suffices to embed steps due to $\mathcal{R}$ into $\blacktriangleright_k^{+}$, and steps due to $\mathcal{S}$ into $\underset{\sim}{\blacktriangleright}{}_k^{*}$:

Lemma 5.16. *Let* $\mathsf{Q} : \mathcal{T} \to \mathcal{S}\mathrm{eq}$ *be a mapping from terms to sequences. If* $s \xrightarrow{\mathcal{R} \cup \mathcal{S}}_{\mathcal{R}} t \implies \mathsf{Q}(s) \blacktriangleright_k^{+} \mathsf{Q}(t)$, *and moreover* $s \xrightarrow{\mathcal{R} \cup \mathcal{S}}_{\mathcal{S}} t \implies \mathsf{Q}(s) \underset{\sim}{\blacktriangleright}{}_k^{*} \mathsf{Q}(t)$, *then*

$$s \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} t \implies \mathsf{Q}(s) \blacktriangleright_k^{+} \mathsf{Q}(t)$$

follows.

Proof. Consider a relative rewrite step $s \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} t$. There exist terms u and v such that

$$s \xrightarrow{\mathcal{R} \cup \mathcal{S}}{}^{*}_{\mathcal{S}} u \xrightarrow{\mathcal{R} \cup \mathcal{S}}_{\mathcal{R}} v \xrightarrow{\mathcal{R} \cup \mathcal{S}}{}^{*}_{\mathcal{S}} t \ .$$

From the assumptions we infer

$$\mathsf{Q}(s) \underset{\sim}{\blacktriangleright}{}_k^{*} \mathsf{Q}(u) \blacktriangleright_k^{+} \mathsf{Q}(v) \underset{\sim}{\blacktriangleright}{}_k^{*} \mathsf{Q}(t) \ .$$

From the inclusions $\sim \cdot \blacktriangleright_k \subseteq \blacktriangleright_k$ and $\blacktriangleright_k \cdot \sim \subseteq \blacktriangleright_k$ (cf. Lemma 3.5) and from the definition $\underset{\sim}{\blacktriangleright}{}_k = \blacktriangleright_k \cup \sim$ we conclude $\mathsf{Q}(s) \blacktriangleright_k^{+} \mathsf{Q}(t)$ as desired. ☐

5.1 A Simulation of Innermost Steps

Consider the relation $\xrightarrow{\mathsf{v}}_{\mathcal{R}}$, defined as follows:

Definition 5.17. Let $\mathcal{R}$ be a TRS, and let $\mathcal{Q} = \{f(x_1, \dots, x_n) \to \bot \mid f \in \mathcal{D}\}$ for some fresh constant $\bot$. We define

$$\xrightarrow{\mathsf{v}}_{\mathcal{R}} = \xrightarrow{\mathcal{Q}}_{\mathcal{R}} \ .$$

Observe that $\mathsf{NF}(\mathcal{Q}) = \mathsf{Val}$ for $\mathcal{Q}$ as given in the definition above. So in other words, whenever $s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t$ then $s = C[f(\ell_1\sigma, \dots, \ell_n\sigma)]$ for some rule $f(\ell_1, \dots, \ell_n) \to r$ with $\ell_i\sigma \in \mathsf{Val}$ for $i \in \{1, \dots, n\}$. In the remaining, we embed $\xrightarrow{\mathsf{v}}_{\mathcal{R}}$ into $\blacktriangleright_k$ and likewise $\xrightarrow{\mathsf{v}}_{\mathcal{S}}$ into $\underset{\sim}{\blacktriangleright}_k$. We will see that this establishes an embedding of innermost steps: when $\mathcal{R}$ and $\mathcal{S}$ are constructor TRSs, we can simulate $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$ with the help of $\xrightarrow{\mathsf{v}}_{\mathcal{R}}$ and $\xrightarrow{\mathsf{v}}_{\mathcal{S}}$ provided we add suitable additional rules U. Before we continue with the embedding, we formally proof the simulation.

Definition 5.18. Let $(\mathcal{F}, \mathcal{R})$ be a TRS. Without loss of generality, suppose $\bot \in \mathcal{F}$ is a constructor symbol not appearing in $\mathcal{R}$. We define the system $U(\mathcal{R})$ as

$$U(\mathcal{R}) = \{f(t_1, \dots, t_n) \to \bot \mid f(t_1, \dots, t_n) \in \mathsf{NF}(\mathcal{R}) \text{ and } f \in \mathcal{D}\} \ .$$

Observe that $U(\mathcal{R})$ is in any case terminating and confluent, thus it admits the unique normal form property. So it is justified to write $t{\downarrow}_{U(\mathcal{R})}$ for the normal form of t. The rules from $U(\mathcal{R})$ remove defined symbol f in normal forms $t \in \mathsf{NF}(\mathcal{R})$ that are not values:

Lemma 5.19. *Let $\mathcal{R}$ be a TRS and suppose $t \in \mathsf{NF}(\mathcal{R})$. Then $t{\downarrow}_{U(\mathcal{R})} \in \mathsf{Val}$.*

Proof. Easy. □

Consider a step $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$. When $\mathcal{R}$ is a constructor TRS, we can simulate this step by a sequence $s' \xrightarrow{\mathsf{v}}_{\mathcal{R}} \cdot \to_{U(\mathcal{R})} t'$. Here s' and t' are obtained from s and t by normalizing with respect to the TRS $U(\mathcal{R})$. More general, the following lemma holds:

Lemma 5.20. *Let $\mathcal{R}$ and $\mathcal{S}$ be two constructor TRS. Then*

$$s \xrightarrow{\mathcal{R} \cup \mathcal{S}}_{\mathcal{R}} t \Longrightarrow s{\downarrow}_{U(\mathcal{R}\cup\mathcal{S})} \xrightarrow{\mathsf{v}}_{\mathcal{R}} \cdot \to^{*}_{U(\mathcal{R}\cup\mathcal{S})} t{\downarrow}_{U(\mathcal{R}\cup\mathcal{S})} \ .$$

Proof. For brevity, let $U = U(\mathcal{R} \cup \mathcal{S})$. Assume a step $s \xrightarrow{\mathcal{R} \cup \mathcal{S}}_{\mathcal{R}} t$. By definition $s = C[f(\ell_1\sigma, \dots, \ell_n\sigma)]$ and $t = C[r\sigma]$ for some context C, substitution σ and rule $f(\ell_1, \dots, \ell_n) \to r \in \mathcal{R}$. Moreover $\ell_i\sigma \in \mathsf{NF}(\mathcal{R} \cup \mathcal{S})$ for $i \in \{1, \dots, n\}$. We proof the claim by induction on C.

For the base case, assume $C = \Box$. Let $i \in \{1, \dots, n\}$. As $\mathcal{R}$ is a constructor TRS, $\ell_i \in \mathsf{Val}$. From this it is easy to see that $(\ell_i\sigma){\downarrow}_U = \ell_i\tau$ where τ is given such that $\tau(x) = \sigma(x){\downarrow}_U$. Moreover, by Lemma 5.19 it follows that

$(\ell_i\sigma){\downarrow_U} \in \mathsf{Val}$, and so $\ell_i\tau \in \mathsf{Val}$. From the assumption $s \xrightarrow{\mathcal{R}\cup\mathcal{S}}_{\mathcal{R}} t$ we derive $s \notin \mathsf{NF}(\mathcal{R}\cup\mathcal{S})$. We conclude

$$s{\downarrow_U} = f((\ell_1\sigma){\downarrow_U},\dots,(\ell_n\sigma){\downarrow_U}) = f(\ell_1\tau,\dots,\ell_n\tau) \xrightarrow{\mathsf{v},\varepsilon}_{\mathcal{R}} r\tau \to_U^* (r\tau){\downarrow_U} = (r\sigma){\downarrow_U} \ .$$

Next, assume we have shown the property for $s_i \xrightarrow{\mathcal{Q}}_{\mathcal{R}} t_i$ and we want to lift the claim to $s = g(s_1,\dots,s_i,\dots,s_n) \xrightarrow{\mathcal{R}\cup\mathcal{S}}_{\mathcal{R}} g(s_1,\dots,t_i,\dots,s_n) = t$. Then $s{\downarrow_U} = g(s_1{\downarrow_U},\dots,s_i{\downarrow_U},\dots,s_n{\downarrow_U})$, where we use $s_i{\downarrow_U} \notin \mathsf{NF}(\mathcal{R}\cup\mathcal{S})$. The latter is a consequence of the induction hypothesis. Applying the induction hypothesis we see

$$s{\downarrow_U} \xrightarrow{\mathsf{v}}_{\mathcal{R}} \cdot \to_U^* g(s_1{\downarrow_U},\dots,t_i{\downarrow_U},\dots,s_n{\downarrow_U}) \to_U^* t{\downarrow_U}$$

as desired. □

Observe that $U(\mathcal{R})$ just replaces terms by $\bot$, thus rewrite steps due to $U(\mathcal{R})$ can be easily embedded in $\underset{\sim}{\blacktriangleright}_k$:

Lemma 5.21. *Let* $\mathsf{Q} \in \{\mathsf{S},\mathsf{N}\}$ *and* π *be an argument filtering. Then*

$$s \to_{U(\mathcal{R})} t \Longrightarrow \mathsf{Q}(\pi(s)) \mathrel{\underset{\sim}{\blacktriangleright}_k} \mathsf{Q}(\pi(t)) \ .$$

Proof. The lemma can be shown by a straight forward inductive argument, where we employ that $\mathsf{Q}(\pi(s)) \mathrel{\underset{\sim}{\blacktriangleright}_k} \mathsf{Q}(\bot)$ holds independent on s. □

Via the above simulation and Lemma 5.21, it is easy to establish an embedding of $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$ into $\blacktriangleright_k^+$, provided we can embed $\xrightarrow{\mathsf{v}}_{\mathcal{R}}$ and $\xrightarrow{\mathsf{v}}_{\mathcal{S}}$ correspondingly.

Lemma 5.22. *Let* $\mathcal{R}$ *and* $\mathcal{S}$ *be two constructor TRSs. Let* $\mathsf{Q} \in \{\mathsf{S},\mathsf{N}\}$. *If* $s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t \Longrightarrow \mathsf{Q}(\pi(s)) \blacktriangleright_k \mathsf{Q}(\pi(t))$, *and moreover* $s \xrightarrow{\mathsf{v}}_{\mathcal{S}} t \Longrightarrow \mathsf{Q}(\pi(s)) \mathrel{\underset{\sim}{\blacktriangleright}_k} \mathsf{Q}(\pi(t))$, *then*

$$s \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} t \Longrightarrow \mathsf{Q}(\pi(s{\downarrow_{U(\mathcal{R}\cup\mathcal{S})}})) \blacktriangleright_k^+ \mathsf{Q}(\pi(t{\downarrow_{U(\mathcal{R}\cup\mathcal{S})}}))$$

follows.

Proof. For brevity, let $U = U(\mathcal{R}\cup\mathcal{S})$. First, consider a step $s \xrightarrow{\mathcal{R}\cup\mathcal{S}}_{\mathcal{R}} t$ for some terms s and t. From Lemma 5.20 we infer the existence of a term t' such that $s{\downarrow_U} \xrightarrow{\mathsf{v}}_{\mathcal{R}} t' \to_U^* t{\downarrow_U}$. Consider the step $s{\downarrow_U} \xrightarrow{\mathsf{v}}_{\mathcal{R}} t'$. Then from the assumptions of the lemma $\mathsf{Q}(\pi(s{\downarrow_U})) \blacktriangleright_k \mathsf{Q}(\pi(t'))$ follows, and moreover $\mathsf{Q}(\pi(t')) \mathrel{\underset{\sim}{\blacktriangleright}_k^*} \mathsf{Q}(\pi(t{\downarrow_U}))$ follows from $t' \to_U^* t{\downarrow_U}$ according to Lemma 5.21. Summing up, from the definition $\underset{\sim}{\blacktriangleright}_k = \blacktriangleright_k \cup \sim$, compatibility $\underset{\sim}{\blacktriangleright}_k \cdot \sim\, \subseteq \blacktriangleright_k$, and likewise $\sim \cdot \underset{\sim}{\blacktriangleright}_k \subseteq \blacktriangleright_k$ we conclude

$$s \xrightarrow{\mathcal{R}\cup\mathcal{S}}_{\mathcal{R}} t \Longrightarrow \mathsf{Q}(\pi(s{\downarrow_U})) \blacktriangleright_k^+ \mathsf{Q}(\pi(t{\downarrow_U})) \ . \qquad \text{(i)}$$

By same reasoning, we derive

$$s \xrightarrow{\mathcal{R}\cup\mathcal{S}}_{\mathcal{S}} t \Longrightarrow \mathsf{Q}(\pi(s{\downarrow_U})) \mathrel{\underset{\sim}{\blacktriangleright}_k^+} \mathsf{Q}(\pi(t{\downarrow_U})) \ . \qquad \text{(ii)}$$

And so the Lemma follows from (i) and (ii) by the application of Lemma 5.16. □

In turn, via an appropriate embedding of $\xrightarrow{\mathsf{v}}_{\mathcal{R}}$ and $\xrightarrow{\mathsf{v}}_{\mathcal{S}}$ we can estimate the number of relative steps $\xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}$ in terms of $\blacktriangleright_k$-descents. This is reflected in the following Lemma.

Lemma 5.23. *Let $\mathcal{R}$ and $\mathcal{S}$ be two constructor TRSs. Let $\mathsf{Q} : \mathcal{T} \to \mathcal{S}\mathrm{eq}$ be a mapping from terms to sequences. If $s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t \Longrightarrow \mathsf{Q}(\pi(s)) \blacktriangleright_k \mathsf{Q}(\pi(t))$, and moreover $s \xrightarrow{\mathsf{v}}_{\mathcal{S}} t \Longrightarrow \mathsf{Q}(\pi(s)) \underset{\sim}{\blacktriangleright}_k \mathsf{Q}(\pi(t))$, then*

$$\mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}) \leqslant \mathsf{G}_k(\mathsf{Q}(\pi(t))) + c$$

for $t \in \mathcal{T}_{\mathsf{b}}$ and constant $c \in \mathbb{N}$ depending only on Q.

Proof. Define $U = U(\mathcal{R} \cup \mathcal{S})$. Assume a maximal derivation starting from $t \in \mathcal{T}_{\mathsf{b}}$:

$$t = t_0 \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} t_1 \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} \cdots \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} t_\ell \ .$$

Thus it suffices to show $\ell \leqslant \mathsf{G}_k(\mathsf{Q}(\pi(t))) + c$. From the assumptions, by Lemma 5.22 we derive

$$\mathsf{Q}(\pi(t{\downarrow_U})) = \mathsf{Q}(\pi(t_0{\downarrow_U})) \blacktriangleright_k^+ \mathsf{Q}(\pi(t_1{\downarrow_U})) \blacktriangleright_k^+ \cdots \blacktriangleright_k^+ \mathsf{Q}(\pi(t_\ell{\downarrow_U})) \ .$$

Define $c = \mathsf{G}_k(\mathsf{Q}(\bot))$. We differentiate two cases. If $t \notin \mathsf{NF}(\mathsf{Q})$, from $t \in \mathcal{T}_{\mathsf{b}}$ we derive $t{\downarrow_U} = t$, and clearly $\ell \leqslant \mathsf{G}_k(\mathsf{Q}(\pi(t{\downarrow_U}))) = \mathsf{G}_k(\mathsf{Q}(\pi(t)))$ proves the lemma. For the case when $t \in \mathsf{NF}(\mathsf{Q})$, $t{\downarrow_U} = \bot$ and hence $\mathsf{G}_k(\mathsf{Q}(\pi(t{\downarrow_U}))) = \mathsf{G}_k(\mathsf{Q}(\bot)) = c$. Again the lemma follows. □

Naturally, for the application of the above lemma it suffices to only embed those steps that can be employed in a derivation starting from a constructor-based term with respect to the considered relative rewrite relation. For instance, by inspecting the proofs of Lemma 5.22 and Lemma 5.23, it is easy to see that when we consider the relation $\xrightarrow{\mathsf{i}}^{\varepsilon}_{\mathcal{R}/\mathcal{S}}$ only an embedding of top-steps due to $\mathcal{R}$ needs to be provided.

5.2 The Embedding of Top-Steps

The observations from the previous section justify to consider only steps $s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t$. Below, we first show an embedding of top-steps $\xrightarrow{\mathsf{v}}^{\varepsilon}_{\mathcal{R}}$ into $\blacktriangleright_k$, that is, from compatibility of $\mathcal{R}$ with an instance $>^{\pi}_{\mathsf{pop}*}$ we show that $s \xrightarrow{\mathsf{v}}^{\varepsilon}_{\mathcal{R}} t$ implies $\mathsf{N}(\pi(s)) \blacktriangleright_k \mathsf{N}(\pi(s))$ for some fixed k. In the subsequent lemma, we then lift this embedding to steps below the root. In particular, we show that under the assumption $\mathcal{S} \subseteq \gtrsim^{\pi}_{\mathsf{pop}*}$ it follows that $s \xrightarrow{\mathsf{v}}_{\mathcal{S}} t$ implies $\mathsf{N}(\pi(s)) \underset{\sim}{\blacktriangleright}_k \mathsf{N}(\pi(t))$. As a consequence of Lemma 5.23, the number of $\xrightarrow{\mathsf{i}}^{\varepsilon}_{\mathcal{R}/\mathcal{S}}$ descents can be estimated in terms of $\blacktriangleright_k$-descents. We continue with two simple observations.

Lemma 5.24. *Let π be an argument filtering and let $s, t \in \mathcal{T}$ be two terms. If $s \overset{s}{\sim}_{\pi} t$ then $\|\pi(s)\| = \|\pi(t)\|$ and $\mathsf{Q}(\pi(s)) \sim \mathsf{Q}(\pi(t))$ for $\mathsf{Q} \in \{\mathsf{S}, \mathsf{N}\}$.*

Proof. The lemma follows by a straight forward inductive argument. □

Lemma 5.25. *Let π be an argument filtering and let $s, t \in \mathcal{T}$ be two terms such that $\pi(s) \in \mathsf{Val}$. If $s >^{\pi}_{\mathsf{pop}*} t$ or $s >^{\pi}_{\mathsf{pop}} t$ then $\|\pi(s)\| > \|\pi(t)\|$, $\mathsf{N}(\pi(s)) \gtrdot_1 \mathsf{N}(\pi(t))$ and $\mathsf{S}(\pi(s)) = \varnothing = \mathsf{S}(\pi(t))$.*

Proof. As $>^{\pi}_{\mathsf{pop}} \subseteq >^{\pi}_{\mathsf{pop}*}$, it suffices to show the lemma for $>^{\pi}_{\mathsf{pop}*}$. Let $s, t \in \mathcal{T}$ with $s >^{\pi}_{\mathsf{pop}*} t$ and $\pi(s) \in \mathsf{Val}$. According to Lemma 5.12 there exists some position $p \neq \varepsilon$ such that $\pi(s)|_p \overset{\mathsf{s}}{\sim} \pi(t)$. From this it is easy to see that $\pi(t) \in \mathsf{Val}$ and moreover $\|\pi(s)\| > \|\pi(t)\|$. By inspecting Definition 5.13 the claim follows. $\square$

The embedding of $\xrightarrow{\mathsf{v}}{}^{\varepsilon}_{\mathcal{R}}$ into $\blacktriangleright_k$ is a direct consequence of the following two Lemmata.

Lemma 5.26. *Let $\sigma \colon \mathcal{V} \to \mathsf{Val}$, and let $s = f(s_1, \dots, s_n) \in \mathcal{T}_{\mathsf{b}}$ with $f \in \mathcal{D}$ and $t \in \mathcal{T}$ be two terms. If $s >^{\pi}_{\mathsf{pop}} t$ and π does not collapse f then*

$$f^{\mathsf{n}}(\mathsf{N}(\pi(s_{i_1}\sigma)), \dots, \mathsf{N}(\pi(s_{i_l}\sigma))) \gtrdot_k \mathsf{N}(\pi(t\sigma))$$

where $\mathsf{nrm}_{\pi}(s) = \{s_{i_1}, \dots, s_{i_l}\}$ and $k = 3 \cdot \|\pi(t)\|$.

Proof. For the ease of presentation, without loss of generality we assume that $\mathsf{safe}(f) = \{p+1, \dots, n\}$, and $\pi(f) = \{1, \dots, l\} \uplus \{p+1, \dots, m\}$. Let $u = f^{\mathsf{n}}(\mathsf{N}(\pi(s_1\sigma)), \dots, \mathsf{N}(\pi(s_l\sigma)))$, and thus we need to show $u \gtrdot_k \mathsf{N}(\pi(t\sigma))$. We perform induction on $>^{\pi}_{\mathsf{pop}}$.

- Case $s >^{\pi}_{\mathsf{pop}} t$ (1): For this case, $\pi(f) = i$. As π collapses f, the lemma is vacuously satisfied.

- Case $s >^{\pi}_{\mathsf{pop}} t$ (2): By Definition 5.8, since $f \in \mathcal{D}$, there exists some normal argument position $i \in \{1, \dots, l\}$ such that $s_i \gtrsim^{\pi}_{\mathsf{pop}} t$ holds. Observe that $\mathsf{N}(\pi(s_i\sigma))$ is a direct subterm of u. It suffices to show $\mathsf{N}(\pi(s_i\sigma)) \gtrsim_k \mathsf{N}(\pi(t\sigma))$. By closure under substitution of $>^{\pi}_{\mathsf{pop}*}$ (Lemma 5.12) we derive $s_i\sigma >^{\pi}_{\mathsf{pop}} t\sigma$. Finally, from $\pi(s_i\sigma) \in \mathsf{Val}$ we conclude the claim with the help of Lemma 5.25.

- Case $s >^{\pi}_{\mathsf{pop}} t$ (3a): Assume $t = g(t_1, \dots, t_{n'})$, $\pi(g) = j$ and $s >^{\pi}_{\mathsf{pop}} t_j$. Then $\pi(t\sigma) = \pi(t_j\sigma)$, and we directly conclude the lemma from the induction hypothesis $u \gtrdot_k \mathsf{N}(\pi(t_j\sigma)) = \mathsf{N}(\pi(t\sigma))$. Notice that we employ $\gtrdot^{\ell}_{l} \subseteq \gtrdot_k$ for $k \geqslant \ell$ and $k \geqslant l$. Below we use this fact frequently without explicitly referring to it.

- Case $s >^{\pi}_{\mathsf{pop}} t$ (3b): For this case, let $t = g(t_1, \dots, t_{n'})$. Without loss of generality, we assume $\mathsf{safe}(g) = \{p'+1, \dots, n'\}$ and $\pi(g) = \{1, \dots, l'\} \uplus \{p'+1, \dots, m'\}$. Furthermore, $f \succ g$ and $s >^{\pi}_{\mathsf{pop}} t_j$ for all $j \in \pi(g)$. For $j \in \pi(g)$, since $\|\pi(t)\| > \|\pi(t_j)\|$, specializing the induction hypothesis (IH) yields

$$u \gtrdot_{k-3} (\mathsf{S}(\pi(t_j\sigma))) \mathbin{@} \mathsf{BN}(\pi(t_j\sigma)) = \mathsf{N}(\pi(t_j\sigma)) \,. \tag{IH}$$

First, we show $u >_{k-1} \mathsf{S}(\pi(t\sigma))$. When $\pi(t\sigma) \in \mathsf{Val}$, then $\mathsf{S}(\pi(t\sigma)) = \varnothing$ and $u >_k \mathsf{S}(\pi(t\sigma))$ trivially holds. So assume $\pi(t\sigma) \notin \mathsf{Val}$. By definition,

$$\mathsf{S}(\pi(t\sigma)) = (g^{\mathsf{n}}(\mathsf{N}(\pi(t_1\sigma)), \dots, \mathsf{N}(\pi(t_{l'}\sigma)))\ \mathsf{S}(\pi(t_{p'+1}\sigma)) \cdots \mathsf{S}(\pi(t_{m'}\sigma)))\ .$$

First, observe that

$$u = f^{\mathsf{n}}(\mathsf{N}(\pi(s_1\sigma)), \dots, \mathsf{N}(\pi(s_l\sigma)) >_{k-2} g^{\mathsf{n}}(\mathsf{N}(\pi(t_1\sigma)), \dots, \mathsf{N}(\pi(t_{l'}\sigma))) \quad \text{(i)}$$

holds by clause (2) from Definition 3.3: We have $f \succ g$ by assumption, $u >_{k-2} \mathsf{N}(\pi(t_j\sigma))$ follows from (IH) and moreover $\mathrm{width}(u) + (k-2) > l'$ is a consequence of $k = 3 \cdot \|\pi(t)\|$, $\|\pi(t)\| > l'$. By (i), and through similar reasoning as above, we infer

$$\begin{aligned} u &>_{k-1} (g^{\mathsf{n}}(\mathsf{N}(\pi(t_1\sigma)), \dots, \mathsf{N}(\pi(t_{l'}\sigma)))\ \mathsf{S}(\pi(t_{p'+1}\sigma)) \cdots \mathsf{S}(\pi(t_{m'}\sigma))) \quad \text{(ii)} \\ &= \mathsf{S}(\pi(t\sigma))\ . \end{aligned}$$

Here we employ $u >_{k-3} \mathsf{S}(\pi(t_j\sigma))$ for $j \in \{p'+1, \dots, m'\}$, which is a direct consequence of (IH). Finally, as we show below,

$$\mathrm{width}(u) + k > \mathrm{width}(\mathsf{N}(\pi(t\sigma))) = \|\pi(t\sigma)\| + 1 \quad \text{(iii)}$$

holds. Here the last equality follows from Lemma 5.14. By inspecting clause (3) from Definition 3.3, it can be easily seen that (ii) together with $u >_1 \mathsf{s}$, $k > 1$ and (iii) suffices to show

$$u \blacktriangleright_k (\mathsf{S}(\pi(t\sigma)))\ @\ \mathsf{BN}(\pi(t\sigma)) = \mathsf{N}(\pi(t\sigma))\ .$$

We finish with a proof of (iii). Let $\ell = l' + m' - p'$, that is, the number of direct subterms of $\pi(t\sigma)$. By definition

$$\|\pi(t\sigma)\| = \max\{\ell, \max\{\|\pi(t_j\sigma)\| \mid j \in \pi(g)\}\} + 1\ .$$

Thus either $\|\pi(t\sigma)\| = \ell + 1$ or $\|\pi(t\sigma)\| = \pi(t_j\sigma) + 1$ for some argument position j not erased by the argument filtering. For the first case, it is easy to see that also $\|\pi(t)\| = \ell + 1$, and thus

$$\mathrm{width}(u) + k = \mathrm{width}(u) + 3 \cdot (\ell + 1) > (\ell + 1) + 1 = \|\pi(t\sigma)\| + 1\ .$$

For the second case, assume $\|\pi(t\sigma)\| = \|\pi(t_j\sigma)\| + 1$ for some $j \in \pi(g)$. From the induction hypothesis (IH) for the particular subterm t_j, with the help of Lemma 3.6 and Lemma 5.14, we infer

$$\mathrm{width}(u) + (k-3) > \mathrm{width}(\mathsf{N}(\pi(t_j\sigma))) = \|\pi(t_j\sigma)\| + 1 = \|\pi(t\sigma)\|\ .$$

Thus (iii) follows, which proves the lemma.

□

Lemma 5.27. *Let $\sigma\colon \mathcal{V} \to \mathsf{Val}$, and let $s = f(s_1, \dots, s_n) \in \mathcal{T}_{\mathsf{b}}$ with $f \in \mathcal{D}$ and $t \in \mathcal{T}$ be two terms. If $s >^{\pi}_{\mathsf{pop}*} t$ and π does not collapse f then*

(1) $f^{\mathsf{n}}(\mathsf{N}(\pi(s_{i_1}\sigma)), \dots, \mathsf{N}(\pi(s_{i_l}\sigma))) \blacktriangleright_k \mathsf{S}(\pi(t\sigma))$*, and*

(2) $(f^{\mathsf{n}}(\mathsf{N}(\pi(s_{i_1}\sigma)), \dots, \mathsf{N}(\pi(s_{i_l}\sigma)))) \mathbin{@} \mathsf{BN}(\pi(s\sigma)) \blacktriangleright_k \mathsf{N}(\pi(t\sigma))$*.*

Here $\mathsf{nrm}_\pi(s) = \{s_{i_1}, \dots, s_{i_l}\}$ *and* $k = 3 \cdot \|\pi(t)\|$*.*

Proof. We perform induction on $>_{\mathsf{pop}*}$. For this, without loss of generality assume $\mathsf{safe}(f) = \{p+1, \dots, n\}$, and $\pi(f) = \{1, \dots, l\} \uplus \{p+1, \dots, m\}$. Let $u = f^{\mathsf{n}}(\mathsf{N}(\pi(s_1\sigma)), \dots, \mathsf{N}(\pi(s_l\sigma)))$. If $s >_{\mathsf{pop}} t$, then the claim is immediate by Lemma 5.26. We continue by case analysis. Below, we present only non-trivial cases.

– Case $s >_{\mathsf{pop}*} t$ (2): By assumption, there exists some $i \in \pi(f)$ such that $s_i \gtrsim^{\pi}_{\mathsf{pop}*} t$ holds. Since $\pi(s_i\sigma) \in \mathsf{Val}$, Lemma 5.25 reveal $\mathsf{N}(\pi(s_i\sigma)) \underset{\sim}{\blacktriangleright}_k \mathsf{N}(\pi(t\sigma))$ and $\mathsf{S}(\pi(t\sigma)) = \varnothing$. Property (1) is immediate. In order to conclude Property (2), observe that since $\pi(s_i\sigma)$ is a subterm of $\pi(s\sigma)$, $\|\pi(s\sigma)\| > \|\pi(s_i\sigma)\|$ holds. From this, by Lemma 5.25 together with the assumption $s_i \gtrsim_{\mathsf{pop}*} t$, closure under substitution of $>_{\mathsf{pop}*}$ (Lemma 5.12) and $\pi(s_i\sigma) \in \mathsf{Val}$ we derive

$$\|\pi(s\sigma)\| + k > \|\pi(s_i\sigma)\| + k \geqslant \|\pi(t\sigma)\| \ .$$

From this and Property (1), by one application of clause (5) from Definition 3.4 we conclude Property (2).

– Case $s >^{\pi}_{\mathsf{pop}} t$ (3*b*): For this case, let $t = g(t_1, \dots, t_{n'})$. Without loss of generality, we assume $\mathsf{safe}(g) = \{p'+1, \dots, n'\}$ and $\pi(g) = \{1, \dots, l'\} \uplus \{p'+1, \dots, m'\}$. Furthermore we have $f \succ g$ in the precedence, $s >^{\pi}_{\mathsf{pop}*} t_{j_0}$ for some $j_0 \in \{p'+1, \dots, m'\}$, and for all $j \neq j_0$ either $s >^{\pi}_{\mathsf{pop}} t_j$, or $\pi(s) \rhd \pi(t_j)$ with $j \in \{p'+1, \dots, m'\}$.

First, we show $u \gtrdot_k \mathsf{S}(\pi(t\sigma))$. The only non-trivial case is $\pi(t\sigma) \notin \mathsf{Val}$. For this case, by definition

$$\mathsf{S}(\pi(s\sigma)) = (g^{\mathsf{n}}(\mathsf{N}(\pi(t_1\sigma)), \dots, \mathsf{N}(\pi(t_{l'}\sigma)))\ \ \mathsf{S}(\pi(t_{p'+1}\sigma)) \cdots \mathsf{S}(\pi(t_{m'}\sigma))) \ .$$

From $s >^{\pi}_{\mathsf{pop}} t_j$ for all normal argument positions $\{1, \dots, l'\}$, exactly as in the corresponding case of Lemma 5.26, we derive

$$u \gtrdot_{k-2} g^{\mathsf{n}}(\mathsf{N}(\pi(t_1\sigma)), \dots, \mathsf{N}(\pi(t_{l'}\sigma))) \ . \tag{i}$$

Next, observe

$$u \gtrdot_{k-2} \mathsf{S}(\pi(t_j\sigma)) \text{ for all } j \neq j_0,\ j \in \{p'+1, \dots, m'\} \ . \tag{ii}$$

We either have $s_i \trianglerighteq t_j$ for some $i \in \pi(f)$ or $s >_{\mathsf{pop}} t_j$. For the first case, from $\pi(s_i\sigma) \in \mathsf{Val}$ we infer $\pi(t_j\sigma) \in \mathsf{Val}$, and thus $u \gtrdot_{k-2} \varnothing = \mathsf{S}(\pi(t_j\sigma))$ trivially holds. For the second case, by Lemma 5.26 we see $u \gtrdot_{k-2} \mathsf{N}(\pi(t_j\sigma))$ which shows (ii). Furthermore, the induction hypothesis reveals

$$u \blacktriangleright_{k-3} \mathsf{S}(\pi(t_{j_0}\sigma)) \text{ and } (u) \mathbin{@} \mathsf{BN}(\pi(s\sigma)) \blacktriangleright_{k-3} \mathsf{N}(\pi(t_{j_0}\sigma)) \ . \tag{IH}$$

Let $\ell = l' + m' - p'$ be the number of direct subterms of $\pi(t\sigma)$. Observe that $k = 3 \cdot \|\pi(t\sigma)\| \geqslant 3 \cdot (\ell + 1) > (m' - p') + 1$ holds. Combining this with (i), (ii) and (IH) we derive

$$u \blacktriangleright_k \mathsf{S}(\pi(t\sigma)) \tag{iii}$$

by one application of clause (3) from Definition 3.4. From (iii), in order to show property (2), that is

$$(u) \mathbin{@} \mathsf{BN}(\pi(s\sigma)) \blacktriangleright_k (\mathsf{S}(\pi(t\sigma))) \mathbin{@} \mathsf{BN}(\pi(t\sigma)) = \mathsf{N}(\pi(t\sigma)) \ ,$$

it suffices (by clause (5) from Definition 3.4 and (iii)) to show

$$\|\pi(s\sigma)\| + k > \|\pi(t\sigma)\| \ . \tag{iv}$$

In (iv) we employ that both $\mathrm{width}((u) \mathbin{@} \mathsf{BN}(\pi(s\sigma))) = \|\pi(s\sigma)\| + 1$ and $\mathrm{width}(\mathsf{N}(\pi(t\sigma))) = \|\pi(t\sigma)\| + 1$, which are consequences of Lemma 5.14. Let $\ell = l' + m' - p'$, and thus

$$\|\pi(t\sigma)\| = \max\{\ell, \max\{\|\pi(t_j\sigma)\| \mid j \in \pi(g)\}\} + 1 \ .$$

When $\|\pi(t\sigma)\| = \ell+1$, as in the corresponding case in the proof of Lemma 5.26, we see that $k > \|\pi(t\sigma)\|$ and conclude (iv). So assume $\|\pi(t\sigma)\| = \|\pi(t_j\sigma)\| + 1$ for some $j \in \pi(g)$. By the assumptions, either $s >^{\pi}_{\mathsf{pop*}} t_j$, $s >^{\pi}_{\mathsf{pop}} t_j$ or $s \rhd t_j$. We continue by case analysis.

First, assume $s >^{\pi}_{\mathsf{pop*}} t_j$, and thus by the specialized induction hypothesis (IH) we obtain $(u) \mathbin{@} \mathsf{BN}(\pi(s\sigma)) \blacktriangleright_{k-3} \mathsf{N}(\pi(t_j\sigma))$. From Lemma 3.6 and 5.14 we conclude $\|\pi(s\sigma)\| + (k-3) > \|\pi(t_j\sigma)\|$, and (iv) follows.

Next, assume $s >^{\pi}_{\mathsf{pop}} t_j$. Thus $u \gtrdot_{k-3} \mathsf{N}(\pi(t_j\sigma))$ holds according to Lemma 5.26, and by the inclusion $\gtrdot_k \subseteq \blacktriangleright_k$ we conclude (iv) as above with the help of Lemma 3.6 and 5.14.

Finally, assume $s \rhd t_j$, so $s_i \trianglerighteq t_j$ for some unfiltered argument position i. Clearly $\|\pi(s_i\sigma)\| \geqslant \|\pi(t_j\sigma)\|$. From $\|\pi(s\sigma)\| > \|\pi(s_i\sigma)\| \geqslant \|\pi(t_j\sigma)\|$ we infer (iv).

- Case $s >^{\pi}_{\mathsf{pop}} t$ (3c): Again, let $t = g(t_1, \dots, t_{n'})$ and assume $\mathsf{safe}(g) = \{p'+1, \dots, n'\}$ and $\pi(g) = \{1, \dots, l'\} \uplus \{p'+1, \dots, m'\}$. Furthermore, assume we have $f \sim g$ and both $\mathsf{nrm}_\pi(s) >^{\pi,\mathsf{mul}}_{\mathsf{pop*}} \mathsf{nrm}_\pi(t)$, as well as $\mathsf{safe}_\pi(s) \gtrsim^{\pi,\mathsf{mul}}_{\mathsf{pop*}} \mathsf{safe}_\pi(t)$, holds.

  From the strict multiset decrease of normal arguments, we infer

$$(\mathsf{N}(\pi(s_1\sigma)) \ \cdots \ \mathsf{N}(\pi(s_l\sigma))) \blacktriangleright_{k-1} (\mathsf{N}(\pi(t_1\sigma)) \ \cdots \ \mathsf{N}(\pi(t_{l'}\sigma)))$$

  from Lemma 5.25 applied to the arguments $s_1, \dots, s_l$ and $t_1, \dots, t_{l'}$ respectively. Thus

$$f^{\mathsf{n}}(\mathsf{N}(\pi(s_1\sigma)), \dots, \mathsf{N}(\pi(s_l\sigma))) \blacktriangleright_{k-1} g^{\mathsf{n}}(\mathsf{N}(\pi(t_1\sigma)), \dots, \mathsf{N}(\pi(t_{l'}\sigma))) \tag{v}$$

follows from $f^{\mathsf{n}} \sim g^{\mathsf{n}}$. Furthermore, by the assumptions $\pi(s_i\sigma) \in \mathsf{Val}$ for $i \in \pi(f)$, the multiset comparison of safe arguments reveals $\pi(t_j\sigma) \in \mathsf{Val}$ for $j \in \{p'+1, \ldots, m'\}$ in combination with Lemma 5.12 and Lemma 5.25. From this, we conclude property (1), since

$$u \blacktriangleright_k (g^{\mathsf{n}}(\mathsf{N}(\pi(t_1\sigma)), \ldots, \mathsf{N}(\pi(t_{l'}\sigma)))\ \varnothing \cdots \varnothing) = \mathsf{S}(\pi(t\sigma))$$

follows from (v) and one application of clause (3) from Definition 3.4. Here, we additionally employ $k > \ell + 1$, where $\ell = l' + (m' - p')$.

It is easy to see that property (2) follows from property (1) by one application of clause (5) from Definition 3.4. However, we additionally need to proof

$$\mathrm{width}((u)\ @\ \mathsf{BN}(\pi(s\sigma))) + k > \mathrm{width}(\mathsf{N}(\pi(t\sigma)))\ . \tag{vi}$$

Remember that

$$\mathrm{width}((u)\ @\ \mathsf{BN}(\pi(s\sigma))) = \|\pi(s\sigma)\| + 1 \text{ and}$$
$$\mathrm{width}(\mathsf{N}(\pi(t\sigma))) = \|\pi(t\sigma)\| + 1$$

are consequences of Lemma 5.14. Since $k \geqslant 1$ it is easy to see that $\|\pi(s\sigma)\| \geqslant \|\pi(t\sigma)\|$ establishes (vi). We finish with a proof of $\|\pi(s\sigma)\| \geqslant \|\pi(t\sigma)\|$: As $\mathsf{safe}_\pi(s) \gtrsim^{\pi,\mathsf{mul}}_{\mathsf{pop*}} \mathsf{safe}_\pi(t)$ and $\mathsf{nrm}_\pi(s) >^{\pi,\mathsf{mul}}_{\mathsf{pop*}} \mathsf{nrm}_\pi(t)$ hold by assumption, for each argument t_j not erased by π we infer $s_i \gtrsim^{\pi}_{\mathsf{pop*}} t_j$ for some s_i with $i \in \pi(f)$. From this, since all arguments $s_i \in \mathsf{Val}$ by assumption, by the combination of Lemma 5.12 and Lemma 5.25 we conclude $\|\pi(s_i\sigma)\| \geqslant \|\pi(t_j\sigma)\|$ for all arguments t_j not erased by the argument filtering. Thus

$$\begin{aligned} \|\pi(s\sigma)\| &= \max\{\ell, \max\{\|\pi(s_i\sigma)\| \mid i \in \pi(f)\}\} + 1 \\ &\geqslant \max\{\ell, \max\{\|\pi(t_j\sigma)\| \mid j \in \pi(g)\}\} + 1 \\ &= \|\pi(t\sigma)\| \end{aligned}$$

is immediate.

□

Lemma 5.28. *Let $\mathcal{R}$ be a TRS compatible with $>^{\pi}_{\mathsf{pop*}}$, i.e. suppose $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop*}}$ holds. Define $k = 3 \cdot \max\{\|\pi(r)\| \mid \ell \to r \in \mathcal{R}\}$. If $s \xrightarrow{\mathsf{v}}{}^{\epsilon}_{\mathcal{R}} t$ then*

(1) either $\mathsf{S}(\pi(s)) \blacktriangleright_k \mathsf{S}(\pi(t))$ or $\mathsf{S}(\pi(s)) = \varnothing = \mathsf{S}(\pi(t))$, and

(2) $\mathsf{N}(\pi(s)) \blacktriangleright_k \mathsf{N}(\pi(t))$.

Proof. By the assumptions, $s = f(\ell_1\sigma, \ldots, \ell_n\sigma)$ and $t = r\sigma$ for $\sigma : \mathcal{V} \to \mathsf{Val}$ and rule $f(\ell_1, \ldots, \ell_n) \to r \in \mathcal{R}$. Moreover, $f(\ell_1, \ldots, \ell_n) >^{\pi}_{\mathsf{pop*}} r$ by the assumption $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop*}}$. We proceed by case analysis on $\pi(f)$. First, assume $\pi(f) = i$ for some $i \in \{1, \ldots, n\}$, and thus $\pi(s) = \pi(\ell_i\sigma) \in \mathsf{Val}$. By closure under substitution of $>^{\pi}_{\mathsf{pop*}}$ and Lemma 5.25 we conclude $\mathsf{N}(\pi(s)) \blacktriangleright_k \mathsf{N}(\pi(t))$ with $\pi(t) \in \mathsf{Val}$. Moreover, from $\pi(s) \in \mathsf{Val}$ and $\pi(t) \in \mathsf{Val}$ it is easy to see that $\mathsf{S}(\pi(s)) = \varnothing = \mathsf{S}(\pi(t))$ holds.

Next, without loss of generality assume $\pi(f) = \{1, \dots, q\} \uplus \{p+1, \dots, m\}$ with $\{1, \dots, q\} \subseteq \mathsf{nrm}(f)$ and $\{p+1, \dots, m\} \subseteq \mathsf{safe}(f)$. Furthermore, let $u = f^{\mathsf{n}}(\mathsf{N}(\pi(\ell_1\sigma)), \dots, \mathsf{N}(\pi(\ell_q\sigma)))$. By the assumptions on π, we see $\pi(s) \notin \mathsf{Val}$ and so

$$\begin{aligned} \mathsf{S}(\pi(s)) &= (u \varnothing \cdots \varnothing), \text{ and} \\ \mathsf{N}(\pi(s)) &= (\mathsf{S}(\pi(s))) \mathbin{@} \mathsf{BN}(\pi(s)) \ . \end{aligned}$$

Lemma 5.27 reveals $u \blacktriangleright_k \mathsf{S}(\pi(t))$ and $(u) \mathbin{@} \mathsf{BN}(\pi(s)) \blacktriangleright_k \mathsf{N}(\pi(t))$. From the first inequality, $\mathsf{S}(\pi(s)) \blacktriangleright_k \mathsf{S}(\pi(t))$ is immediate, and it is not difficult to reason that from the second inequality also $\mathsf{N}(s) \blacktriangleright_k \mathsf{N}(t)$ follows. This establishes the lemma. □

We now lift the embedding to arbitrary contexts. Naturally, from the assumption $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop}*}$ we can only achieve $\mathsf{Q}(\pi(s)) \mathrel{\underset{\sim}{\blacktriangleright}} \mathsf{Q}(\pi(t))$ from $s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t$ and predicative interpretation Q. The primary reason is that π may remove the rewrite position of $s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t$, and for that case clearly $\pi(s) = \pi(t)$ holds.

Lemma 5.29. *Let $\mathcal{R}$ be a TRS compatible with $>^{\pi}_{\mathsf{pop}*}$, i.e $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop}*}$ holds. Define $k = 3 \cdot \max\{\|\pi(r)\| \mid \ell \to r \in \mathcal{R}\}$, and let $\mathsf{Q} \in \{\mathsf{S}, \mathsf{N}\}$. Then*

$$s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t \Longrightarrow \mathsf{Q}(\pi(s)) \mathrel{\underset{\sim}{\blacktriangleright}}_k \mathsf{Q}(\pi(t)) \ .$$

Proof. Assume $s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t$, and thus $s = C[\ell\sigma]$ and $t = C[r\sigma]$ for some context C, $\ell \to r \in \mathcal{R}$ and substitution $\sigma : \mathcal{V} \to \mathsf{Val}$. We continue with a proof by induction on the context C, where beside $\mathsf{Q}(\pi(s)) \mathrel{\underset{\sim}{\blacktriangleright}}_k \mathsf{Q}(\pi(t))$ we show

$$\mathsf{S}(\pi(s)) \sim \mathsf{S}(\pi(t)) \Longrightarrow \|\pi(s)\| \geqslant \|\pi(t)\| \ .$$

For the base case $C = \Box$, the Lemma follows from Lemma 5.28. Observe that when $\mathsf{S}(\pi(s)) \sim \mathsf{S}(\pi(t))$ then $\mathsf{S}(\pi(s)) = \varnothing = \mathsf{S}(\pi(t))$ and so $\mathsf{N}(\pi(t)) = (\varnothing) \mathbin{@} \mathsf{BN}(\pi(t)) \mathrel{\underset{\sim}{\blacktriangleright}}_k (\varnothing) \mathbin{@} \mathsf{BN}(\pi(s)) = \mathsf{N}(\pi(s))$. From this it is easy to see that $\|\pi(s)\| \geqslant \|\pi(t)\|$ holds.

For the inductive step, suppose $C = f(s_1, \dots, C', \dots, s_n)$ for some context C'. Let $s_i = C'[\ell\sigma]$ and $t_i = C'[r\sigma]$. The induction hypothesis yields $\mathsf{Q}(\pi(s_i)) \mathrel{\underset{\sim}{\blacktriangleright}}_k \mathsf{Q}(\pi(t_i))$ for $\mathsf{Q} \in \{\mathsf{S}, \mathsf{N}\}$. Furthermore, $\|\pi(s_i)\| \geqslant \|\pi(t_i)\|$ when $\mathsf{S}(\pi(s_i)) \sim \mathsf{S}(\pi(t_i))$. When $\pi(f) = i$ then the Lemma follows directly from induction hypothesis, so assume $\pi(f) = \{1, \dots, l\} \uplus \{p+1, \dots, m\}$ and let $\mathsf{Q} \in \{\mathsf{S}, \mathsf{N}\}$. When $i \notin \pi(f)$ then $\mathsf{Q}(\pi(s)) = \mathsf{Q}(\pi(t))$ and we directly conclude the lemma. Notice that $\mathsf{N}(\pi(s)) = \mathsf{N}(\pi(t))$ almost trivially yields $\|\pi(s)\| = \|\pi(t)\|$. On the other hand, assume $i \in \pi(f)$. Two cases are possible:

- Case $\pi(s) \in \mathsf{Val}$: Then clearly $\pi(s_i) \in \mathsf{Val}$, and the induction hypothesis can be specialized to $\mathsf{S}(\pi(s_i)) = \varnothing = \mathsf{S}(\pi(t_i))$. So $\pi(t_i) \in \mathsf{Val}$ by definition of S and hence $\pi(t) \in \mathsf{Val}$ follows. Moreover, the induction hypothesis yields $\|\pi(s_i)\| \geqslant \|\pi(t_i)\|$ and we conclude $\|\pi(s)\| \geqslant \|\pi(t)\|$ by definition of $\|\cdot\|$. Clearly $\mathsf{N}(\pi(t)) = (\varnothing) \mathbin{@} \mathsf{BN}(\pi(t)) \mathrel{\underset{\sim}{\blacktriangleright}}_k (\varnothing) \mathbin{@} \mathsf{BN}(\pi(s)) = \mathsf{N}(\pi(s))$ follows.

– CASE $\pi(s) \notin \mathsf{Val}$: Then

$$\mathsf{S}(\pi(s)) = (f^{\mathsf{n}}(\mathsf{N}(\pi(s_1)), \ldots, \mathsf{N}(\pi(s_l)))\ \ \mathsf{S}(\pi(s_{p+1})) \cdots \mathsf{S}(\pi(s_m)))\ .$$

From the induction hypothesis, we infer $\mathsf{Q}(\pi(s_i)) \mathrel{\underset{\sim}{\blacktriangleright}}_k \mathsf{Q}(\pi(t_i))$. It is not difficult to argue that $\mathsf{S}(\pi(s)) \mathrel{\underset{\sim}{\blacktriangleright}}_k \mathsf{S}(\pi(t))$ follows by induction hypothesis: From $\mathsf{Q}(\pi(s_i)) \sim \mathsf{Q}(\pi(t_j))$ we see that $\mathsf{S}(\pi(s)) \sim \mathsf{S}(\pi(t))$ follows. From $\mathsf{Q}(\pi(s_i)) \blacktriangleright_k \mathsf{Q}(\pi(t_j))$ we infer $\mathsf{S}(\pi(s)) \blacktriangleright_k \mathsf{S}(\pi(t))$ with either one or two applications of Definition 3.4, depending on whether $i \in \mathsf{safe}(f)$ or not. Thus, in order to conclude the lemma, we need to show

$$\mathsf{N}(\pi(s)) = (\mathsf{S}(\pi(s)))\ @\ \mathsf{BN}(\pi(s)) \mathrel{\underset{\sim}{\blacktriangleright}}_k (\mathsf{S}(\pi(t)))\ @\ \mathsf{BN}(\pi(t)) = \mathsf{N}(\pi(t))$$

and $\|\pi(s)\| \geqslant \|\pi(t)\|$ when $\mathsf{S}(\pi(s)) \sim \mathsf{S}(\pi(t))$.

First, assume $\mathsf{S}(\pi(s)) \sim \mathsf{S}(\pi(t))$. Let $\mathsf{Q} = \mathsf{S}$ if $i \in \mathsf{safe}(f)$ and $\mathsf{Q} = \mathsf{N}$ otherwise. As $\mathsf{S}(\pi(t))$ can be obtained from $\mathsf{S}(\pi(s))$ by replacing $\mathsf{Q}(\pi(s_i))$ with $\mathsf{Q}(\pi(t_i))$ we conclude $\mathsf{Q}(\pi(s_i)) \sim \mathsf{Q}(\pi(t_i))$ and in particular $\mathsf{S}(\pi(s_i)) \sim \mathsf{S}(\pi(t_i))$ independent on Q. Thus by the induction hypothesis $\|\pi(s_i)\| \geqslant \|\pi(t_i)\|$ and so we derive $\|\pi(s)\| \geqslant \|\pi(t)\|$. Clearly, from this also $\mathsf{N}(\pi(s)) \mathrel{\underset{\sim}{\blacktriangleright}}_k \mathsf{N}(\pi(t))$ follows.

On the other hand, assume $\mathsf{S}(\pi(s)) \blacktriangleright_k \mathsf{S}(\pi(t))$. In order to conclude the lemma, we show $\mathsf{N}(\pi(s)) \blacktriangleright_k \mathsf{N}(\pi(t))$. By the assumption and clause (5) from Definition 3.4 it suffices to prove

$$\mathrm{width}(\mathsf{N}(\pi(s))) + k > \mathrm{width}(\mathsf{N}(\pi(t)))\ .$$

From the induction hypothesis we infer $\mathsf{N}(\pi(s_i)) \mathrel{\underset{\sim}{\blacktriangleright}}_k \mathsf{N}(\pi(t_i))$, so either $\mathsf{N}(\pi(s_i)) \sim \mathsf{N}(\pi(t_i))$ or $\mathsf{N}(\pi(s_i)) \blacktriangleright_k \mathsf{N}(\pi(t_i))$. For the former case we see $\mathrm{width}(\mathsf{N}(\pi(s_i))) = \mathrm{width}(\mathsf{N}(\pi(t_i)))$, and for the latter $\mathrm{width}(\mathsf{N}(\pi(s_i))) + k > \mathrm{width}(\mathsf{N}(\pi(t_i)))$ follows from Lemma 3.6. Observe that $k \geqslant 1$. With the help of Lemma 5.14 the above inequalities translate to $\|\pi(s_i)\| + k > \|\pi(t_i)\|$. Thus $\|\pi(s)\| + k > \|\pi(t)\|$, and with Lemma 5.14 we conclude $\mathrm{width}(\mathsf{N}(\pi(s))) + k > \mathrm{width}(\mathsf{N}(\pi(t)))$ as desired.

□

Lemma 5.30. *Let $\mathcal{R}$ be a TRS compatible with $\gtrsim^{\pi}_{\mathsf{pop*}}$, i.e $\mathcal{R} \subseteq \gtrsim^{\pi}_{\mathsf{pop*}}$ holds. Define $k = 3 \cdot \max\{\|\pi(r)\| \mid \ell \to r \in \mathcal{R}\}$, and let $\mathsf{Q} \in \{\mathsf{S}, \mathsf{N}\}$. Then*

$$s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t \Longrightarrow \mathsf{Q}(\pi(s)) \mathrel{\underset{\sim}{\blacktriangleright}}_k \mathsf{Q}(\pi(t))\ .$$

Proof. By Lemma 5.29, it suffices to consider the additional case $s = C[\ell\sigma] \xrightarrow{\mathsf{v}}_{\mathcal{R}} C[r\sigma] = t$ with $\ell \overset{\mathsf{s}}{\sim}_{\pi} r$. From $\ell \overset{\mathsf{s}}{\sim}_{\pi} r$, as $\overset{\mathsf{s}}{\sim}_{\pi}$ is closed under context and substitutions, we derive $s \overset{\mathsf{s}}{\sim} t$ and thus $\mathsf{Q}(\pi(s)) \sim \mathsf{Q}(\pi(t))$ follows by Lemma 5.24. From $\sim \subseteq \underset{\sim}{\blacktriangleright}$ we conclude the claim. □

Theorem 5.31. *Let $\mathcal{R}$ and $\mathcal{S}$ be two constructor TRSs over the signature $\mathcal{F}$. If $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop*}}$ and $\mathcal{S} \subseteq \gtrsim^{\pi}_{\mathsf{pop*}}$ then there exists a polynomial p with*

$$\mathrm{dl}(t, \xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{R}/\mathcal{S}}) \leqslant p(|t|)$$

for any constructor-based term $t \in \mathcal{T}_{\mathsf{b}}$. The polynomial p depends only on $\mathcal{R}$ and $\mathcal{S}$.

Proof. Let $k = 3 \cdot \max\{\|\pi(r)\| \mid \ell \to r \in \mathcal{R}\}$. From the assumption $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop}*}$ and $\mathcal{S} \subseteq \gtrsim^{\pi}_{\mathsf{pop}*}$, we derive

$$s \xrightarrow{\mathsf{v}}{}^{\varepsilon}_{\mathcal{R}} t \Longrightarrow \mathsf{N}(\pi(s)) \blacktriangleright_k \mathsf{N}(\pi(t)) \text{ and } s \xrightarrow{\mathsf{v}}_{\mathcal{S}} t \Longrightarrow \mathsf{N}(\pi(s)) \underset{\sim}{\blacktriangleright}_k \mathsf{N}(\pi(t))$$

from Lemma 5.28 and Lemma 5.30 respectively. As $\mathcal{R}$ and $\mathcal{S}$ are constructor TRSs, the above implications yield

$$\mathrm{dl}(t, \xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{R}/\mathcal{S}}) \leqslant \mathsf{G}_k(\mathsf{N}(\pi(t))) + c$$

for $t \in \mathcal{T}_{\mathsf{b}}$ and fixed constant c according to Lemma 5.23 (again notice that as mentioned earlier, an embedding of top-steps $s \xrightarrow{\mathsf{v}}{}^{\varepsilon}_{\mathcal{R}} t$ is sufficient for the application of Lemma 5.23). With the help of Lemma 5.15 we derive constants $d_1, d_2 \in \mathbb{N}$ such that

$$\mathsf{G}_k(\mathsf{N}(\pi(t))) \leqslant d_1 \cdot |\pi(t)|^{d_2} .$$

The constants d_1 and d_2 depend only on k and the signature $\mathcal{F}$. As $|\pi(t)| \leqslant |t|$, we conclude the theorem. □

5.3 The Embedding of Arbitrary Steps

Up to now we have only seen howto embed top-steps into the strict order $\blacktriangleright_k$, whereas we have embedded steps below the root into the extension $\underset{\sim}{\blacktriangleright}_k$. Below, we present an embedding of $\xrightarrow{\mathsf{v}}_{\mathcal{R}}$ into $\blacktriangleright_k$ from the assumption $\mathcal{R} \subseteq >_{\mathsf{pop}*}$. We specialize Lemma 5.28 and Lemma 5.29 to the polynomial path order without argument filterings. A proof of these lemmas can be easily obtained by restricting the considerations in the proceeding section to those cases where π does not collapse symbols.

Lemma 5.32. *Let $\mathcal{R}$ be a TRS compatible with $>_{\mathsf{pop}*}$, i.e $\mathcal{R} \subseteq >_{\mathsf{pop}*}$ holds. Define $k = 3 \cdot \max\{\|r\| \mid \ell \to r \in \mathcal{R}\}$, and let $\mathsf{Q} \in \{\mathsf{S}, \mathsf{N}\}$. Then*

$$s \xrightarrow{\mathsf{v}}{}^{\varepsilon}_{\mathcal{R}} t \Longrightarrow \mathsf{Q}(s) \blacktriangleright_k \mathsf{Q}(t) .$$

Lemma 5.33. *Let $\mathcal{R}$ be a TRS compatible with $>_{\mathsf{pop}*}$, i.e $\mathcal{R} \subseteq >_{\mathsf{pop}*}$ holds. Define $k = 3 \cdot \max\{\|r\| \mid \ell \to r \in \mathcal{R}\}$, and let $\mathsf{Q} \in \{\mathsf{S}, \mathsf{N}\}$. Then*

$$s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t \Longrightarrow \mathsf{Q}(s) \blacktriangleright_k \mathsf{Q}(t) .$$

Theorem 5.34. *Let $\mathcal{R}$ and $\mathcal{S}$ be two constructor TRSs over the signature $\mathcal{F}$. If $\mathcal{R} \subseteq >_{\mathsf{pop}*}$ and $\mathcal{S} \subseteq \gtrsim_{\mathsf{pop}*}$ then there exists a polynomial p with*

$$\mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}}) \leqslant p(|t|)$$

for any constructor-based term $t \in \mathcal{T}_{\mathsf{b}}$. The polynomial p depends only on $\mathcal{R}$ and $\mathcal{S}$.

Corollary 5.35. *Let $\mathcal{R}$ be a constructor TRS compatible with $>_{\mathsf{pop}*}$, i.e. $\mathcal{R} \subseteq >_{\mathsf{pop}*}$ holds. There exists a polynomial $p \colon \mathbb{N} \to \mathbb{N}$ such that*

$$\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}(n) \leqslant p(n)$$

for all $n \in \mathbb{N}$. The polynomial p depends only on the cardinality of $\mathcal{F}$ and the sizes of the right-hand sides in $\mathcal{R}$.

Proof. As $\xrightarrow{\mathsf{i}}_{\mathcal{R}} = \xrightarrow{\mathsf{i}}_{\mathcal{R}/\varnothing}$, the claim becomes immediate from the definition of $\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}$ and Theorem 5.34. □

5.4 Controlling the Growth Rate of Term-Sizes

Finally, we show that under the assumption that the signature is simple, compatibility of $\mathcal{R}$ with a polynomial path order certifies that the size of each term in a derivation is polynomially bounded with respect to the size of the starting term. For that, we employ that a simple signature certifies the following relationship between the Buchholz-norm and size for values:

Proposition 5.36. *Let $\mathcal{C}$ be a set of constructors from a simple signature. There exists a constant $d \in \mathbb{N}$ such that for each term $t \in \mathsf{Val}$ whose rank is r, $|t| \leqslant d^r \cdot \|t\|^{r+1}$.*

Proof. The easy proof can be found in [40, Proposition 17]. □

We now proof the claim employed in Theorem 4.15.

Lemma 5.37. *Let $\mathcal{R}$ be a constructor TRS based on a simple signature such that $\mathcal{R}$ is compatible with an instance $>_{\mathsf{pop*}}$, i.e. the inclusions $\mathcal{R} \subseteq >_{\mathsf{pop*}}$ holds. Let $t \in \mathcal{T}_{\mathsf{b}}$. There exists a polynomial q such that for all s with $t \xrightarrow{\mathsf{i}}^*_{\mathcal{R}} s$ it holds that $|s| \leqslant q(|t|)$.*

Proof. Assume $t \xrightarrow{\mathsf{i}}^*_{\mathcal{R}} s$. As observed in the proof of Theorem 5.34 the sequence from t to s translates to a descent with respect to $\blacktriangleright_k$ for some fixed k. That is, $\mathsf{N}(t) \blacktriangleright_k^* \mathsf{N}(s)$ holds (again we employ $\rightarrow_{\mathcal{R}} = \rightarrow_{\mathcal{R}/\varnothing}$). In particular, this implies $\mathsf{G}_k(\mathsf{N}(t)) \geqslant \mathsf{G}_k(\mathsf{N}(s))$. One can show that for all terms t,

$$\mathsf{G}_k(\mathsf{N}(t)) + 1 \geqslant |\mathsf{N}(t)| \text{ and} \tag{i}$$

$$c|\mathsf{N}(t)|^d \geqslant |t| \tag{ii}$$

for some constants $0 < c, d \in \mathbb{N}$. These properties are simple to verify: property (i) follows from induction on the structure of terms where we employ for the inductive step that $f(t_1, \dots, t_n) \blacktriangleright_k (t_1 \cdots t_n)$ and $\mathsf{G}_k((t_1 \cdots t_n)) = \Sigma_{i=1}^n \mathsf{G}_k(t_i) + n$ (cf. Lemma 3.8). For property (ii), one shows by a straight forward induction on t that $e^r \cdot (|\mathsf{S}(t)| \cdot \|t\|^{r+1}) \geqslant |t|$ where r is the maximal rank of a symbol in $\mathcal{C}$ and e is as given from Proposition 5.36. As $|\mathsf{N}(t)| > |\mathsf{S}(t)|$ and $|\mathsf{N}(t)| > \|t\|$, property (ii) follows. From the assumption $t \in \mathcal{T}_{\mathsf{b}}$, by Lemma 5.15 there exists a monotone polynomial p such that $\mathsf{G}_k(\mathsf{N}(t)) \leqslant p(|t|)$. Putting things together, we conclude

$$c \cdot (p(|t|) + 1)^d \geqslant c \cdot (\mathsf{G}_k(\mathsf{N}(t)) + 1)^d \geqslant c \cdot (\mathsf{G}_k(\mathsf{N}(s)) + 1)^d \geqslant c \cdot |\mathsf{N}(s)|^d \geqslant |s| \ .$$

The lemma follows by setting $q(m) = c \cdot (p(m) + 1)^d + 1$. □

6 Transformation Techniques

Early efforts to automatically establish termination of rewrite systems were mainly centered around direct methods, for instance the use of recursive path orders or polynomial interpretations. In recent years, the attention shifted towards transformation techniques. In the context of termination analysis, the employed transformation technique has to preserve non-termination at least. In the context of complexity analysis, complexity certificates on the transformed problem need to be transferable back to the original problem additionally.

The *dependency pair method* [5] and *semantic labeling* [58] are in particular popular instances of transformation techniques. Both methods significantly increase the possibility to verify termination. In this Chapter, we show that suitable adaptations can be employed together with polynomial path orders for an analysis of the runtime-complexity.

6.1 POP* and Dependency Pairs

The *dependency pair* method [5] is a well known transformation technique established for the termination analysis of rewrite systems. It is extremely powerful, and most decent automatic termination provers rely on the dependency pair method nowadays. Let $\mathcal{R}$ be a TRS. In order to study the termination behavior of $\mathcal{R}$, a set of *dependency pairs* $\mathsf{DP}(\mathcal{R})$ is extracted. Termination of $\mathcal{R}$ is given when $\rightarrow^{\varepsilon}_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}$ is well-founded. Recent work by Hirokawa and Moser [29] has shown that a similar observation carries over to the analysis of the runtime-complexity of $\mathcal{R}$. We first recall the central concepts from [29].

Definition 6.1. Let $\mathcal{F}$ be a signature partitioned into defined symbols $\mathcal{D}$ and constructors $\mathcal{C}$. With $\mathcal{D}^\sharp$ we denote the set of (*marked*) defined symbols $\mathcal{D} \cup \{f^\sharp | f \in \mathcal{D}\}$. Here $f^\sharp$ denotes a fresh function symbol with the same arity as f. For a term $t = f(t_1, \dots, t_n)$ we write $t^\sharp$ to denote the term $t = f^\sharp(t_1, \dots, t_n)$. Furthermore, we define $\mathcal{F}^\sharp = \mathcal{D}^\sharp \cup \mathcal{C}$.

We abbreviate the set of terms $\mathcal{T}(\mathcal{F}^\sharp, \mathcal{V})$ by $\mathcal{T}^\sharp$, likewise we write $\mathcal{T}_{\mathsf{b}}^{\#}$ for the set of constructor-based terms $\mathcal{T}_{\mathsf{b}}(\mathcal{F}^\sharp, \mathcal{V})$ with respect to the set of defined symbols $\mathcal{D}^\sharp$ and constructors $\mathcal{C}$.

The notion of dependency pairs is often to weak for a complexity analysis. To deal with that, the so called *weak dependency pairs* or *weak innermost dependency pairs* respectively are introduced:

Definition 6.2. Let $\mathcal{R}$ be a TRS. Let $\mathrm{COM}(t_1, \dots, t_n) = t_1$ if $n = 1$ and $\mathrm{COM}(t_1, \dots, t_n) = \mathsf{c}(t_1, \dots, t_n)$. Here c is a fresh constructor symbol, called a *compound symbol* and collected in $\mathcal{C}_{\mathrm{COM}}$.

If $\ell \to r \in \mathcal{R}$ and $r = C\langle u_1, \dots, u_n\rangle_{\mathcal{D}\cup\mathcal{V}}$ then the rule $\ell^\sharp \to \mathrm{COM}(u_1^\sharp, \dots, u_n^\sharp)$ is called a *weak dependency pair* of $\mathcal{R}$. The set of all weak dependency pairs is denoted by $\mathsf{WDP}(\mathcal{R})$. Furthermore, if $\ell \to r \in \mathcal{R}$ and $r = C\langle u_1, \dots, u_n\rangle_{\mathcal{D}}$ then the rewrite rule $\ell^\sharp \to \mathrm{COM}(u_1^\sharp, \dots, u_n^\sharp)$ 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 6.3. Reconsider the rewrite system $\mathcal{R}_{\mathsf{bits}}$ from Example 4.9. The set of weak and weak innermost dependency pairs is given as follows:

$$\begin{aligned}
\mathsf{half}^\sharp(0) &\to \mathsf{c}_1 & \mathsf{bits}^\sharp(0) &\to \mathsf{c}_3 \\
\mathsf{half}^\sharp(\mathsf{s}(0)) &\to \mathsf{c}_2 & \mathsf{bits}^\sharp(\mathsf{s}(0)) &\to \mathsf{c}_4 \\
\mathsf{half}^\sharp(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{half}^\sharp(x) & \mathsf{bits}^\sharp(\mathsf{s}(\mathsf{s}(x))) &\to \mathsf{bits}^\sharp(\mathsf{s}(\mathsf{half}(x)))
\end{aligned}$$

Any sequence of $\to_{\mathcal{R}}$ steps starting from a term t can be simulated as a sequence of $\to_{\mathsf{WDP}(\mathcal{R})\cup\mathcal{R}}$ steps starting from the marked term $t^\sharp$. Furthermore, for innermost derivations we are allowed to replace $\mathsf{WDP}(\mathcal{R})$ by $\mathsf{WIDP}(\mathcal{R})$. Let $\mathcal{P}$ denote the set of weak or weak innermost dependency pairs, and let $t^\sharp$ denote the marked version of a constructor-based term t. In a $\to_{\mathcal{P}\cup\mathcal{R}}$ derivation, usually not every rule from $\mathcal{R}$ can be triggered, independent on t. In fact, only the *usable rules* $\mathcal{U}(\mathcal{P}) \subseteq \mathcal{R}$ can be triggered. Often, these are a strict subset of $\mathcal{R}$.

Definition 6.4. We write $f \rhd_{\mathrm{d}} g$ if there exists a rewrite rule $\ell \to r \in \mathcal{R}$ such that $f = \mathrm{root}(\ell)$ and g is a defined symbol in $\mathsf{Fun}(r)$. For a set $\mathcal{G}$ of defined symbols we denote by $\mathcal{R}{\restriction}\mathcal{G}$ the set of rewrite rules $\ell \to r \in \mathcal{R}$ with $\mathrm{root}(\ell) \in \mathcal{G}$. The set $\mathcal{U}(t)$ of usable rules of a term t is defined as

$$\mathcal{R}{\restriction}\{g \mid f \rhd_{\mathrm{d}}^* g \text{ for some } f \in \mathsf{Fun}(t)\}\ .$$

Finally, if $\mathcal{P}$ is a set of weak or weak innermost dependency pairs then

$$\mathcal{U}(\mathcal{P}) = \bigcup_{\ell\to r\in\mathcal{P}} \mathcal{U}(r)\ .$$

Example 6.5. Reconsider the rewrite system $\mathcal{R}_{\mathsf{bits}}$ from Example 4.9 together with the weak innermost dependency pairs as given in Example 6.3. The set of usable rules $\mathcal{U}(\mathsf{WIDP}(\mathcal{R}))$ consists of the following three rules

$$\mathsf{half}(0) \to 0 \qquad \mathsf{half}(\mathsf{s}(0)) \to 0 \qquad \mathsf{half}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{half}(x)\ .$$

By the above observations, the following proposition is immediate:

Proposition 6.6 ([29])**.** *Let $\mathcal{R}$ be a terminating TRS, and let $\mathcal{P}$ denote the set of weak or weak innermost dependency pairs. Then*

$$\mathrm{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \leqslant \mathrm{dl}(t^\sharp, \xrightarrow{\mathrm{i}}_{\mathcal{P}\cup\mathcal{U}(\mathcal{P})})$$

for all constructor-based terms $t \in \mathcal{T}_{\mathsf{b}}$.

As a consequence of Theorem 4.10 and Proposition 6.6, from compatibility of $\mathcal{P} \cup \mathcal{U}(\mathcal{P})$ with a polynomial path order $>_{\mathsf{pop*}}$ we derive a polynomial bound on the innermost runtime-complexity of $\mathcal{R}$. Ultimately, we want to estimate $\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}$ in terms of $\mathcal{P}$ steps relative to the usable rules $\mathcal{U}(\mathcal{P})$. In [29] it is shown that at least under certain conditions, this is indeed possible:

Proposition 6.7 ([29])**.** *Let $\mathcal{R}$ be a terminating TRS, and let $\mathcal{P}$ denote the set of weak or weak innermost dependency pairs. Assume $\mathcal{P}$ is non-duplicating, and suppose $\mathcal{U}(\mathcal{P}) \subseteq >_{\mathcal{A}}$ for some strongly linear interpretation $\mathcal{A}$. Then there exist constants $K, L \leqslant 0$ (depending on $\mathcal{R}$ and $\mathcal{A}$ only) such that*

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

for all constructor-based terms $t \in \mathcal{T}_{\mathsf{b}}$.

Consider a derivation starting from the constructor-based term $t^\sharp$:

$$t^\sharp = t_0^\sharp \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t_0^\sharp \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t_1^\sharp \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} \cdots \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t_\ell^\sharp$$

In the previous chapter, we have seen that polynomial path orders can be employed for the analysis of relative steps. For the case when all compound symbols in $\mathcal{P}$ are nullary, it is easy to see that all steps due to $\mathcal{P}$ happen at the root, that is, $t_i^\sharp \xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t_{i+1}^\sharp$ holds for all i. By Theorem 5.31, the inclusions $\mathcal{P} \subseteq >^{\pi}_{\mathsf{pop*}}$ and $\mathcal{U} \subseteq \gtrsim^{\pi}_{\mathsf{pop*}}$ certify a polynomial bound on $\mathrm{dl}(t^\sharp, \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})})$ in the size of $t^\sharp$. Proposition 6.7 translates this bound to a polynomial bound on $\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}$.

On the other hand, when $\mathcal{P}$ contains n-ary compound symbols with $n \geqslant 1$, then steps due to $\mathcal{P}$ need not be top-steps anymore. That is, rather than $t_i^\sharp \xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t_{i+1}^\sharp$ we have

$$t_i^\sharp = C[u_1^\sharp, \dots, u_i^\sharp, \dots, u_n^\sharp] \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} C[v_1^\sharp, \dots, \mathrm{COM}(w_1^\sharp, \dots, w_m^\sharp), \dots, v_n^\sharp] = t_{i+1}^\sharp$$

where $u_i^\sharp \xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} \mathrm{COM}(w_1^\sharp, \dots, w_m^\sharp)$ and $u_j^\sharp \xrightarrow{\mathcal{P} \cup \mathcal{U}(\mathcal{P})}_{\mathcal{U}(\mathcal{P})} v_j^\sharp$ for $j \neq i$. In particular, the context C is solely build from compound symbols. In order to handle steps of the above shape, we require that the argument filtering π is *safe*, that is it does not erase redexes with respect to the dependency pairs $\mathcal{P}$:

Definition 6.8. An argument filtering π is called *safe* if $\pi(\mathsf{c}) = [1, \dots, n]$ for each n-ary compound symbol c.

Unfortunately, Theorem 5.31 still does not extend to $\xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$ in general. Remember that in the reasoning carried out, we essentially rely on the embedding of $\xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$ into $\blacktriangleright_k^+$ via the predicative interpretation N. However, this embedding fails, even for safe argument filterings. The following example clarifies the situation:

Example 6.9. Consider the TRS given by the following rewrite rules

$$\mathsf{f}(0) \to \mathsf{nil} \qquad \mathsf{p}(\mathsf{s}(x)) \to x \qquad \mathsf{f}(\mathsf{s}(x)) \to \mathsf{cons}(\mathsf{p}(\mathsf{s}(x)), \mathsf{f}(x))$$

The set of weak dependency pairs $\mathcal{P}$ is given by

$$\mathsf{f}^\sharp(0) \to \mathsf{c}_1 \qquad \mathsf{p}^\sharp(\mathsf{s}(x)) \to x \qquad \mathsf{f}^\sharp(\mathsf{s}(x)) \to \mathsf{c}_2(\mathsf{p}^\sharp(\mathsf{s}(x)), \mathsf{f}^\sharp(x)) \ ,$$

and $\mathcal{U}(\mathcal{P}) = \varnothing$. Then $\xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} \; = \xrightarrow{\mathsf{i}}_{\mathcal{P}}$ admits the derivation

$$\mathsf{f}^\sharp(\mathsf{s}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{P}} \mathsf{c}_2(\mathsf{p}^\sharp(\mathsf{s}(0)), \mathsf{f}^\sharp(0)) \xrightarrow{\mathsf{i}}_{\mathcal{P}} \mathsf{c}_2(\mathsf{p}^\sharp(\mathsf{s}(0)), c_1) \xrightarrow{\mathsf{i}}_{\mathcal{P}} \mathsf{c}_2(0, c_1) \ .$$

Let π denote the safe argument filtering that collapses $\mathsf{p}^\sharp$, but else maps each function symbol to the complete list of argument positions. Then it can be shown that $\mathcal{P} \subseteq >^{\pi}_{\mathsf{pop}*}$ for some precedence and safe mapping. On the other hand, we cannot embed the final step $\mathsf{c}_2(\mathsf{p}^\sharp(\mathsf{s}(0)), c_1) \xrightarrow{\mathsf{i}}_{\mathcal{P}} \mathsf{c}_2(0, c_1)$:

$$\mathsf{N}(\pi(\mathsf{c}_2(\mathsf{p}^\sharp(\mathsf{s}(0)), c_1))) = (\varnothing \ \ \mathsf{s} \ \ \mathsf{s} \ \ \mathsf{s}) = \mathsf{N}(\pi(c_1(0, c_2))) \ .$$

The problem above is introduced from interpreting the redex $\mathsf{p}^\sharp(\mathsf{s}(0))$ via the interpretation S rather than N (remember that all arguments positions of constructors are safe). In the previous chapter, we have seen that $\mathcal{P} \subseteq >^{\pi}_{\mathsf{pop}*}$ does not necessarily guarantee an embedding of innermost top-steps into $\blacktriangleright_k$ via the interpretation S (cf. Lemma 5.28). To circumvent this problem, we slightly adapt the predicative interpretation, so that redexes with respect to $\mathcal{P}$ are interpreted by N. For that, we interpret compound symbols as sequences:

Definition 6.10. The *extended predicative interpretation*

$$\mathsf{N}^{\mathsf{s}} : \mathcal{T}(\mathcal{F} \cup \mathcal{C}_{\mathrm{COM}}, \mathcal{V}) \to \mathcal{S}\mathrm{eq}(\mathcal{F}^{\mathsf{n}} \cup \{\mathsf{s}\}, \mathcal{V})$$

is defined as follows:

$$\mathsf{N}^{\mathsf{s}}(t) = \begin{cases} (\mathsf{N}^{\mathsf{s}}(t_1) \ \cdots \ \mathsf{N}^{\mathsf{s}}(t_n)) & \text{if } t = c(t_1, \dots, t_n) \text{ and } c \in \mathcal{C}_{\mathrm{COM}} \\ (\mathsf{N}(t)) & \text{otherwise.} \end{cases}$$

We now proof an embedding of $\rightarrow_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$ into $\blacktriangleright_k^{+}$ for some fixed k via the interpretation N^{s}. For that, we proceed exactly as in Chapter 5: we first show an embedding of top-steps $\xrightarrow{\mathsf{v}}{}^{\varepsilon}_{\mathcal{R}}$. Afterwards, we lift the embedding to contexts.

Lemma 6.11. *Let $\mathcal{R}$ be a TRS, and let $\mathcal{P}$ denote the set of weak or weak innermost dependency pairs. Suppose $\mathcal{P}$ is compatible with $>^{\pi}_{\mathsf{pop}*}$, i.e. suppose $\mathcal{P} \subseteq >^{\pi}_{\mathsf{pop}*}$ holds. Define $k = 3 \cdot \max\{\|\pi(r)\| \mid \ell \to r \in \mathcal{P}\}$. Then*

$$s \xrightarrow{\mathsf{v}}{}^{\varepsilon}_{\mathcal{P}} t \Longrightarrow \mathsf{N}^{\mathsf{s}}(\pi(s)) \blacktriangleright_k \mathsf{N}^{\mathsf{s}}(\pi(t)) \ .$$

Proof. Assume $s \xrightarrow{\mathsf{v}}{}^{\varepsilon}_{\mathcal{P}} t$ due to a rule $f^\sharp(\ell_1, \dots, \ell_n) \to \mathrm{COM}(r_1^\sharp, \dots, r_m^\sharp) \in \mathcal{P}$. Hence $s = f^\sharp(\ell_1\sigma, \dots, \ell_n\sigma)$ with $\ell_i\sigma \in \mathsf{Val}$ for all $i \in \{1, \dots, n\}$ and $t = \mathrm{COM}(r_1^\sharp\sigma, \dots, r_m^\sharp\sigma)$. From the shape of s we conclude $\mathsf{N}^{\mathsf{s}}(\pi(s)) = (\mathsf{N}(\pi(s)))$. We continue by case analysis on the shape of $r = \mathrm{COM}(r_1^\sharp, \dots, r_m^\sharp)$. When $\mathrm{root}(r) \notin \mathcal{C}_{\mathrm{COM}}$ then $\mathsf{N}^{\mathsf{s}}(\pi(t)) = (\mathsf{N}(\pi(t)))$. From Lemma 5.28 we see $\mathsf{N}(\pi(s)) \blacktriangleright_k \mathsf{N}(\pi(t))$, and by one application of clause (5) from Definition 3.4 the lemma follows. Finally, assume $r = \mathsf{c}(r_1^\sharp, \dots, r_m^\sharp)$ where $\mathsf{c} \in \mathcal{C}_{\mathrm{COM}}$ is a compound symbol. With the help of Lemma 5.12 we see $s >^{\pi}_{\mathsf{pop}*} r_j^\sharp$ for $j \in \{1, \dots, m\}$, and by inspecting Lemma 5.28 together with clause (3) from Definition 3.4 it is easy to conclude $\mathsf{N}(\pi(s)) \blacktriangleright_k (\mathsf{N}(\pi(r_j^\sharp\sigma))) = \mathsf{N}^\mathsf{s}(\pi(r_j^\sharp\sigma))$ for $j \in \{1,\dots,m\}$. From clause (5) of Definition 3.4 we derive

$$\mathsf{N}^\mathsf{s}(\pi(s)) = (\mathsf{N}(\pi(s))) \blacktriangleright_k (\mathsf{N}^\mathsf{s}(\pi(r_1^\sharp\sigma)) \ \dots\ \mathsf{N}^\mathsf{s}(\pi(r_m^\sharp\sigma))) = \mathsf{N}^\mathsf{s}(\pi(t))$$

as desired. For the above, observe $k > m$. □

Next we close the embedding under contexts build from compound symbols.

Lemma 6.12. *Let $\mathcal{R}$ be a TRS, and let $\mathcal{P}$ denote the set of weak or weak innermost dependency pairs. Suppose $\mathcal{P}$ is compatible with $>_{\mathsf{pop}*}^\pi$, i.e. suppose $\mathcal{P} \subseteq >_{\mathsf{pop}*}^\pi$ holds. Define $k = 3 \cdot \max\{\|\pi(r)\| \mid \ell \to r \in \mathcal{P}\}$. Then*

$$s \xrightarrow{\mathsf{v}}{}_{\mathcal{P}}^{\varepsilon} t \Longrightarrow \mathsf{N}^\mathsf{s}(\pi(C[u_1,\dots,s,\dots,u_n])) \blacktriangleright_k \mathsf{N}^\mathsf{s}(\pi(C[u_1,\dots,t,\dots,u_n]))$$

for every safe argument filtering π and context $C \in \mathcal{T}(\mathcal{C}_{\mathrm{COM}} \cup \{\square\}, \mathcal{V})$ build from compound symbols.

Proof. We prove the lemma by induction on C. For the base case, that is when $C = \square$, we conclude the claim with the help of Lemma 6.11. For the inductive step, suppose

$$C[u_1,\dots,s,\dots,u_n] = \mathsf{c}(v_1,\dots,C'[u_{i_1},\dots,s,\dots,u_{i_p}],\dots,v_m)$$

with $\mathsf{c} \in \mathcal{C}_{\mathrm{COM}}$, $C' \in \mathcal{T}(\mathcal{C}_{\mathrm{COM}} \cup \{\square\}, \mathcal{V})$ and $1 \leqslant i_1 \leqslant \cdots \leqslant i_p \leqslant n$. The induction hypothesis reveals

$$\mathsf{N}^\mathsf{s}(\pi(C'[u_{i_1},\dots,s,\dots,u_{i_p}])) \blacktriangleright_k \mathsf{N}^\mathsf{s}(\pi(C'[u_{i_1},\dots,t,\dots,u_{i_p}])) \ .$$

It is easy to see that from the induction hypothesis and one application of clause (5) from Definition 3.4,

$$\begin{aligned}&(\mathsf{N}^\mathsf{s}(\pi(v_1)) \ \cdots\ \mathsf{N}^\mathsf{s}(\pi(C'[u_{i_1},\dots,s,\dots,u_{i_p}])) \ \cdots\ \mathsf{N}^\mathsf{s}(\pi(v_m)))\\ &\qquad \blacktriangleright_k (\mathsf{N}^\mathsf{s}(\pi(v_1)) \ \cdots\ \mathsf{N}^\mathsf{s}(\pi(C'[u_{i_1},\dots,t,\dots,u_{i_p}])) \ \cdots\ \mathsf{N}^\mathsf{s}(\pi(v_m)))\end{aligned}$$

follows. By definition of N^s this proves the lemma. □

Finally, we need to adapt Lemma 5.30 to the interpretation N^s.

Lemma 6.13. *Let $\mathcal{R}$ be a TRS compatible with $\gtrsim_{\mathsf{pop}*}^\pi$, i.e $\mathcal{R} \subseteq \gtrsim_{\mathsf{pop}*}^\pi$ holds. Define $k = 3 \cdot \max\{\|\pi(r)\| \mid \ell \to r \in \mathcal{R}\}$, and let $\mathsf{Q} \in \{\mathsf{S}, \mathsf{N}\}$. Then*

$$s \xrightarrow{\mathsf{v}}{}_{\mathcal{R}} t \Longrightarrow \mathsf{N}^\mathsf{s}(\pi(s)) \underset{\sim}{\blacktriangleright}_k \mathsf{N}^\mathsf{s}(\pi(t)) \ .$$

Proof. Assume $s \xrightarrow{\mathsf{v}}{}_{\mathcal{R}} t$, and thus $s = C[\ell\sigma]$ and $t = C[r\sigma]$ for some context C, $\ell \to r \in \mathcal{R}$ and substitution $\sigma : \mathcal{V} \to \mathsf{Val}$. We continue with a proof by induction on the context C. Again the base case follows either by Lemma 6.11 or Lemma 5.24 correspondingly. For the inductive step, suppose $C = f(s_1,\dots,\square,\dots,s_n)$. When $f \in \mathcal{C}_{\mathrm{COM}}$, then

$$\begin{aligned}&\mathsf{N}^\mathsf{s}(\pi(s)) = (\mathsf{N}^\mathsf{s}(\pi(s_1)),\dots,\mathsf{N}^\mathsf{s}(\pi(C'[\ell\sigma])),\dots,\mathsf{N}^\mathsf{s}(\pi(s_n))) \text{ and}\\ &\mathsf{N}^\mathsf{s}(\pi(t)) = (\mathsf{N}^\mathsf{s}(\pi(s_1)),\dots,\mathsf{N}^\mathsf{s}(\pi(C'[r\sigma])),\dots,\mathsf{N}^\mathsf{s}(\pi(s_n)))\end{aligned}$$

and the induction hypothesis reveals $\mathsf{N^s}(\pi(C'[\ell\sigma])) \mathrel{\blacktriangleright\!\!\!\!\!\sim}_k \mathsf{N^s}(\pi(C'[r\sigma]))$. For the case when $f \notin \mathcal{C}_{\text{COM}}$ it follows that $\mathsf{N^s}(\pi(s)) = (\mathsf{N}(\pi(s)))$ and $\mathsf{N^s}(\pi(t)) = (\mathsf{N}(\pi(t)))$. From Lemma 5.29 it is easy to infer $\mathsf{N}(\pi(s)) \mathrel{\blacktriangleright\!\!\!\!\!\sim}_k \mathsf{N}(\pi(t))$. Thus for both cases, either one application of Definition 5.7 or Definition 5.9 suffices to conclude the lemma. □

Theorem 6.14. *let $\mathcal{R}$ be a constructor TRS, and let $\mathcal{P}$ denote the set of weak or weak innermost dependency pairs. If $\mathcal{P} \subseteq >^{\pi}_{\mathsf{pop*}}$ and $\mathcal{U}(\mathcal{P}) \subseteq \gtrsim^{\pi}_{\mathsf{pop*}}$ then there exists a polynomial p with*

$$\mathrm{dl}(t^\sharp, \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}) \leqslant p(|t|)$$

for any constructor-based term $t \in \mathcal{T}_{\mathsf{b}}$. The polynomial p depends only on $\mathcal{R}$.

Proof. Define $k = 3 \cdot \max\{\|\pi(r)\| \mid \ell \to r \in \mathcal{R}\}$. Let $t \in \mathcal{T}_{\mathsf{b}}$, and assume a maximal derivation

$$t^\sharp = t^\sharp_0 \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t^\sharp_1 \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} \cdots \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t^\sharp_\ell \ .$$

Consider a relative step $t^\sharp_i \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t^\sharp_{i+1}$ for $i \in \{1, \ldots, \ell - 1\}$. There exists terms $u^\sharp$ and $v^\sharp$ such that

$$t^\sharp_i \xrightarrow{\mathcal{P}\cup\mathcal{U}(\mathcal{P})}{}^{*}_{\mathcal{U}(\mathcal{P})} u^\sharp \xrightarrow{\mathcal{P}\cup\mathcal{U}(\mathcal{P})}{}_{\mathcal{P}} v^\sharp \xrightarrow{\mathcal{P}\cup\mathcal{U}(\mathcal{P})}{}^{*}_{\mathcal{U}(\mathcal{P})} t^\sharp_{i+1} \ .$$

From the shape of $\mathcal{P}$ and the assumption $t \in \mathcal{T}_{\mathsf{b}}$ we conclude that for some context $C \in \mathcal{T}(\mathcal{C}_{\text{COM}} \cup \{\Box\}, \mathcal{V})$, we have $u^\sharp = C[u^\sharp_1, \ldots, u^\sharp_i, \ldots, u^\sharp_p]$ and $v^\sharp = C[u^\sharp_1, \ldots, v^\sharp_i, \ldots, u^\sharp_p]$ with $u^\sharp_i \xrightarrow{\mathsf{v}}{}^{\varepsilon}_{\mathcal{P}} v^\sharp_i$. Observe that

$$3 \cdot \max\{\|\pi(r)\| \mid \ell \to r \in \mathcal{P} \cup \mathcal{U}(\mathcal{P})\} \leqslant k \ ,$$

and moreover since $\mathcal{R}$ is a constructor TRS, $\mathcal{P}$ and $\mathcal{U}(\mathcal{P})$ are constructor TRSs with respect to the signature $\mathcal{F}^\sharp$ partitioned into defined symbols $\mathcal{D}^\sharp$ and constructors $\mathcal{C} \cup \mathcal{C}_{\text{COM}}$. Hence for the assumed maximal derivation, by Lemma 6.12 we infer that whenever a rule from $\mathcal{P}$ is triggered, i.e. $u \xrightarrow{\mathsf{v}}_{\mathcal{P}} v$ for some terms u and v, then $\mathsf{N^s}(\pi(u)) \blacktriangleright_k \mathsf{N^s}(\pi(v))$ follows. Furthermore, $u \xrightarrow{\mathsf{v}}_{\mathcal{U}(\mathcal{P})} v$ implies $\mathsf{N^s}(\pi(u)) \mathrel{\blacktriangleright\!\!\!\!\!\sim}_k \mathsf{N^s}(\pi(v))$ for arbitrary terms u and v according to Lemma 6.13. Notice that Lemma 5.23 can be easily adapted to the interpretation $\mathsf{N^s}$. From this and the above observations, we derive

$$\mathrm{dl}(t^\sharp, \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}) \leqslant \mathsf{G}_k(\mathsf{N^s}(\pi(t^\sharp))) + c$$

for some constant $c \in \mathbb{N}$. Thus in order to conclude the theorem, it suffices to verify that $\mathsf{G}_k(\mathsf{N^s}(\pi(t^\sharp)))$ is polynomially bounded. Observe that $t^\sharp \in \mathcal{T}^{\#}_{\mathsf{b}}$ implies $\mathsf{N^s}(\pi(t^\sharp)) = (\mathsf{N}(\pi(t^\sharp)))$. With the help of Lemma 5.15 we derive constants $d_1, d_2 \in \mathbb{N}$ such that

$$\mathsf{G}_k(\mathsf{N}(\pi(t^\sharp))) \leqslant d_1 \cdot |\pi(t^\sharp)|^{d_2} \ .$$

The constants d_1 and d_2 depend only on k and the signature $\mathcal{F}$. Moreover, $\mathsf{G}_k((s_1 \cdots s_n)) = n + \sum_{i=1}^{n} \mathsf{G}_k(s_i)$ for sequences s_i according to Lemma 3.8, and hence

$$\mathsf{G}_k(\mathsf{N^s}(t^\sharp)) = \mathsf{G}_k(\mathsf{N}(\pi(t^\sharp))) + 1 \leqslant c \cdot |\pi(t^\sharp)|^d + 1 \ .$$

As $|\pi(t^\sharp)| \leqslant |t|$, it is easy to see how to define the polynomial p. We conclude the theorem. □

Corollary 6.15. *Let $\mathcal{R}$ be a terminating constructor TRS, and let $\mathcal{P}$ denote the set of weak or weak innermost dependency pairs. Assume $\mathcal{P}$ is non-duplicating. Let $\gtrsim$ be a precedence,* safe *be a safe mapping and π be a safe argument filtering for $\mathcal{F}^\sharp \cup \mathcal{C}_{\mathrm{COM}}$. If $\mathcal{P} \subseteq >^{\pi}_{\mathsf{pop*}}$ and $\mathcal{U}(\mathcal{P}) \subseteq \gtrsim^{\pi}_{\mathsf{pop*}}$ for the induced order $>^{\pi}_{\mathsf{pop*}}$ then there exists a polynomial p with*

$$\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}(n) \leqslant p(n) \ .$$

The polynomial p depends only on $\mathcal{R}$ and $\mathcal{A}$.

Proof. The corollary is immediate from Proposition 6.7 and Theorem A.20. □

We conclude this section with a final example.

Example 6.16. Reconsider the rewrite system $\mathcal{R}_{\mathsf{bits}}$ from Example 4.9. The set $\mathcal{P}$ of weak innermost dependency pairs is given as follows:

$$\begin{array}{rrcl}
1: & \mathsf{half}^\sharp(0) \to \mathsf{c}_1 & 4: & \mathsf{bits}^\sharp(0) \to \mathsf{c}_3 \\
2: & \mathsf{half}^\sharp(\mathsf{s}(0)) \to \mathsf{c}_2 & 5: & \mathsf{bits}^\sharp(\mathsf{s}(0)) \to \mathsf{c}_4 \\
3: & \mathsf{half}^\sharp(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{half}^\sharp(x) & 6: & \mathsf{bits}^\sharp(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{bits}^\sharp(\mathsf{s}(\mathsf{half}(x)))
\end{array}$$

The set of usable rules $\mathcal{U}(\mathcal{P})$ of consists of the three rewrite rules

$$\begin{array}{rrcl}
7: & \mathsf{half}(0) \to 0 & 9: & \mathsf{half}(\mathsf{s}(0)) \to 0 \\
8: & \mathsf{half}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{half}(x) \ . & &
\end{array}$$

By taking the strongly linear interpretation $\mathcal{A}$ with

$$0_{\mathcal{A}} = 0 \qquad \mathsf{s}_{\mathcal{A}}(x) = x + 1 \qquad \mathsf{half}_{\mathcal{A}}(x) = x + 1$$

we obtain $\mathcal{U}(\mathcal{P}) \subseteq >_{\mathcal{A}}$ and moreover, observe that $\mathcal{P}$ is non-duplicating. Next, we choose the safe argument filtering π defined by

$$\pi(\mathsf{bits}^\sharp) = [1] \qquad \pi(\mathsf{half}) = 1 \qquad \pi(\mathsf{half}^\sharp) = [1] \qquad \pi(\mathsf{s}) = [1] \ .$$

We define the safe mapping safe such that all argument positions of defined symbols are marked normal, that is

$$\mathsf{safe}(\mathsf{bits}^\sharp) = \varnothing \qquad \mathsf{safe}(\mathsf{half}) = \varnothing \qquad \mathsf{safe}(\mathsf{half}^\sharp) = \varnothing \qquad \mathsf{safe}(\mathsf{s}) = \{1\} \ .$$

Furthermore we define the precedence $\gtrsim$ such that $0 \sim \mathsf{c}_1 \sim \mathsf{c}_2 \sim \mathsf{c}_3 \sim \mathsf{c}_4$. For the induced order $>_{\mathsf{pop*}}$ we derive

$$\begin{array}{rrcl}
1: & \mathsf{half}^\sharp(0) >_{\mathsf{pop*}} \mathsf{c}_1 & 4: & \mathsf{bits}^\sharp(0) >_{\mathsf{pop*}} \mathsf{c}_3 \\
2: & \mathsf{half}^\sharp(\mathsf{s}(0)) >_{\mathsf{pop*}} \mathsf{c}_2 & 5: & \mathsf{bits}^\sharp(\mathsf{s}(0)) >_{\mathsf{pop*}} \mathsf{c}_4 \\
3: & \mathsf{half}^\sharp(\mathsf{s}(\mathsf{s}(x))) >_{\mathsf{pop*}} \mathsf{half}^\sharp(x) & 6: & \mathsf{bits}^\sharp(\mathsf{s}(\mathsf{s}(x))) >_{\mathsf{pop*}} \mathsf{bits}^\sharp(\mathsf{s}(x))
\end{array}$$

$$\begin{array}{rrcl}
7: & 0 \gtrsim_{\mathsf{pop*}} 0 & 9: & \mathsf{s}(0) \gtrsim_{\mathsf{pop*}} 0 \\
8: & \mathsf{s}(\mathsf{s}(x)) \gtrsim_{\mathsf{pop*}} x \ , & &
\end{array}$$

that is $\pi(\mathcal{P}) \subseteq >_{\mathsf{pop*}}$ and $\pi(\mathcal{U}(\mathcal{P})) \subseteq \gtrsim_{\mathsf{pop*}}$. According to Lemma 5.11 this establishes $\mathcal{P} \subseteq >^{\pi}_{\mathsf{pop*}}$ and $\mathcal{U}(\mathcal{P}) \subseteq \gtrsim^{\pi}_{\mathsf{pop*}}$. By Corollary 6.15 we conclude a polynomial runtime-complexity of $\mathcal{R}_{\mathsf{bits}}$.

We stress that all steps in the above examples can be performed in a purely mechanical fashion. We investigate the automation in the next Chapter. Before that, we present polynomial path orders together with the semantic labeling transformation.

6.2 POP* and Semantic Labeling

Besides the dependency pair method, semantic labeling [58] is another popular transformation technique. The idea behind semantic labeling is to incorporate semantic information of a TRS $\mathcal{R}$ directly into the rewrite system, yielding a labeled rewrite system $\mathcal{R}_{\mathsf{lab}}$. Termination of $\mathcal{R}_{\mathsf{lab}}$ is equivalent to termination of $\mathcal{R}$. More precise, every step from $\mathcal{R}$ corresponds to a step from $\mathcal{R}_{\mathsf{lab}}$, and vice versa. In this section, we briefly recall semantic labeling as initially proposed by Zantema [58] for the termination analysis of rewrite systems. Furthermore, we will see that semantic labeling, although in a restricted setting, can also be applied for the analysis of the runtime-complexity of $\mathcal{R}$.

In order to incorporate semantic information, one labels the rules of the input system by labeling certain function symbols according to the value of their arguments. The value of an argument is obtained by the interpretation under an $\mathcal{F}$-algebra $\mathcal{A}$. Below, we always write $\mathcal{A}$ for an $\mathcal{F}$-algebra with carrier A.

Definition 6.17. A *labeling* ℓ for $\mathcal{F}$ consists of a set of labels L_f for each n-ary function symbol $f \in \mathcal{F}$ together with labeling function $\ell_f : A^n \to L_f$ whenever $L_f \neq \varnothing$. For a labeling ℓ and signature $\mathcal{F}$, the labeled signature $\mathcal{F}_{\mathsf{lab}}$ is given by the set

$$\mathcal{F}_{\mathsf{lab}} = \{f_a \mid f \in \mathcal{F} \text{ and } a \in L_f\} \ .$$

The labeling ℓ defines the label attached to a root-symbol in a term $f(t_1, \dots, t_n)$ based on the values of the arguments $t_1, \dots, t_n$.

Definition 6.18. For a labeling ℓ and every *assignment* $\alpha : \mathcal{V} \to A$ we define a mapping $\mathsf{lab}_\alpha : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}_{\mathsf{lab}}, \mathcal{V})$ inductively defined by

$$\mathsf{lab}_\alpha(t) = \begin{cases} t & \text{if } t \in \mathcal{V}, \\ f(\mathsf{lab}_\alpha(t_n), \dots, \mathsf{lab}_\alpha(t_n)) & \text{if } t = f(t_1, \dots, t_n) \text{ and } L_f = \varnothing, \\ f_a(\mathsf{lab}_\alpha(t_n), \dots, \mathsf{lab}_\alpha(t_n)) & \text{if } t = f(t_1, \dots, t_n) \text{ and } L_f \neq \varnothing. \end{cases}$$

Here a denotes the *label* $\ell_f([\alpha]_{\mathcal{A}}(t_1), \dots, [\alpha]_{\mathcal{A}}(t_n))$.

In order to obtain from $\mathcal{R}$ a labeled rewrite system $\mathcal{R}_{\mathsf{lab}}$, one labels all rewrite rules from $\mathcal{R}$ according to lab_α for every assignment α.

Definition 6.19. The *labeled rewrite system* $\mathcal{R}_{\mathsf{lab}}$ over the signature $\mathcal{F}_{\mathsf{lab}}$ is given by

$$\mathcal{R}_{\mathsf{lab}} = \{\mathsf{lab}_\alpha(\ell) \to \mathsf{lab}_\alpha(r) \mid \ell \to r \in \mathcal{R} \text{ and assignment } \alpha\}.$$

One of the main results from [58] states that termination of $\mathcal{R}_{\mathsf{lab}}$ is equivalent to termination of $\mathcal{R}$ when the employed $\mathcal{F}$-algebra $\mathcal{A}$ is a *model* of $\mathcal{R}$.

Definition 6.20. An $\mathcal{F}$-algebra $\mathcal{A}$ is a *model* of $\mathcal{R}$ if $[\alpha]_{\mathcal{A}}(\ell) = [\alpha]_{\mathcal{A}}(r)$ holds for every rule $\ell \to r \in \mathcal{R}$ and assignment $\alpha : \mathsf{Var}(\ell) \to A$.

Proposition 6.21. *Let $\mathcal{A}$ be a non-empty model of $\mathcal{R}$ and ℓ a labeling.*

$$s \to_{\mathcal{R}} t \iff \mathsf{lab}_\alpha(s) \to_{\mathcal{R}_{\mathsf{lab}}} \mathsf{lab}_\alpha(t)$$

for all assignment α.

In particular, the above proposition yields $\mathrm{dl}(t, \to_{\mathcal{R}}) = \mathrm{dl}(\mathsf{lab}_\alpha(t), \to_{\mathcal{R}_{\mathsf{lab}}})$ for any assignment α. This suggests that in principle semantic labeling can be employed for a complexity analysis of $\mathcal{R}$. Unfortunately, when $\mathcal{R}_{\mathsf{lab}}$ becomes infinite, we cannot conclude from $\mathcal{R}_{\mathsf{lab}} \subseteq >_{\mathsf{pop*}}$ a polynomial innermost runtime-complexity of $\mathcal{R}$. In this case, Theorem 4.10 is not applicable. Finiteness of the signature of the considered rewrite system is essential for Theorem 4.10, as illustrated in the following example.

Example 6.22. Reconsider the rewrite system $\mathcal{R}_{\mathsf{exp}}$ from Example 4.8, modeling the exponentiation function with obvious exponential innermost runtime-complexity. We choose the model $\mathcal{N}$ over the carrier $\mathbb{N}$ giving the most natural interpretation of the symbols from $\mathcal{R}_{\mathsf{exp}}$:

$$\mathsf{double}_{\mathcal{N}}(x) = 2x \qquad \exp_{\mathcal{N}}(x) = 2^x \qquad \mathsf{s}_{\mathcal{N}}(x) = x + 1 \qquad 0_{\mathcal{N}} = 0 \ .$$

We label the symbol double with its argument and leave all other symbols unlabeled. That is, we define $L_{\mathsf{double}} = \mathbb{N}$ with $\ell_{\mathsf{double}}(x) = x$, and furthermore $L_{\exp} = L_{\mathsf{s}}$. Labeling $\mathcal{R}_{\mathsf{exp}}$ results in the infinite rewrite system

$$\begin{aligned} \mathsf{double}_0(0) &\to 0 & \mathsf{exp}(0) &\to \mathsf{s}(0) \\ \mathsf{double}_{i+1}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{s}(\mathsf{double}_i(x))) & \mathsf{exp}(\mathsf{s}(x)) &\to \mathsf{double}_{2^i}(\mathsf{exp}(x)) \ . \end{aligned}$$

Then we can orient the labeled rewrite system using the polynomial path order induced by the safe mapping

$$\mathsf{safe}(\mathsf{exp}) = \varnothing \qquad \mathsf{safe}(\mathsf{double}_i) = \{1\} \qquad \mathsf{safe}(\mathsf{s}) = \{1\}$$

and the infinite precedence

$$\mathsf{exp} \succ \mathsf{double}_{i+1} \succ \mathsf{double}_i \succ \mathsf{s} \ .$$

The above example illustrates that the polynomial path order cannot certify polytime runtime-complexities in conjunction with (unrestricted) semantic labeling. The problem appears even in a more general setting, and is inherent to simplification orders like MPO. Middeldorp, Ohsaki and Zantema [43] have shown that every rewrite system $\mathcal{R}$ can be labeled in such a way that $\mathcal{R}_{\mathsf{lab}}$ is precedence terminating (it should be noted that they employ *quasi-models* [58] instead of models). It is clear howto define a rewrite system $\mathcal{R}$ admitting a non-primitive recursive derivation length, for instance by encoding the

Ackermann-function. The above observation implies that the labeled system $\mathcal{R}_{\mathsf{lab}}$ is compatible with an instance $>_{\mathsf{mpo}}$, and clearly the result established by Hofbauer [31] does not extend to the system $\mathcal{R}_{\mathsf{lab}}$ and in turn $\mathcal{R}$ respectively.

We circumvent the problem by demanding that the employed model ranges over a finite carrier. Henceforth, we call semantic labeling over a finite carrier *finite semantic labeling.* For this case, $\mathcal{R}_{\mathsf{lab}}$ becomes finite provided that $\mathcal{R}$ is finite. In particular, the signature of $\mathcal{R}_{\mathsf{lab}}$ is finite.

Theorem 6.23. *Let $\mathcal{R}$ be a constructor TRS. Let $\mathcal{R}_{lab}$ denote the labeled rewrite system for some non-empty model $\mathcal{A}$ with finite carrier. If $\mathcal{R}_{lab}$ is compatible with $>_{\mathsf{pop}*}$, i.e $\mathcal{R}_{lab} \subseteq >_{\mathsf{pop}*}$ holds, then there exists a polynomial p such that*

$$\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}(n) \leqslant p(n)$$

for all $n \in \mathbb{N}$. The polynomial p depends only on the cardinality of $\mathcal{F}$ and the sizes of the right-hand sides in $\mathcal{R}$.

Proof. Assume $\mathcal{A}$ is a non-empty model of $\mathcal{R}$ with finite carrier. Observe that every $\rightarrow_{\mathcal{R}}$ step translates to a corresponding $\rightarrow_{\mathcal{R}_{\mathsf{lab}}}$ step according to proposition 6.21. Notice that when $u \in \mathsf{NF}(\mathcal{R})$ then $\mathsf{lab}_{\alpha}(t) \in \mathsf{NF}(\mathcal{R}_{\mathsf{lab}})$ independent on the assignment α. Thus in particular, every $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ step translates to a corresponding $\xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{lab}}}$ step. Hence

$$\mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}}) \leqslant \mathrm{dl}(\mathsf{lab}_{\alpha}(t), \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{lab}}})$$

for any assignment $\alpha \colon \mathsf{Var}(t) \rightarrow A$. As $\mathcal{R}$ is a (finite) constructor TRS, and since $\mathcal{A}$ ranges over a finite carrier, it follows that $\mathcal{R}_{\mathsf{lab}}$ is a finite constructor TRS. Thus from the assumption $\mathcal{R}_{\mathsf{lab}} \subseteq >_{\mathsf{pop}*}$, according to Theorem 4.10, there exists a polynomial $p_{\mathcal{R}_{\mathsf{lab}}}$ such that

$$\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}(n) \leqslant \mathrm{rc}^{\mathsf{i}}_{\mathcal{R}_{\mathsf{lab}}}(n) \leqslant p_{\mathcal{R}_{\mathsf{lab}}}(n) \ .$$

The polynomial $p_{\mathcal{R}_{\mathsf{lab}}}$ depends only on the cardinality of $\mathcal{F}_{\mathsf{lab}}$ and the sizes of the right-hand sides in $\mathcal{R}_{\mathsf{lab}}$. Since labeling does not effect term sizes, and the cardinality of $\mathcal{F}_{\mathsf{lab}}$ is polynomial in the cardinality of $\mathcal{F}$ (depending only on the carrier size) it is easy to see howto define the polynomial p. ☐

We conclude this section with the application of Theorem 6.23.

Example 6.24. Consider the example AG01/#3.23 from the termination competition database, dubbed $\mathcal{R}_{\mathsf{f}}$:

$$\mathsf{f}(\mathsf{s}(x), y) \rightarrow \mathsf{f}(\mathsf{f}(x, y), y) \qquad\qquad \mathsf{f}(0, y) \rightarrow 0 \ .$$

It cannot be oriented with an instance $>_{\mathsf{mpo}}$ and thus also the orientation with a polynomial path orders fails. We choose the following model $\mathcal{B}$ for $\mathcal{R}_{\mathsf{f}}$: We define the carrier $B = \{0, 1\}$ and we interpret the function symbols by

$$\mathsf{f}_{\mathcal{B}}(x, y) = 0 \qquad\qquad 0_{\mathcal{B}} = 0 \qquad\qquad \mathsf{s}_{\mathcal{B}}(x) = 1 \ .$$

It is easy to verify that $\mathcal{B}$ is a model of $\mathcal{R}_{\mathsf{f}}$. We choose to label the symbol f via its first argument, that is we define $L_{\mathsf{f}} = \{0, 1\}$ and $\ell_{\mathsf{f}}(x, y) = x$. We choose not to label the remaining symbols, that is we define $L_{\mathsf{s}} = L_0 = \varnothing$. Then the labeled rewrite system $\mathcal{R}_{\mathsf{flab}}$ is given by the following three rewrite rules:

$$\begin{aligned} \mathsf{f}_1(\mathsf{s}(x), y) &\to \mathsf{f}_0(\mathsf{f}_0(x, y), y) & \qquad \mathsf{f}_0(0, y) &\to 0 \\ \mathsf{f}_1(\mathsf{s}(x), y) &\to \mathsf{f}_0(\mathsf{f}_1(x, y), y) \end{aligned}$$

We define some precedence $\gtrsim$ such that $\mathsf{f}_1 > \mathsf{f}_0$ and set the second argument of f_1 and all arguments of f_0 safe. That is, the safe mapping is given by

$$\mathsf{safe}(\mathsf{f}_0) = \{1, 2\} \qquad \mathsf{safe}(\mathsf{f}_1) = \{2\} \qquad \mathsf{safe}(\mathsf{s}) = \{2\} .$$

Then $\mathcal{R}_{\mathsf{flab}} \subseteq >_{\mathsf{pop*}}$ and as a consequence of Theorem 6.23 we conclude a polynomial innermost runtime-complexity of the unlabeled system $\mathcal{R}_{\mathsf{f}}$.

Again we stress that polynomial path orders together with semantic labeling gives rise to a completely automatic complexity analysis. We give an efficient implementation in the following Chapter.

7 Implementation Matters

In this chapter we investigate on the technical issues arising from the implementation of the polynomial path order. We provide an automation of the order in combination with finite semantic labeling. Moreover, we give an implementation of the polynomial path order in combination with argument filtering.

The orientation of a rewrite system $\mathcal{R}$ with a recursive path order amounts to finding a precedence $\gtrsim$ such that $\mathcal{R}$ is compatible with the induced order. The task becomes more involved if we want to show compatibility with a polynomial path order. Besides the precedence a suitable safe mapping needs to be found. Observe that a signature with m symbols gives rise to $\Omega(m!)$ total quasi precedences. Moreover, we can associate with each n-ary function symbol 2^n different safe positions. In other words, the search space is huge. In earlier work [6] we have shown that the precedence and safe mapping can be found via a constraint solving approach in reasonable time. However, this approach gets impractical if one wants to integrate semantic labeling or argument filtering. For the combination of the polynomial path order together with finite semantic labeling additionally a suitable model as well as a labeling needs to be found. Again this severely widens the search space, as a single function symbol of arity n gives rise to m^n different interpretation and labeling functions. Here m denotes the size of the carrier. Moreover, all those parameters are tightly inter-related: the rewrite system should be labeled suitable for the polynomial path order, and the precedence and safe mapping should be chosen such that compatibility with the labeled system holds. The situation is similar if we want to incorporate argument filterings. Each n-ary function symbol gives rise to $2^n + n$ different filterings. Again the choice of the filtering is tightly coupled to choice on the precedence and safe mapping.

Kurihara and Kondo [36] where the first to encode compatibility with a recursive path order as a satisfiability problem of propositional logic. Recently, this approach gained in popularity significantly [3, 24, 48, 57], in particular for the combination of basic termination methods with transformation techniques [19, 34, 35, 56]. We follow this approach, and describe both the polynomial path order and finite semantic labeling as well as the polynomial path order in combination with argument filterings as Boolean satisfiability problems (SAT). Via this reduction, all choices on the involved parameters can be left to a state-of-the-art SAT-solver.

A reduction to SAT of recursive path orders with semantic labeling over an infinite carrier is given in [35], and a reduction of recursive path orders together with *predictive labeling* is covered in [34]. As highlighted in Section 6.2, these approaches are inapplicable in our context since the employed carrier is usually infinite. Beside these, semantic labeling over a binary carrier is implemented in the termination prover TORPA [59]. Here (quasi-)model and labeling are found according to a heuristic, where the interpretation and labeling functions are chosen from a restricted set of constants, unary- and binary functions. For this it is a necessity to preprocess the input system such that all function symbols are of arity less or equal to two. After the labeling, compatibility with polynomial interpretations is checked. This approach admits several disadvantages: clearly the aforementioned transformation breaks the separation of safe and normal argument positions, it is thus not well-suited for our concerns; the overall procedure is based on trial and error, as labeling and compatibility with polynomial interpretations are independent. We consider the proposed translation to SAT favorable: the choice of labeling suitable for the base order is governed by a state-of-the-art SAT-solver, embracing the inter-relation ship between labeling and orientation; our reduction does not rely on heuristics nor on a restriction of the considered interpretation and labeling functions; experimental evidence confirm the feasibility of our approach.

We also want to mention [51] where *root-labeling* is introduced. Root-labeling is in essence an instance of finite semantic labeling. Here a function symbol is labeled with the root-symbols of direct subterms. In order to achieve the model condition, rules need to be closed under *flat contexts*, that is contexts of depth one. In particular this implies that root-labeling is inapplicable for our concerns: from a polynomial (innermost) runtime-complexity of the labeled rewrite system we cannot conclude a polynomial runtime-complexity of the original system in general.

The automation of the argument filtering transformation in combination with recursive path orders has also been extensively studied. Early approaches mainly concentrated on restricting the search space reasonably [27, 28] such that trial and error becomes feasible. Nowadays most implementations of argument filterings work via a transformation to SAT [19, 56]. In this work, we essentially follow [19, 56] but suitably adapt their encoding for the combination with the polynomial path order.

7.1 POP* Semantic Labeling

In this section we introduce our encoding of the polynomial path order together with finite semantic labeling. From an unlabeled rewrite system we construct a propositional formula ψ such that satisfiability of ψ certifies the existence of a labeled rewrite system $\mathcal{R}_{\mathsf{lab}}$ compatible with an instance $>_{\mathsf{pop}*}$. From a satisfying assignment, the labeled system $\mathcal{R}_{\mathsf{lab}}$ as well as the order $>_{\mathsf{pop}*}$ can be extracted. In our encoding, every function symbol is subject to labeling. This minor delineation from the standard definition of semantic labeling does not narrow the applicability: labeling every otherwise unlabeled symbol uniformly results in a renaming of the otherwise labeled system. For the ease of presentation, we assume a two-valued carrier $\mathbb{B} = \{\text{false}, \text{true}\}$ below. The encoding can be easily extended to arbitrary finite carriers.

In order to find suitable labeling- and interpretation functions, we first provide a way to propositional encode Boolean functions in a generic fashion. Via this encoding, it is possible to restrict the interpretations and labeling suitably. Let $b : \mathbb{B}^n \to \mathbb{B}$ denote an arbitrary Boolean function. To encode b, we make use of unique propositional atoms b_w for every sequence of arguments $w = w_1, \dots, w_n \in \mathbb{B}^n$. The atoms b_w denote the result of applying the arguments $w_1, \dots, w_n$ to b. Let $a_1, \dots, a_n$ be propositional formulas. To impose restrictions on the encoded function b, we introduce the formula $\ulcorner b \urcorner(a_1, \dots, a_n)$ such that for a satisfying assignment ν, the equality $\nu(\ulcorner b \urcorner(a_1, \dots, a_n)) = b(\nu(a_1), \dots, \nu(a_n))$ holds. Here ν is lifted to formulas in the obvious way. For this, when b denotes a constant, i.e. when $n = 0$, we define $\ulcorner b \urcorner = b_\varepsilon$. For $n \geqslant 1$, we let

$$\ulcorner b \urcorner(a_1, \dots, a_n) = \bigwedge_{w_1,\dots,w_n \in \mathbb{B}^n} \Big((\bigwedge_{i=1}^{n} w_i \leftrightarrow a_i) \to b_w \Big) .$$

For instance, the propositional formula $\ulcorner b \urcorner(a_1, a_2) \leftrightarrow r$ assesses that the encoded function b satisfies $b(\nu(a_1), \nu(a_2)) = \nu(r)$.

For the search of a suitable model $\mathcal{B}$ and labeling ℓ, for every assignment $\alpha : \mathcal{V} \to A$ and term t appearing in $\mathcal{R}$ we introduce fresh atoms $\mathrm{int}_{\alpha,t}$ and $\mathrm{lab}_{\alpha,t}$ for $t \notin \mathcal{V}$. The meaning of $\mathrm{int}_{\alpha,t}$ is the result of $[\alpha]_{\mathcal{B}}(t)$, whereas $\mathrm{lab}_{\alpha,t}$ denotes the label of the root symbol of t labeled under the assignment α. In order to ensure a correct valuation of $\mathrm{int}_{\alpha,t}$ and $\mathrm{lab}_{\alpha,t}$ for terms $t = f(t_1, \dots, t_n)$ and particular assignment α, we define

$$\begin{aligned} \mathrm{INT}_\alpha(t) &= \mathrm{int}_{\alpha,t} \leftrightarrow \ulcorner f_{\mathcal{B}} \urcorner(\mathrm{int}_{\alpha,t_1}, \dots, \mathrm{int}_{\alpha,t_n}), \text{ and} \\ \mathrm{LAB}_\alpha(t) &= \mathrm{lab}_{\alpha,t} \leftrightarrow \ulcorner \ell_f \urcorner(\mathrm{int}_{\alpha,t_1}, \dots, \mathrm{int}_{\alpha,t_n}). \end{aligned}$$

Here $\ulcorner f_{\mathcal{B}} \urcorner$ and $\ulcorner \ell_f \urcorner$ correspond to unique encodings for labeling and interpretation functions for the symbol f. Furthermore for $t \in \mathcal{V}$ we identify $\mathrm{int}_{\alpha,t}$ with $\alpha(t)$. Besides the model condition, the above constraints have to be enforced for every assignment α and every term appearing in $\mathcal{R}$. This is covered by

$$\mathrm{LAB}(\mathcal{R}) = \bigwedge_{l \to r \in \mathcal{R}} \bigwedge_{\alpha : \mathsf{Var}(l) \to \mathcal{B}} \Big(\mathrm{int}_{\alpha,l} \leftrightarrow \mathrm{int}_{\alpha,r} \wedge \bigwedge_{\substack{l \trianglerighteq t \text{ or} \\ r \trianglerighteq t}} (\mathrm{INT}_\alpha(t) \wedge \mathrm{LAB}_\alpha(t)) \Big) .$$

Assume ν is a satisfying assignment for $\mathrm{LAB}(\mathcal{R})$ and $\mathcal{R}_{\mathsf{lab}}$ denotes the system obtained by labeling $\mathcal{R}$ according to the encoded labeling and model. In order to prove compatibility of $\mathcal{R}_{\mathsf{lab}}$ with the polynomial path order, we need to find a precedence $\succsim$ and safe mapping safe such that $\mathcal{R}_{\mathsf{lab}} \subseteq >_{\mathsf{pop}*}$ holds for the induced order $>_{\mathsf{pop}*}$. Let f_a and g_b denote two labeled function symbols. For the propositional encoding of $f_a \succ g_b$, we introduce fresh atoms $\succ_{f_a,g_b}$, likewise for $f_a \sim g_b$ we introduce the atoms $\sim_{f_a,g_b}$. Let a and b be two propositional formulas that evaluate to the label of f and g respectively. Observe that besides the condition that the atoms $\succ_{f_a,g_b}$ and $\sim_{f_a,g_b}$ form a quasi-precedence, we also need to ensure the conditions of Definition 4.2: constructors need to be minimal in the precedence, and furthermore, the equivalence relation $\sim \subseteq \succsim$ needs to respects the separation of the signature $\mathcal{F}_{\mathsf{lab}}$ into defined and constructor symbols as given by $\mathcal{R}_{\mathsf{lab}}$. However, a labeled symbol f_a need not be defined with respect to $\mathcal{R}_{\mathsf{lab}}$, although the unlabeled symbol f is a defined symbol of

$\mathcal{R}$. To circumvent this problem, we add the rule $f_a(x_1, \ldots, x_n) \to \mathsf{c}$ with c a fresh constant to the labeled system and require $f_a \succ \mathsf{c}$ in the precedence. Alternatively one could propositionally encode whether f_a is a defined symbol with respect to $\mathcal{R}_{\mathsf{lab}}$ and adapt the formulas constructed below accordingly. However, experimental findings indicate that the former and here described approach is favorable. We define

$$\begin{aligned}\ulcorner f_a > g_b \urcorner = {} & (\ \ulcorner g \in \mathcal{D} \urcorner \to \ulcorner f \in \mathcal{D} \urcorner\ ) \\ & \wedge (\ \ a \wedge b \ \ \to \succ_{f_{\text{true}}, g_{\text{true}}}) \\ & \wedge (\ \neg a \wedge b \ \ \to \succ_{f_{\text{false}}, g_{\text{true}}}) \\ & \wedge (\ \ a \wedge \neg b \ \ \to \succ_{f_{\text{true}}, g_{\text{false}}}) \\ & \wedge (\ \neg a \wedge \neg b \ \ \to \succ_{f_{\text{false}}, g_{\text{false}}}) \ .\end{aligned}$$

Here $\ulcorner f \in \mathcal{D} \urcorner = \top$ when $f \in \mathcal{D}$ and $\bot$ else.

Remark. Observe that when $f \in \mathcal{C}$ and $g \in \mathcal{D}$, then the above formula simplifies to $\bot$. In our implementation, we exploit this fact via lazy construction and intermediate simplification. For instance, for a formula $A \wedge B$, one can first construct the formula A, simplify A by applying the laws of propositional logic and check whether the result equals $\bot$. In this case, it does not make sense to even construct B. Similar observations apply for the Boolean connectives $\vee, \to, \leftrightarrow, \ldots$. For the encoding $\ulcorner f_a > g_b \urcorner$ from above, such an evaluation strategy might save us from constructing the case analysis on the labels and allocating four fresh atoms. At the same time, the approach gives a clean and readable presentation as well as implementation.

Correspondingly, we define the formula $\ulcorner f_a \sim g_b \urcorner$ asserting $f_a \sim g_a$. It is given by

$$\begin{aligned}\ulcorner f_a \sim g_b \urcorner = {} & (\ \ulcorner f \in \mathcal{D} \urcorner \leftrightarrow \ulcorner g \in \mathcal{D} \urcorner\ ) \\ & \wedge (\ \ a \wedge b \ \ \to \sim_{f_{\text{true}}, g_{\text{true}}}) \\ & \wedge (\ \neg a \wedge b \ \ \to \sim_{f_{\text{false}}, g_{\text{true}}}) \\ & \wedge (\ \ a \wedge \neg b \ \ \to \sim_{f_{\text{true}}, g_{\text{false}}}) \\ & \wedge (\ \neg a \wedge \neg b \ \ \to \sim_{f_{\text{false}}, g_{\text{false}}}) \ .\end{aligned}$$

Beside the precedence, for each labeled function symbol the safe mapping needs to be encoded. For that, we employ for each defined symbol $f \in \mathcal{F}$, label $a \in \mathbb{B}$ and argument positions i of f fresh atoms $\mathsf{safe}_{f_a,i}$. For the unlabeled symbol $f \in \mathcal{D}$ and formula a representing the label, the formula

$$\mathrm{SF}(f_a, i) = (a \to \mathsf{safe}_{f_{\text{true}},i}) \wedge (\neg a \to \mathsf{safe}_{f_{\text{false}},i})$$

assesses depending on the valuation of a that either the i-th position of f_{true} or of f_{false} is safe. In a similar spirit we encode that the i-th position of f_a needs to be normal with

$$\mathrm{NRM}(f_a, i) = (a \to \mathsf{nrm}_{f_{\text{true}},i}) \wedge (\neg a \to \mathsf{nrm}_{f_{\text{false}},i}) \ .$$

For the case $f \in \mathcal{C}$ we define $\mathrm{SF}(f_a, i) = \top$ and $\mathrm{NRM}(g_a, i) = \bot$ (remember that all argument positions of constructor symbols need to be safe).

In the following, let s and t be two concrete (unlabeled) terms, and let $\alpha : \mathcal{V} \to A$ denote an assignment. In order to compare the labeled versions of s and t under the equivalence relation $\overset{\mathsf{s}}{\sim}$ we introduce the formula $\ulcorner s \overset{\mathsf{s}}{\sim} t \urcorner_\alpha$. Below we assume that all cases not mentioned are defined as $\bot$. Suppose $s = f(s_1, \ldots, s_n)$ and $t = g(t_1, \ldots, t_n)$. For the comparison $s \overset{\mathsf{s}}{\sim} t$, we need to assert that $f \sim g$ and the existence of a permutation π such that $s_i \overset{\mathsf{s}}{\sim} t_{\pi(t)}$ holds for all argument positions i. Furthermore, we have to demand $i \in \mathsf{safe}(f)$ exactly if $\pi(i) \in \mathsf{safe}(g)$. In order to describe the permutation, we essentially follow the idea of *multiset covers* (cf. [48]), also employed later on. We introduce fresh atoms $\beta_{i,j}$ for $i, j \in \{1, \ldots, n\}$. These represent π in such a way that $\beta_{i,j}$ is true if and only if $\pi(i) = j$. We define

$$\ulcorner \beta \urcorner = \bigwedge_{i=1}^{n} \mathrm{one}(\beta_{i,1}, \ldots, \beta_{i,n}) \wedge \mathrm{one}(\beta_{1,i}, \ldots, \beta_{n,i}) \ .$$

Here the formula $\mathrm{one}(a_1, \ldots, a_n)$ ensures that exactly one of the formulas a_i for $i \in \{1, \ldots, n\}$ holds. It is given recursively by

$$\begin{aligned} \mathrm{one}() &= \bot \\ \mathrm{one}(a_1, \ldots, a_n) &= (a_1 \to \mathrm{zero}(a_2, \ldots, a_n)) \wedge (\neg a_1 \to \mathrm{one}(a_2, \ldots, a_n)) \\ \mathrm{zero}() &= \top \\ \mathrm{zero}(a_1, \ldots, a_n) &= \neg a_1 \wedge \mathrm{zero}(a_2, \ldots, a_n) \ . \end{aligned}$$

Hence when $\ulcorner \beta \urcorner$ holds, the atoms $\beta_{i,j}$ resemble a one-to-one mapping between the argument positions of f and g. Summing up, the equivalence $s \overset{\mathsf{s}}{\sim} t$ becomes expressible by

$$\begin{aligned} \ulcorner f(s_1, \ldots, s_n) \overset{\mathsf{s}}{\sim} g(t_1, \ldots, t_n) \urcorner_\alpha = {} & \ulcorner s = t \urcorner_\alpha \\ & \vee \Big(\ulcorner f_{\mathrm{lab}_{\alpha,s}} \sim g_{\mathrm{lab}_{\alpha,t}} \urcorner \wedge \ulcorner \beta \urcorner \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} (\beta_{i,j} \to \ulcorner s_i \overset{\mathsf{s}}{\sim} t_j \urcorner_\alpha \\ & \qquad \wedge \big(\mathrm{SF}(f_{\mathrm{lab}_{\alpha,s}}, i) \leftrightarrow \mathrm{SF}(g_{\mathrm{lab}_{\alpha,t}}, j)))\Big) \end{aligned}$$

where $\ulcorner s = t \urcorner_\alpha$ evaluates to $\top$ or $\bot$ accordingly. It is straight forward to verify that the subformula $\mathrm{SF}(f_{\mathrm{lab}_{\alpha,s}}, i) \leftrightarrow \mathrm{SF}(g_{\mathrm{lab}_{\alpha,t}}, j)$ specifies $i \in \mathsf{safe}(f_a) \iff j \in \mathsf{safe}(g_b)$ where a and b denote the labels of root symbols of $\mathsf{lab}_s(\alpha)$ and $\mathsf{lab}_t(\alpha)$ respectively.

Based on the encoding for $\overset{\mathsf{s}}{\sim}$, we give a propositional encoding of $\mathsf{lab}_\alpha(s) >_{\mathsf{pop}*} \mathsf{lab}_\alpha(t)$ as well as $\mathsf{lab}_\alpha(s) \gtrsim_{\mathsf{pop}*} \mathsf{lab}_\alpha(t)$. In the following, we suppose that $s = f(s_1, \ldots, s_n)$. We introduce formulas

$$\ulcorner s >_{\mathsf{pop}*} t \urcorner_\alpha = \ulcorner s >_{\mathsf{pop}} t \urcorner_\alpha \vee \ulcorner s >^{(1)}_{\mathsf{pop}*} t \urcorner_\alpha \vee \ulcorner s >^{(2)}_{\mathsf{pop}*} t \urcorner_\alpha \vee \ulcorner s >^{(3)}_{\mathsf{pop}*} t \urcorner_\alpha, \text{ and}$$

$$\ulcorner s \gtrsim_{\mathsf{pop}*} t \urcorner_\alpha = \ulcorner s_i >_{\mathsf{pop}*} t \urcorner_\alpha \vee \ulcorner s_i \overset{\mathsf{s}}{\sim} t \urcorner_\alpha \ .$$

Here $\ulcorner s >_{\mathsf{pop}} t \urcorner_\alpha$ refers to the encoding of $\mathsf{lab}_\alpha(s) >_{\mathsf{pop}} \mathsf{lab}_\alpha(t)$, and $\ulcorner s >^{(i)}_{\mathsf{pop}*} t \urcorner$ refer to the encodings for orienting $\mathsf{lab}_\alpha(s)$ and $\mathsf{lab}_\alpha(t)$ via the clause (i) from

Definition 4.5. In the following we discuss the clauses (1) – (3) from the definition of $>_{\mathsf{pop*}}$, the comparison using the auxiliary order $>_{\mathsf{pop}}$ essentially amounts to the disjunction of the encodings for clauses (1) and (2).

For the encoding of clause (1), notice that $s_i = t$ implies $\mathsf{lab}_\alpha(s_i) = \mathsf{lab}_\alpha(t)$. Thus it is legal to define $\ulcorner f(s_1,\dots,s_n) >^{(1)}_{\mathsf{pop*}} t\urcorner_\alpha = \top$ if $s_i = t$ holds for some s_i. Otherwise, we define

$$\ulcorner f(s_1,\dots,s_n) >^{(1)}_{\mathsf{pop*}} t\urcorner_\alpha = \bigvee_{i=1}^{n} \ulcorner s \gtrsim_{\mathsf{pop*}} t\urcorner_\alpha \ .$$

For the encoding of $\mathsf{lab}_\alpha(f(s_1,\dots,s_n)) >_{\mathsf{pop*}} \mathsf{lab}_\alpha(g(t_1,\dots,t_m))$ via the second clause of $>_{\mathsf{pop*}}$, we introduce fresh atoms δ_j for each argument position j of g. The formula $\ulcorner\delta\urcorner = \mathrm{one}(\delta_1,\dots,\delta_m)$ assures that exactly one atom δ_j is true. This particular atom marks the unique safe argument position j_0 of t where $\mathsf{lab}_\alpha(s) >_{\mathsf{pop*}} \mathsf{lab}_\alpha(t_j)$ holds. Assume the unlabeled symbol f is a defined symbol of $\mathcal{R}$. By the above introduced dummy rules $f_a \in \mathcal{F}_{\mathsf{lab}}$ is a defined symbol of $\mathcal{R}_{\mathsf{lab}}$, independent on the label a. Clause (2) is expressed by

$$\begin{aligned}\ulcorner f(s_1,\dots,s_n) >^{(2)}_{\mathsf{pop*}} g(t_1,\dots,t_m)\urcorner_\alpha &= \ulcorner f \in \mathcal{D}\urcorner \wedge \ulcorner f_{\mathrm{lab}_{\alpha,s}} > g_{\mathrm{lab}_{\alpha,t}}\urcorner \wedge \ulcorner\delta\urcorner \\ &\wedge \bigwedge_{j=1}^{m} \Big(\big(\delta_j \to \ulcorner s >_{\mathsf{pop*}} t_j\urcorner_\alpha \wedge \mathrm{SF}(g_{\mathrm{lab}_{\alpha,t}}, j)\big) \\ &\qquad \wedge \big(\neg\delta_j \to \ulcorner s >^{\pi}_{\mathsf{pop}} t_j\urcorner_\alpha \vee \ulcorner s \rhd t_j\urcorner \wedge \mathrm{SF}(g_{\mathrm{lab}_{\alpha,t}}, j)\big)\Big) \ ,\end{aligned}$$

where $\ulcorner s \rhd t_i\urcorner = \top$ when $s \rhd t_i$ holds, and $\bot$ otherwise. This is legal, as the subterm property is closed under labeling.

To encode multiset comparisons, we make use of *multiset covers* as employed in [48]. A multiset cover is a pair of total mappings $\gamma\colon \{1,\dots,n\} \to \{1,\dots,m\}$ and $\varepsilon\colon \{1,\dots,n\} \to \mathbb{B}$, encoded using fresh atoms $\gamma_{i,j}$ and ε_i. The underlying idea is as follows: assume $[s_1,\dots,s_n] \succ_{\mathsf{mul}} [t_1,\dots,t_m]$ holds for the strict multiset extension $\succ_{\mathsf{mul}}$ of some preorder $\succsim$, and let $\succ$ and $\sim$ correspond to the strict and equivalent part of $\succsim$ respectively. Equivalently, every term t_j is *covered* by some term s_i, either by $s_i \sim t_j$ or $s_i \succ t_j$. For the case when $s_i \sim t_j$, the element s_i covers no element besides t_j and at least one cover is due to a strict comparison. Again equivalently, there exists a multiset cover (γ,ε) such that for $\gamma(i) = j$, $s_i \succsim t_j$ holds, $\varepsilon(i)$ implies $s_i \sim t_j$ and $\varepsilon(i)$ does not hold for at least one $i \in \{1,\dots,n\}$. We encode $\gamma(i) = j$ by $\gamma_{i,j}$ and indicate $\varepsilon(i)$ by ε_i. To assess the above mentioned constraints for a multiset cover (γ,ε), we introduce the formula

$$\ulcorner(\gamma,\varepsilon)\urcorner = \bigwedge_{j=1}^{m} \mathrm{one}(\gamma_{1,j},\dots,\gamma_{n,j}) \wedge \bigwedge_{i=1}^{n} (\varepsilon_i \to \mathrm{one}(\gamma_{i,1},\dots,\gamma_{i,m})) \ .$$

Clause (3) from the definition of $>_{\mathsf{pop*}}$ requires both $\mathsf{nrm}(s) >^{\mathsf{mul}}_{\mathsf{pop*}} \mathsf{nrm}(t)$ and $\mathsf{safe}(s) \gtrsim^{\mathsf{mul}}_{\mathsf{pop*}} \mathsf{safe}(t)$. It can be verified that the above corresponds to $\mathsf{nrm}(s) \uplus \mathsf{safe}(s) \gtrsim^{\mathsf{mul}}_{\mathsf{pop*}} \mathsf{nrm}(t) \uplus \mathsf{safe}(t)$ provided the corresponding multiset cover (γ,ε) satisfies that (i) the mapping γ respects the separation of safe and normal argument positions and furthermore (ii) $\varepsilon(i)$ is false for some normal argument position i of $\mathrm{root}(s)$. The latter observation gives rise to a concise encoding of clause (3). We define

$$\begin{aligned}
&\ulcorner f(s_1,\dots,s_n) >^{(3)}_{\mathsf{pop*}} g(t_1,\dots,t_m)\urcorner_\alpha = \\
&\qquad \ulcorner f \in \mathcal{D}\urcorner \wedge \ulcorner f \sim g\urcorner \wedge \ulcorner(\gamma,\varepsilon)\urcorner \wedge \bigvee_{i=1}^{n}\big(\mathrm{NRM}(f_{\mathrm{lab}_{\alpha,s}}, i) \wedge \neg\varepsilon_i\big) \\
&\qquad \wedge \bigwedge_{i=1}^{m}\bigwedge_{j=1}^{n}\Big(\gamma_{i,j} \to \big((\varepsilon_i \to \ulcorner s_i \overset{\mathsf{s}}{\sim} t_j\urcorner_\alpha) \wedge (\neg\varepsilon_i \to \ulcorner s_i >_{\mathsf{pop*}} t_j\urcorner_\alpha) \\
&\qquad\qquad \wedge (\mathrm{SF}(f_{\mathrm{lab}_{\alpha,s}}, i) \leftrightarrow \mathrm{SF}(g_{\mathrm{lab}_{\alpha,t}}, j))\big)\Big) \ .
\end{aligned}$$

Above we enforce that for a valid multiset cover, depending on ε, either $s_i \overset{\mathsf{s}}{\sim} t_j$ or $s_i >^{\pi}_{\mathsf{pop*}} t_j$ holds for all i, j with $\gamma(i) = j$. We assert the separation of normal and safe argument positions by the subformula $\mathrm{SF}(f_{\mathrm{lab}_{\alpha,s}}, i) \leftrightarrow \mathrm{SF}(g_{\mathrm{lab}_{\alpha,t}}, j)$. Moreover, the subformula $\bigvee_{i=1}^{n}\big(\mathrm{NRM}(f_{\mathrm{lab}_{\alpha,s}}, i) \wedge \neg\varepsilon_i\big)$ demands at least one strict comparison of normal arguments. This completes the definition of $\ulcorner f(s_1,\dots,s_n) >_{\mathsf{pop*}} g(t_1,\dots,t_n)\urcorner_\alpha$.

To verify that the labeled system $\mathcal{R}_{\mathsf{lab}}$ is compatible with an instance of $>_{\mathsf{pop*}}$, we enforce the constraint $\ulcorner l >_{\mathsf{pop*}} r\urcorner_\alpha$ for every rule $l \to r \in \mathcal{R}$ and assignment $\alpha \colon \mathsf{Var}(l) \to \mathcal{B}$. Moreover, we need to assert a valid encoding for the quasi precedence, that is the employed atoms $\ulcorner f_a > g_b\urcorner$ form a proper order, and the atoms $\ulcorner f_a \sim g_b\urcorner$ define an equivalence relation on $\mathcal{F}_{\mathsf{lab}}$. One way to do so is to encode transitivity and so forth directly. However, a straight forward encoding is of size $\Omega(n^2)$ where n is the cardinality of $\mathcal{F}_{\mathsf{lab}}$. For a more concise encoding we follow [19]. We embed the precedence $\succsim$ into the usual preorder $\geqslant$ on the segment $\{0,\dots,2^{\lceil\log(n)\rceil} - 1\}$ of natural numbers. Then $\succsim$ is a quasi-precedence if there exists a order preserving map ϕ such that when $f \succ g$ then $\phi(f) > \phi(g)$ and $f \sim g$ implies $\phi(f) = \phi(g)$. For each function symbol $f \in \mathcal{F}_{\mathsf{lab}}$ we introduce a list of $\lceil\log(n)\rceil$ fresh atoms encoding $\phi(f)$ and denoted by $\vec{\phi_f}$. We define[1]

$$\mathrm{PREC}(\mathcal{F}) = \bigwedge_{f\in\mathcal{F}}\bigwedge_{g\in\mathcal{F}} (\ulcorner f_a > g_b\urcorner \to \ulcorner\vec{\phi_f} > \vec{\phi_g}\urcorner) \wedge (\ulcorner f_a \sim g_b\urcorner \to \ulcorner\vec{\phi_f} = \vec{\phi_g}\urcorner) \ .$$

Here $\ulcorner\vec{\phi_f} > \vec{\phi_g}\urcorner$ and $\ulcorner\vec{\phi_f} = \vec{\phi_g}\urcorner$ denote the straight forward encoding of $>$ and $=$ over k-bit naturals into propositional formulas. The resulting constraint is of size $\mathcal{O}(n \cdot \log(n))$. Summing up, satisfiability of the formula

$$\mathrm{POP}^*_{\mathrm{SL}}(\mathcal{R}) = \mathrm{PREC}(\mathcal{F}_{\mathsf{lab}}) \wedge \mathrm{LAB}(\mathcal{R}) \wedge \bigwedge_{\alpha:\mathsf{Var}(l)\to\mathcal{B}}\ \bigwedge_{l\to r\in\mathcal{R}} \ulcorner l >_{\mathsf{pop*}} r\urcorner_\alpha$$

certifies the existence of a model $\mathcal{B}$ and labeling ℓ such that the rewrite system

$$\mathcal{R}_{\mathsf{lab}} \cup \{f_a(x_1,\dots,x_n) \to \mathsf{c} \mid f \in \mathcal{D}\}$$

is compatible with an instance $>_{\mathsf{pop*}}$. Thus from Theorem 6.23, satisfiability of $\mathrm{POP}^*_{\mathrm{SL}}(\mathcal{R})$ asserts a polynomial bound on the innermost runtime-complexity of $\mathcal{R}$.

[1] In our implementation, we restrict the constraint to atoms $\ulcorner f_a > g_b\urcorner$ referenced by $\bigwedge_{l\to r\in\mathcal{R}}\bigwedge_{\alpha}\ulcorner l >_{\mathsf{pop*}} r\urcorner_\alpha$.

7.2 POP* and Dependency Pairs

In order to encode $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop}*}$ for some instance $>^{\pi}_{\mathsf{pop}*}$ and rewrite system $\mathcal{R}$, we essentially follow the approach described in Section 7.1. For each pair of function symbols we again introduce fresh atoms $\succ_{f,g}$ and $\sim_{f,g}$. To assess that the encoded precedence $\succsim$ is admissible for $>^{\pi}_{\mathsf{pop}*}$, and for a concise encoding, we identify $\succ_{f,g}$ with $\bot$ for $f \in \mathcal{C}$ and $g \in \mathcal{D}$. Moreover, we identify $\sim_{f,g}$ with $\bot$ when $f \in \mathcal{D}$ and $g \in \mathcal{C}$ and vice versa. Again we use fresh atoms $\mathsf{safe}_{f,i}$ with the meaning that the i-th position of the function symbol f is safe and identify $\mathsf{safe}_{f,i}$ with $\top$ for $f \in \mathcal{C}$ and argument position i of f. Beside the precedence and safe mapping, an instance $>^{\pi}_{\mathsf{pop}*}$ is induced by an argument filtering π. In order to encode π, we follow the standard approach, cf. [56] for instance. For each function symbol f of arity n, we introduce an atom π_f^{list} with the meaning that π is non-collapsing for f, i.e. $\pi(f) = [i_1, \dots, i_j]$ for some $1 \leqslant i_1 \leqslant \cdots \leqslant i_j \leqslant n$. Beside these, we introduce for each argument position i of f the atom π_f^i, indicating that depending on π_f^{list}, either $i \in \pi(f)$ or $\pi(f) = i$. In the following, we exploit that when π collapses f, then exactly one of the atoms π_f^i holds. We enforce this via the formula

$$\ulcorner \pi \urcorner = \bigwedge_{f \in \mathcal{F}} \neg \pi_f^{\mathsf{list}} \to \mathrm{one}(\pi_f^1, \dots, \pi_f^n) \ .$$

In a similar spirit to Section 7.1, we first give constraints for $s \overset{\mathsf{s}}{\sim}_\pi t$, $s >^{\pi}_{\mathsf{pop}} t$ and $s >^{\pi}_{\mathsf{pop}*} t$ with concrete terms s and t. We continue with the encoding of $s \overset{\mathsf{s}}{\sim}_\pi t$. Again, all cases not defined below are defined as $\bot$. Let s and t be two terms. We define

$$\ulcorner s \overset{\mathsf{s}}{\sim}_\pi t \urcorner = \ulcorner s = t \urcorner \vee \ulcorner s \overset{\mathsf{s}}{\sim}{}^{(2)}_\pi t \urcorner \vee \ulcorner s \overset{\mathsf{s}}{\sim}{}^{(3)}_\pi t \urcorner$$

where the formulas $\ulcorner s \overset{\mathsf{s}}{\sim}{}^{(2)}_\pi t \urcorner$ and $\ulcorner s \overset{\mathsf{s}}{\sim}{}^{(3)}_\pi t \urcorner$ correspond to the encoding of clause (2) and (3) of Definition 5.7 respectively. For the case when $s = f(u_1, \dots, u_n)$ and $t = x \in \mathcal{V}$, or $s = x \in \mathcal{V}$ and $t = f(u_1, \dots, u_n)$ we define

$$\ulcorner s \overset{\mathsf{s}}{\sim}{}^{(2)}_\pi t \urcorner = \neg \pi_f^{\mathsf{list}} \wedge \bigwedge_{i=1}^{n} (\pi_f^i \to \ulcorner u_i \overset{\mathsf{s}}{\sim}_\pi x \urcorner)$$

In the above constraint, $\neg \pi_f^{\mathsf{list}}$ enforces that the argument filtering π collapses f. Observe that by the additional constraints on the encoding of π there exists exactly one argument position such that π_f^i evaluates to a true value. For this particular argument position i, the above formula requires $u_i \overset{\mathsf{s}}{\sim}_\pi x$.

When $s = f(s_1, \dots, s_n)$ and $t = g(t_1, \dots, t_m)$ we define

$$\begin{aligned}
\ulcorner s \overset{\mathsf{s}}{\sim}{}^{(3)}_\pi t \urcorner = \ & \big(\neg \pi_f^{\mathsf{list}} \to \bigwedge_{i=1}^{n} (\pi_f^i \to \ulcorner s_i \overset{\mathsf{s}}{\sim}_\pi t \urcorner)\big) \\
& \wedge \ \big(\pi_f^{\mathsf{list}} \to \big((\neg \pi_g^{\mathsf{list}} \to \bigwedge_{j=1}^{m} (\pi_g^j \to \ulcorner s \overset{\mathsf{s}}{\sim}_\pi t_j \urcorner)) \\
& \qquad \wedge (\pi_g^{\mathsf{list}} \to \ulcorner f \sim g \urcorner \wedge \ulcorner [s_1, \dots, s_n] \overset{\mathsf{s}}{\sim}_\pi [t_1, \dots, t_m] \urcorner)\big) \ ,
\end{aligned}$$

where we perform case distinction on π_f^{list} and π_g^{list}: when π collapses f or g, then $\ulcorner s_i \overset{\mathsf{s}}{\sim}_\pi t \urcorner$ or $\ulcorner s \overset{\mathsf{s}}{\sim}_\pi t_j \urcorner$ is enforced respectively. For the remaining case, by the formula $\ulcorner [s_1, \dots, s_n] \overset{\mathsf{s}}{\sim}_\pi [t_1, \dots, t_m] \urcorner$ we encode that the multisets $\mathsf{nrm}_\pi(s)$ and $\mathsf{nrm}_\pi(t)$ as well as $\mathsf{safe}_\pi(s)$ and $\mathsf{safe}_\pi(t)$ are equivalent under $\overset{\mathsf{s}}{\sim}_\pi$. For that, similar to the corresponding encoding from Section 7.1, we introduce the atoms $\beta_{i,j}$ for argument positions i of f and j of g. Again, the meaning of $\beta_{i,j}$ is that the i-th (not erased) argument of s is compared with the j-th (not erased) argument of t. We define

$$\ulcorner [s_1, \dots, s_n] \overset{\mathsf{s}}{\sim}_\pi [t_1, \dots, t_m] \urcorner = \\ \ulcorner \beta \urcorner \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} (\beta_{i,j} \to \pi_f^i \wedge \pi_g^j \wedge (\mathsf{safe}_{f,i} \leftrightarrow \mathsf{safe}_{g,j}) \wedge \ulcorner s_i \overset{\mathsf{s}}{\sim}_\pi t_j \urcorner) \ ,$$

where the formula

$$\ulcorner \beta \urcorner = \bigwedge_{i=1}^{n} (\pi_f^i \to \mathrm{one}(\beta_{i,1}, \dots, \beta_{i,m})) \wedge \bigwedge_{i=1}^{n} (\pi_g^j \to \mathrm{one}(\beta_{1,j}, \dots, \beta_{n,j}))$$

enforces that every argument of s and t not erased by π is covered respectively.

Next, we develop the encoding for $>_{\mathsf{pop}*}^{\pi}$ and $>_{\mathsf{pop}}^{\pi}$ in two steps: we provide an encoding of the individual clauses of Definition 5.8 and 5.9. For each order the disjunction of those formulas provide a sound but quite inefficient encoding for the comparison of two terms. Unlike the polynomial path order together with semantic labeling, the encodings of the individual clauses from $>_{\mathsf{pop}*}^{\pi}$ and $>_{\mathsf{pop}}^{\pi}$ share recursively computed subformulas. For instance, both clauses (1) and (2) from the definition of $>_{\mathsf{pop}}^{\pi}$ reference $s_i >_{\mathsf{pop}}^{\pi} t$. It is thus natural to combine the encoding of both clauses, by applying laws of propositional logic. This severely decreases the size of the encoding.

Let $s = f(s_1, \dots, s_n)$ and t be two terms. We start with $\ulcorner s >_{\mathsf{pop}}^{\pi} t \urcorner$, the encoding for auxiliary order $>_{\mathsf{pop}}^{\pi}$. Below, we assume

$$\ulcorner s \gtrsim_{\mathsf{pop}}^{\pi} t \urcorner = \ulcorner s >_{\mathsf{pop}}^{\pi} t \urcorner \vee \ulcorner s \overset{\mathsf{s}}{\sim}_\pi t \urcorner \ .$$

The encoding of clause (1) in Definition 5.8 is expressible by

$$\neg \pi_f^{\mathsf{list}} \wedge \bigvee_{i=1}^{n} \pi_f^i \wedge \ulcorner s_i >_{\mathsf{pop}}^{\pi} t \urcorner \ . \tag{i}$$

In a similar spirit, we encode the second clause by

$$\pi_f^{\mathsf{list}} \wedge \bigvee_{i=1}^{n} \pi_f^i \wedge \ulcorner s_i \gtrsim_{\mathsf{pop}}^{\pi} t \urcorner \wedge (\ulcorner f \not\in \mathcal{D} \urcorner \vee \neg\, \mathsf{safe}_{f,i}) \ . \tag{ii}$$

As $\ulcorner s_i \gtrsim_{\mathsf{pop}}^{\pi} t \urcorner = \ulcorner s_i >_{\mathsf{pop}}^{\pi} t \urcorner \vee \ulcorner s_i \overset{\mathsf{s}}{\sim}_\pi t \urcorner$, it is not difficult to argue that the formula

$$\ulcorner s >_{\mathsf{pop}}^{\pi\ (1,2)} t \urcorner = \\ \bigvee_{i=1}^{n} \pi_f^i \wedge \big(\ulcorner s_i >_{\mathsf{pop}}^{\pi} t \urcorner \vee (\ulcorner s_i \sim t \urcorner \wedge \pi_f^{\mathsf{list}})\big) \wedge (\neg \pi_f^{\mathsf{list}} \vee \ulcorner f \not\in \mathcal{D} \urcorner \vee \neg\, \mathsf{safe}_{f,i})$$

exactly expresses the disjunction of (i) and (ii), yielding a concise encoding of clauses (1) and (2). Next, assume $t = g(t_1, \dots, t_m)$. We initially formulate clause (3) by the propositional formula

$$\pi_f^{\mathsf{list}} \land \Big(\neg\pi_g^{\mathsf{list}} \to \bigwedge_{j=1}^{m} (\pi_g^j \to \ulcorner s >_{\mathsf{pop}}^{\pi} t_j \urcorner)\Big)$$
$$\Big(\pi_g^{\mathsf{list}} \to \bigwedge_{j=1}^{m} \big(\pi_g^j \to \ulcorner s >_{\mathsf{pop}}^{\pi} t_j \urcorner\big) \land \ulcorner f \in \mathcal{D} \urcorner \land \succ_{f,g}\Big) \; .$$

where we perform case distinction on whether π collapses g or not (again we exploit that whenever π_g^{list} does not hold, then exactly one of π_g^i for $1 \leqslant i \leqslant m$ evaluates to $\top$). From the above, it is easy to see that

$$\ulcorner s >_{\mathsf{pop}}^{\pi\ (3)} t \urcorner = \pi_f^{\mathsf{list}} \land \bigwedge_{j=1}^{m} (\pi_g^j \to \ulcorner s >_{\mathsf{pop}}^{\pi} t_j \urcorner) \land (\pi_g^{\mathsf{list}} \to \ulcorner f \in \mathcal{D} \urcorner \land \ulcorner f \succ g \urcorner)$$

expresses clause (3) from Definition 5.8 equivalently. We define

$$\ulcorner s >_{\mathsf{pop}}^{\pi} t \urcorner = \ulcorner s >_{\mathsf{pop}}^{\pi\ (1,2)} t_j \urcorner \lor \ulcorner s >_{\mathsf{pop}}^{\pi\ (3)} t_j \urcorner$$

which gives the final encoding of the auxiliary order $>_{\mathsf{pop}}^{\pi}$.

In order to give the propositional formula $\ulcorner s >_{\mathsf{pop*}}^{\pi} t \urcorner$, that is an encoding of $s >_{\mathsf{pop*}}^{\pi} t$, we essentially proceed as above. First, we provide an encoding for the disjunction of clause (1) and (2):

$$\ulcorner s >_{\mathsf{pop}}^{\pi\ (1,2)} t \urcorner = \bigvee_{i=1}^{n} \pi_f^i \land (\ulcorner s_i >_{\mathsf{pop*}}^{\pi} t \urcorner \lor \ulcorner s_i \overset{\mathsf{s}}{\sim}_{\pi} t \urcorner \land \pi_f^{\mathsf{list}}) \; .$$

For the encoding of clause (3a) and (3b) from the Definition of $>_{\mathsf{pop*}}^{\pi}$, observe that we need to demand in both cases $s >_{\mathsf{pop*}}^{\pi} t_j$ for some argument position j. First, we start with the naive encoding

$$\pi_f^{\mathsf{list}} \land (\neg\pi_g^{\mathsf{list}} \to \ulcorner s >_{\mathsf{pop*}}^{\pi\ (3a)} t \urcorner) \land (\pi_g^{\mathsf{list}} \to \ulcorner s >_{\mathsf{pop*}}^{\pi\ (3b)} t \urcorner) \qquad \text{(iii)}$$

where the formulas

$$\ulcorner s >_{\mathsf{pop*}}^{\pi\ (3a)} t \urcorner = \bigwedge_{j=1}^{m} (\neg\pi_g^j \lor \ulcorner s >_{\mathsf{pop*}}^{\pi} t_j \urcorner) \text{ and}$$
$$\ulcorner s >_{\mathsf{pop*}}^{\pi\ (3b)} t \urcorner = \ulcorner f \in \mathcal{D} \urcorner \land \ulcorner f > g \urcorner \land \mathrm{one}(\delta_1, \dots, \delta_n)$$
$$\land \bigwedge_{j=1}^{m} \Big(\big(\delta_j \to \ulcorner s >_{\mathsf{pop*}}^{\pi} t_j \urcorner \land \pi_g^j \land \mathsf{safe}_{g,j}\big)$$
$$\land \big(\neg\delta_j \to \neg\pi_g^j \lor \ulcorner s >_{\mathsf{pop}}^{\pi} t_j \urcorner \lor \mathsf{safe}_{g,j} \land \ulcorner s \rhd t_j \urcorner\big)\Big)$$

give a direct encoding of clauses (3a) and (3b) respectively. For clause (3b) we again employ m fresh atoms δ_j, and by the formula $\mathrm{one}(\delta_1, \dots, \delta_m)$ we enforce that exactly one of them evaluates to $\top$. This particular positive atom δ_{j_0} marks the safe argument position of g where the comparison using $>^{\pi}_{\mathsf{pop}*}$ is allowed. With all other atoms δ_j with $j \neq j_0$, we enforce that the argument position j is either erased by the argument filtering, or there is a strict decrease with respect to the auxiliary order or subterm relation. For the last case, we additionally demand $j \in \mathsf{safe}(g)$ as required in clause $(3b)$ from Definition 5.9. Consider the abbreviations

$$\begin{aligned}
A &= \ulcorner f \in \mathcal{D} \urcorner \wedge \ulcorner f > g \urcorner \wedge \text{one}(\delta_1, \dots, \delta_n) \text{ and} \\
B &= (\delta_j \rightarrow \pi^j_g \wedge \mathsf{safe}_{g,j}) \\
&\quad \wedge (\neg\delta_j \rightarrow \neg\pi^j_g \vee \ulcorner s >^{\pi}_{\mathsf{pop}} t_j \urcorner \vee \mathsf{safe}_{g,j} \wedge \ulcorner s \rhd t_j \urcorner) \ .
\end{aligned}$$

By exploiting the equivalence

$$\delta_j \rightarrow \ulcorner s >^{\pi}_{\mathsf{pop}*} t_j \urcorner \wedge \pi^j_g \wedge \mathsf{safe}_{g,j} \equiv (\neg\delta_j \vee \ulcorner s >^{\pi}_{\mathsf{pop}*} t_j \urcorner) \wedge (\delta_j \rightarrow \pi^j_g \wedge \mathsf{safe}_{g,j})$$

it is easy to see that for the encoding of clause $(3b)$ given above,

$$\ulcorner s >^{\pi\ (3b)}_{\mathsf{pop}*} t \urcorner \equiv A \wedge \bigwedge_{j=1}^{m} \Big(\big((\neg\delta_j \vee \ulcorner s >^{\pi}_{\mathsf{pop}*} t_j \urcorner) \wedge B \big) \Big)$$

holds. By pushing the implication in $\pi^{\mathsf{list}}_g \rightarrow \ulcorner s >^{\pi\ (3b)}_{\mathsf{pop}*} t \urcorner$ inward, and by the equivalence $a \rightarrow b \equiv \neg a \vee b$, we obtain

$$\begin{aligned}
\pi^{\mathsf{list}}_g \rightarrow \ulcorner s >^{\pi\ (3b)}_{\mathsf{pop}*} t \urcorner &\equiv (\pi^{\mathsf{list}}_g \rightarrow A) \\
&\quad \wedge \bigwedge_{j=1}^{m} \Big((\neg\pi^{\mathsf{list}}_g \vee \neg\delta_j \vee \ulcorner s >^{\pi}_{\mathsf{pop}*} t_j \urcorner) \wedge \big(\pi^{\mathsf{list}}_g \rightarrow B\big) \Big) \ .
\end{aligned}$$

In a similar spirit, we derive

$$\neg\pi^{\mathsf{list}}_g \rightarrow \ulcorner s >^{\pi\ (3a)}_{\mathsf{pop}*} t \urcorner \equiv \bigwedge_{j=1}^{m} \big(\pi^{\mathsf{list}}_g \vee \neg\pi^j_g \vee \ulcorner s >^{\pi}_{\mathsf{pop}*} t_j \urcorner) \big) \ .$$

And so, we can reformulate the propositional formula (iii), the naive encoding of clause $(3a)$ and $(3b)$, as

$$\begin{aligned}
&\pi^{\mathsf{list}}_f \wedge \bigwedge_{j=1}^{m} \big(\pi^{\mathsf{list}}_g \vee \neg\pi^j_g \vee \ulcorner s >^{\pi}_{\mathsf{pop}*} t_j \urcorner \big) \\
&\qquad \wedge (\pi^{\mathsf{list}}_g \rightarrow A) \wedge \bigwedge_{j=1}^{m} \big((\neg\pi^{\mathsf{list}}_g \vee \neg\delta_j \vee \ulcorner s >^{\pi}_{\mathsf{pop}*} t_j \urcorner) \wedge \big(\pi^{\mathsf{list}}_g \rightarrow B\big) \big) \\
&\equiv \pi^{\mathsf{list}}_f \wedge (\pi^{\mathsf{list}}_g \rightarrow A) \wedge \bigwedge_{j=1}^{m} \big((\pi^{\mathsf{list}}_g \vee \neg\pi^j_g \vee \ulcorner s >^{\pi}_{\mathsf{pop}*} t_j \urcorner) \\
&\qquad\qquad \wedge (\neg\pi^{\mathsf{list}}_g \vee \neg\delta_j \vee \ulcorner s >^{\pi}_{\mathsf{pop}*} t_j \urcorner) \wedge (\pi^{\mathsf{list}}_g \rightarrow B) \big) \ .
\end{aligned}$$

The last formula still contains twice the subformula $\ulcorner s >^{\pi}_{\mathsf{pop*}} t_j \urcorner$. We get rid of this duplication by exploiting the equivalence

$$\begin{aligned}(\pi_g^{\mathsf{list}} \vee \neg \pi_g^j \vee \ulcorner s >^{\pi}_{\mathsf{pop*}} t_j \urcorner) \wedge (\neg \pi_g^{\mathsf{list}} \vee \neg \delta_j \vee \ulcorner s >^{\pi}_{\mathsf{pop*}} t_j \urcorner)& \\ \equiv \ulcorner s >^{\pi}_{\mathsf{pop*}} t_j \urcorner \vee (\neg \pi_g^{\mathsf{list}} \wedge \neg \pi_g^j) \vee (\pi_g^{\mathsf{list}} \wedge \neg \delta_j)&\end{aligned}$$

from which we derive the final constraint

$$\begin{aligned}\ulcorner s >^{\pi\ (3a,\, b)}_{\mathsf{pop*}} t \urcorner = \pi_f^{\mathsf{list}} \wedge \big(\pi_g^{\mathsf{list}} \to \ulcorner f \in \mathcal{D} \urcorner \wedge \ulcorner f > g \urcorner \wedge \mathrm{one}(\delta_1, \ldots, \delta_n)\big)& \\ \wedge \bigwedge_{j=1}^{m} \Big(\big(\ulcorner s >^{\pi}_{\mathsf{pop*}} t_j \urcorner \vee (\neg \pi_g^{\mathsf{list}} \wedge \neg \pi_g^j) \vee (\pi_g^{\mathsf{list}} \wedge \neg \delta_j)\big)& \\ \wedge \big(\pi_g^{\mathsf{list}} \to \big(\delta_j \to \pi_g^j \wedge \mathsf{safe}_{g,j}\big)& \\ \wedge \big(\neg \delta_j \to \neg \pi_g^j \vee \ulcorner s >^{\pi}_{\mathsf{pop}} t_j \urcorner \vee \wedge\, \mathsf{safe}_{g,j} \wedge \ulcorner s \rhd t_j \urcorner\big)\big)\Big) .&\end{aligned}$$

For the encoding of the final clause (3c) from the definition of $>^{\pi}_{\mathsf{pop*}}$, suppose we consider terms $s = f(s_1, \ldots, s_n)$ and $s = g(t_1, \ldots, t_m)$ such that the argument filtering π is non-collapsing for both f and g. Furthermore, assume $f \sim g$. In the encoding for this case, we need to assert both $\mathsf{nrm}_{\pi}(s) >^{\pi,\mathsf{mul}}_{\mathsf{pop*}} \mathsf{nrm}_{\pi}(t)$ and $\mathsf{safe}_{\pi}(s) \gtrsim^{\pi,\mathsf{mul}}_{\mathsf{pop*}} \mathsf{safe}_{\pi}(t)$. Similar to the encoding of the corresponding clause in Section 7.1, we make use of multiset covers (γ, ε) and basically encode

$$\mathsf{nrm}_{\pi}(s) \uplus \mathsf{safe}_{\pi}(s) \gtrsim^{\pi,\mathsf{mul}}_{\mathsf{pop*}} \mathsf{nrm}_{\pi}(t) \uplus \mathsf{safe}_{\pi}(t) .$$

Via the atoms $\gamma_{i,j}$, where a true assignment denotes that s_i is compared with t_j, we are able to enforce the separation of safe and normal arguments. Furthermore, we demand that at least one atom ε_i for some (not erased) normal argument position of f evaluates to a false value. This asserts a strict decrease at a normal argument position. In addition to the usual restrictions on the multiset cover (γ, ε), we need to assert that the cover spans only over argument positions not erased by the argument filtering. We enforce this using the propositional formula

$$\begin{aligned}\ulcorner (\gamma, \varepsilon) \urcorner = \bigwedge_{i=1}^{n} (\neg \pi_f^i \to \bigwedge_{j=1}^{m} \neg \gamma_{i,j}) \wedge \bigwedge_{j=1}^{m} (\neg \pi_g^j \to \bigwedge_{i=1}^{n} \neg \gamma_{i,j})& \\ \wedge \bigwedge_{j=1}^{m} (\pi_g^j \to \mathrm{one}(\gamma_{1,j}, \ldots, \gamma_{n,j})) \wedge \bigwedge_{i=1}^{n} (\varepsilon_i \to \mathrm{one}(\gamma_{i,1}, \ldots, \gamma_{i,n})) .&\end{aligned}$$

Here, the first line asserts that whenever the i-th argument position of f or the j-th argument position of g is erased by π, then t_j cannot be covered by s_i. With the second line, we assess that every subterm t_j of t that is not erased by π is covered by exactly one subterm s_i of s. Furthermore, when $s_i \sim t_j$ (indicated by a positive valuation of ε_i) then besides t_j no other element may be covered by s_i. Exactly as in the corresponding encoding described in Section

7.1, the formula

$$\begin{aligned}\ulcorner f(s_1,\dots,s_n) >^{(3c)}_{\mathsf{pop}*} g(t_1,\dots,t_n)\urcorner_\alpha = {}\\ \ulcorner f \in \mathcal{D}\urcorner \wedge \ulcorner f \sim g\urcorner \wedge \ulcorner(\gamma,\varepsilon)\urcorner \wedge \bigvee_{i=1}^{n} \big(\pi^i_f \wedge \neg\, \mathsf{safe}_{f,i} \wedge \neg\varepsilon_i\big)\\ \wedge \bigwedge_{i=1}^{n}\bigwedge_{j=1}^{n}\Big(\gamma_{i,j} \rightarrow (\varepsilon_i \rightarrow \ulcorner s_i \overset{\mathsf{s}}{\sim} t_j\urcorner_\alpha) \wedge (\neg\varepsilon_i \rightarrow \ulcorner s_i >_{\mathsf{pop}*} t_j\urcorner_\alpha)\\ \wedge\, (\mathsf{safe}_{f,i} \leftrightarrow \mathsf{safe}_{g,j})\Big)\end{aligned}$$

expresses the final clause from the definition of $>^{\pi}_{\mathsf{pop}*}$. Summing up, we define

$$\begin{aligned}&\ulcorner s >^{\pi}_{\mathsf{pop}*} t\urcorner = \ulcorner s >^{\pi}_{\mathsf{pop}} t\urcorner \vee \ulcorner s >^{\pi\,(1,2)}_{\mathsf{pop}*} t\urcorner \vee \ulcorner s >^{\pi\,(3a,b)}_{\mathsf{pop}*} t\urcorner \vee \ulcorner s >^{\pi\,(3c)}_{\mathsf{pop}*} t\urcorner \text{ and}\\ &\ulcorner s \gtrsim^{\pi}_{\mathsf{pop}*} t\urcorner = \ulcorner s >^{\pi}_{\mathsf{pop}*} t\urcorner \vee \ulcorner s \overset{\mathsf{s}}{\sim}_\pi t\urcorner\,.\end{aligned}$$

Let $\mathcal{R}$ and $\mathcal{S}$ be two rewrite system. In order to verify $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop}*}$ and $\mathcal{S} \subseteq \gtrsim^{\pi}_{\mathsf{pop}*}$ we orient every rewrite rule $\ell \rightarrow r$ using the propositional formulas $\ulcorner \ell >^{\pi}_{\mathsf{pop}*} r\urcorner$ and $\ulcorner \ell \gtrsim^{\pi}_{\mathsf{pop}*} r\urcorner$ respectively. Also, a valid encoding of the precedence and the safe argument filtering has to be enforced. Remember that $\ulcorner\pi\urcorner$ asserts a valid evaluation of the employed atoms π^{list}_f and π^i_f. In order to apply $>^{\pi}_{\mathsf{pop}*}$ in the context of a dependency pair analysis as described in section 6.1, the argument filtering $>^{\pi}_{\mathsf{pop}*}$ needs to be safe, that is $\pi(c) = [1,\dots,n]$ for any n-ary compound symbol c. For a set of compound symbols $\mathcal{C}$ we define

$$\mathrm{AF}(\mathcal{C}) = \bigwedge_{c\in\mathcal{C}} \pi^{\mathsf{list}}_c \wedge \bigwedge_{i=1}^{\mathrm{arity}(c)} \pi^i_c\,.$$

Let the propositional formula PREC be defined exactly as in Section 7.1. Putting things together, the propositional formula

$$\begin{aligned}\mathrm{POP}^*_{\mathrm{RP}}(\mathcal{R},\mathcal{S}) = \mathrm{PREC}(\mathcal{F}^\sharp \cup \mathcal{C}_{\mathrm{COM}}) \wedge \mathrm{AF}(\mathcal{C}_{\mathrm{COM}})\\ \wedge \bigwedge_{\ell\rightarrow r\in\mathcal{R}} \ulcorner \ell >^{\pi}_{\mathsf{pop}*} r\urcorner \wedge \bigwedge_{\ell\rightarrow r\in\mathcal{S}} \ulcorner \ell \gtrsim^{\pi}_{\mathsf{pop}*} r\urcorner\end{aligned}$$

certifies the existence of a safe mapping safe, precedence $\gtrsim$ and (safe) argument filtering π such that for the induced order $>^{\pi}_{\mathsf{pop}*}$, the inclusions $\mathcal{R} \subseteq >^{\pi}_{\mathsf{pop}*}$ and $\mathcal{S} \subseteq \gtrsim^{\pi}_{\mathsf{pop}*}$ holds.

7.3 Experimental Findings

Both encodings described in Section 7.1 and Section 7.2 have been integrated into the *Tyrolean Complexity Tool* (TCT for short). TCT is a complexity analyzer written in OCaml [50] for automatically proving lower and upper bounds on the derivational and runtime-complexity of term rewriting systems. It is strongly based on the automatic termination prover TTT2[2]. TCT, including the here described implementations of POP*, is open source and freely available under

[2]For more information on TTT2 see `http://colo6-c703.uibk.ac.at/ttt2/`.

`http://cl-informatik.uibk.ac.at/software/tct/` .

Our implementation of POP* extends TCT by a total of 2221 lines of code. Here interface-files are not take into account. From the total, 974 lines of code are devoted to the SAT-encoding of the polynomial path order in conjunction with semantic labeling over a finite, but else arbitrary, carrier. The implementation of the encoding of POP* with argument filterings takes another 499 lines of code. For the construction of the Boolean formulas we make use of the PLogic library from TTT2 integrated into TCT. This library has been extended by an interface for the construction and intermediate simplification of propositional formulas in a lazy fashion, as briefly described in Section 7.1. The lazy interface to PLogic takes another 236 lines of code. The remaining 512 lines of code provide various utility functions, including a general efficient SAT-encoding of argument filterings, precedences and safe mappings.

SAT-solvers expect their input in CNF. For a concise construction of the final CNF formula from our constraints we employ PLogic that implements the transformation proposed by Plaisted and Greenbaum [47] to obtain an equisatisfiable CNF linear in size. This approach is analogous to Tseitin's transformation [54] but additionally takes the plurality of atoms into account. Satisfying assignments for the resulting CNF are found by employing the state-of-the-art SAT solver MiniSat [21].

In order to apply the polynomial path order as (safe) "reduction pair" in the context of the dependency pair method for complexity analysis [29], we make use of the standard implementation of weak and weak innermost dependency pairs found in TCT. Also for the orientation of the usable rules with strongly linear interpretations, we employ the standard implementation from TCT, which works again via a reduction to SAT and essentially follows [24].

In its most simple setting, TCT is invoked via the shell command

```
# tct -s ⟨strategy⟩ ⟨file⟩
```

where ⟨strategy⟩ defines the employed strategy and ⟨file⟩ refers to the input file specifying the input TRS. The input file needs to be in conformance with the *Termination Problem Data Base file format*[3]. With the shell command as given above, TCT will either output YES or MAYBE, depending on whether the given strategy succeeds on the input file or not. A detailed proof output can be obtained by additionally specifying the command line option -p. In conjunction with the polynomial path order, the following strategies are of interest:

– "pop* [-sp] [-s $\langle n \rangle$] [-i]": Without any additional options the strategy "pop*" defines that the input system should be oriented using an instance of the polynomial path order. The optional switch -sp asserts that a strict precedence is employed. Although this limits the applicability, it results in a more concise translation to SAT.

  When the optional switch -s $\langle n \rangle$ is supplied, then the input system is transformed with semantic labeling over a finite carrier. The additional

[3]For more information revise `http://www.lri.fr/~marche/tpdb/format.html`.

argument $n \geqslant 2$ defines the carrier size of the model. Finally, the optional switch -i specifies that on failure, the carrier size of the employed model should be incrementally increased. Thus, the strategy "pop* -i" first tries to prove compatibility of the input system with an instance of the polynomial path order. If the orientation fails, TCT searches for a model with carrier $\{0, 1\}$ and tries to orient a labeled rewrite system. For that, it employs the transformation to SAT as given in Section 7.1. If this fails in turn, the carrier is set to $\{0, 1, 2\}$ and the process is repeated ad infimum. In order to abort the execution, TCT allows the specification of a timeout in square brackets. For instance, the strategy "pop* -i[60]" aborts execution after 60 seconds.

– "wdprp [-b $\langle n \rangle$] [-i]; pop*": In its most simple form "wdprp; pop*", this strategy instructs TCT to compute the weak dependency pairs and usable rules and applies POP* as a reduction pair. For that, it employs the transformation to SAT as given in Section 7.2. Furthermore compatibility of the usable rules with a strongly linear interpretation is asserted. The optional switch -b $\langle n \rangle$ with $n \geqslant 1$ specifies a bound on the employed coefficients, the optional switch -i advises TCT to incrementally increase the bound upon failure. The default bound is 63, that is all employed coefficients are encoded with 8 bits.

– "widprp [-b $\langle n \rangle$] [-i] [-fdp] [-fwdp]; pop*": This strategy works similar to the above strategy for weak dependency pairs, but instead weak innermost dependency pairs or standard dependency pairs are computed. The latter case applies when all compound symbols from the set of weak innermost dependency pairs are nullary. Via the switch -fdp construction of standard dependency pairs are enforced, and the switch -fwdp advises TCT to construct weak innermost dependency pairs.

We demonstrate our implementation on the rewrite system AG01/#3.23 from the termination competition database.

Example 7.1. TCT when run on the TRS $\mathcal{R}_{\mathsf{f}}$ from Example 6.24 with strategy pop* -i produces the following output:

```
# tct -p -s "pop* -i" tpdb-4.0/TRS/AG01/#3.23.trs
YES
TRS:
 { f(0(), y) -> 0(),
   f(s(x), y) -> f(f(x, y), y)}
 POP*:
   Predicative rewrite system:
    {
        f_sl=0(;0_sl=0(),y) -> 0_sl=0(),
     f_sl=1(s_sl=0(;x),y;) -> f_sl=0(;f_sl=0(;x,y),y),
     f_sl=1(s_sl=0(;x),y;) -> f_sl=0(;f_sl=1(x,y;),y)}

   Safe positions:        safe(s_sl=1^1) = [1],
                          safe(s_sl=0^1) = [1],
                          safe(f_sl=0^2) = [1, 2]
```

```
Precedence:           f_sl=1 > f_sl=0

Interpretation:       0^0:
                        0

                      f^2:
                       x_1  x_2  |
                        0    0   |0
                        0    1   |0
                        1    0   |0
                        1    1   |0

                      s^1:
                       x_1  |
                        0   |1
                        1   |1


Labeling:             0^0:
                        0

                      f^2:
                       x_1  x_2  |
                        0    0   |0
                        0    1   |0
                        1    0   |1
                        1    1   |1

                      s^1:
                       x_1  |
                        0   |0
                        1   |0
Qed
```

As visible from the output, TCT finds the labeled rewrite system $\mathcal{R}_{\mathsf{lab}}$ and suitable instance $>_{\mathsf{pop}*}$ exactly as given in Example 6.24. The labeled rewrite system is given in predicative notation, that is a term $f(t_1,\dots,t_n)$ is displayed as $f(t_{i_1},\dots,t_{i_p};t_{j_1},\dots,t_{j_q})$ where $\mathsf{safe}(f) = [i_1,\dots,i_p]$ and $\mathsf{nrm}(f) = [j_1,\dots,j_q]$ respectively. Notice that TCT employs the notation f_sl=a to denote the function symbol f with label a. Below the labeled rewrite system, the safe mapping and precedence defining the concrete instance $>_{\mathsf{pop}*}$ with $\mathcal{R}_{\mathsf{lab}} \subseteq >_{\mathsf{pop}*}$ is given. These are followed by a textual representation of the interpretation- and labeling functions.

7.4 Experimental Results

We have tested the implementation of our SAT-encodings on two testbeds: Testbed **C** constitutes of the 638 constructor TRSs of the 1393 examples from the Termination Problem Database Version 4.0[4] that were used in the runtime-complexity category of the termination competition 2008[5]. We have chosen testbed **C** as our main results are only stated for constructor TRSs. Furthermore, testbed **FP** is a meaningful subset of **C** in the sense that first order functional programs can be understood as orthogonal rewrite systems [10].

All experiments were performed on a standard PC with a Intel® Pentium™ IV processor of 2.66 GHz, 512kB L2 cache and 512Mb SD-RAM running under GNU Linux 2.6. For all experiments, we have set a timeout of 10 seconds. In the tables below, we highlight for each testbed and method the total number of yes-instances (that is those systems that can be oriented), the total number of maybe-instances (i.e. those examples where a proof fails) and the total number of timeouts. Furthermore, for each method we give average orientation time of yes-instances in seconds.

Below, we compare the polynomial path order to multiset path orders as implemented in $\mathsf{T_TT_2}$ and *simple-mixed, constructor-restricted polynomial interpretations* [13] as implemented in $\mathsf{T_CT}$. For SMC, defined symbols are interpreted by simple-mixed polynomials [13] (a restrictive form of quadratic polynomial interpretations), whereas constructors are interpreted by weight functions. The former comparison is meaningful as the polynomial path order is a tamed version of MPO. For the latter, it is trivial to show that SMC interpretations induce polynomial runtime complexities. Thus compatibility yields similar results to our techniques.

Table 7.1: Polynomial Path Order

		MPO	POP*		SMC
			strict	quasi	
C	Yes	66	38	40	78
	Maybe	572	600	598	368
	Timeout	0	0	0	137
	Average time	0.05	0.02	0.02	3.38
FP	Yes	59	33	35	65
	Maybe	321	347	345	221
	Timeout	0	0	0	94
	Average time	0.05	0.02	0.02	3.21

What is noteworthy is that almost two from three yes-instances of MPO can be handled by POP*directly. As MPO induces primitive recursive derivational complexity (and this bound is tight), from a complexity theoretic perspective compatibility with a polynomial path order yields a much stronger result at a reasonable expense. SMC polynomial interpretations outperform POP* if we compare numbers of yes-instances, independent on the testbed. On the

[4]Available at `http://www.lri.fr/~marche/tpdb`.

[5] `Cf.http://colo5-c703.uibk.ac.at:8080/termcomp.`

other hand, POP* is a purely syntactic technique. Those tend to scale better than semantic methods like polynomial interpretations. This is reflected if one compares the number of timeouts and the average execution times. Moreover, we also want to highlight the execution time of our implementation compared to MPO. Even thought the definition of POP* relies on two separate orderings, the average time needed for orientation of a rewrite system lies around 20 milliseconds. This is twice as fast as the implementation of MPO (and more that 150 times faster than SMC). Compared to MPO the gain in speed seems to be mainly due to the lazy interface to the PLogic library.

Table 7.2: Polynomial Path Order and Transformations

		dependency pairs		semantic labeling		
		WIDP	WDP	CS 2	CS 3	CS 4
C	Yes	55	45	74	75	70
	Maybe	580	590	547	408	238
	Timeout	3	3	17	155	330
	Average time	0.30	0.31	0.09	0.43	0.83
FP	Yes	47	39	64	65	61
	Maybe	331	339	300	167	226
	Timeout	1	2	16	148	309
	Average time	0.27	0.27	0.07	0.39	0.84

In Table 7.2 we present experimental results where the polynomial path order is applied together with the transformation techniques as described in Chapter 6. In the first two columns we highlight the application of the polynomial path on weak dependency pairs (column WDP) and weak innermost dependency pairs (column WIDP). Numbers drawn reflect the total on the union of yes-instances (maybe-instances, no-instances) given by the application of Proposition 6.6 and Corollary 6.15 respectively. More precise, each yes-instance $\mathcal{R}$ obeys either $\mathcal{P} \cup \mathcal{U}(\mathcal{P}) \subseteq >_{\mathsf{pop}*}$ or $\mathcal{U}(\mathcal{P}) \subseteq \gtrsim^{\pi}_{\mathsf{pop}*}$, $\mathcal{P} \subseteq >^{\pi}_{\mathsf{pop}*}$ and the side-conditions of Corollary 6.15. Here $\mathcal{P}$ refers to $\mathsf{WIDP}(\mathcal{R})$ or $\mathsf{WDP}(\mathcal{R})$ depending on the column. For the orientation, we use the propositional encoding of Section 7.2. The results indicate that polynomial path orders can certify polynomial runtime-complexities of 55 examples in combination with WIDPs, achieving slightly better results than in combination with WDPs.

Remark. Although reflected in our tests, polynomial path orders and weak innermost dependency pairs do not subsume polynomial path orders in combination with weak dependency pairs. For this, suppose the input system $\mathcal{R}$ contains the rule $\mathsf{f}(\mathsf{s}(x)) \to x$. Then $\mathsf{WIDP}(\mathcal{R})$ contains $\mathsf{f}^\sharp(\mathsf{s}(x)) \to \mathsf{c}$ for some compound symbol c, whereas $\mathsf{WDP}(\mathcal{R})$ contains $\mathsf{f}^\sharp(\mathsf{s}(x)) \to x$. Fix an argument filtering π such that $\pi(\mathsf{f}^\sharp) = 1$. Then $\mathsf{f}^\sharp(\mathsf{s}(x)) >^{\pi}_{\mathsf{pop}*} x$ independent on $>^{\pi}_{\mathsf{pop}*}$. To the contrary, the rule from $\mathsf{WIDP}(\mathcal{R})$ cannot be oriented, which is a consequence of Lemma 5.12.

In the last three columns of Table 7.2 we present the application of the polynomial path order together with finite semantic labeling and carrier sizes of two, three and four respectively. Here we employ the propositional encoding from Section 7.1 generalized to arbitrary finite carriers. Best results are achieved with carriers of size 3. Observe that increasing the carrier size above 3 results in a decrease of the total on yes-instances. Simultaneously number of timeouts significantly increase, which explains the drop in yes-instances. Remember that in the encoding, we need to apply the constraint $\ulcorner \ell >_{\mathsf{pop*}} r \urcorner_\alpha$ for every rule $\ell \to r$ and assignment α. That is, the encoding grows linear in the number of rules and assignments. The latter is given by s^n where s is the size of the carrier and n the number of variables from ℓ, which explains the high amount of timeouts in the last column.

We conclude this Chapter with Table 7.3, where we summarize results drawn in Table 7.1 and Table 7.2. In Table 7.3 we contrast SMC polynomial interpretations and polynomial path orders together with selected transformations. For a meaningful comparison of polynomial path orders and SMC polynomial interpretations, we additionally state experimental results of SMC interpretations in combination with weak innermost dependency pairs. Again we highlight experiments resulting from Proposition 6.6 and Proposition 6.7 with SMC in correspondence to POP* and WIDP. On the other hand, we do not combine SMC with finite semantic labeling, due to the following two reasons: SMC together with finite semantic labeling is hard to implement efficiently; we do not expect a practical increase in power by combining two semantical techniques.

Table 7.3: Polynomial Runtime-Complexity

		SMC			POP*			
		direct	WIDP	union	direct	WIDP	SL	union
C	Yes	78	83	83	40	55	75	91
	Maybe	368	337	–	598	580	408	–
	Timeout	137	218	–	0	3	155	–
	Average time	3.38	0.71	–	0.02	0.31	0.43	–
FP	Yes	65	70	70	35	39	65	78
	Maybe	221	166	–	345	339	167	–
	Timeout	94	144	–	0	2	148	–
	Average time	3.21	0.68	–	0.02	0.27	0.39	–

Our experimental results show that weak innermost dependency pairs give 15 additional constructor TRSs from testbed **C** that cannot be shown compatible with an instance $>_{\mathsf{pop*}}$ directly. For finite semantic labeling 35 examples are gained, and hence experimental results indicate that the strength of POP* is almost doubled by labeling. In Table 7.3 we additionally highlight the union of yes-instances, both for POP* and SMC. We stress that the combination of polynomial path orders with transformations clearly outperform SMC polynomial interpretations in power. The difference is even more significant if we compare the execution time. Our experiments show that the here proposed techniques certify a polynomial innermost runtime-complexity of 91 TRSs for testbed **C**, and 78 TRSs for testbed **FP** respectively. That is, from our testbeds polynomial path orders certify feasible complexity of approximately every seventh system.

8 Related Work

The polynomial path order is to some extent related to the *light multiset path order* (*LMPO* for short) introduced by Marion [40]. Roughly speaking the light multiset path order is a miniaturization of the multiset path order, characterizing the functions computable in polynomial time. To assert polytime computability, LMPO relies on non-standard evaluation techniques that involve eager evaluation and *memoization.* To the contrary, POP^* relies only on a standard concept from term rewriting, namely rewriting under the innermost rewrite relation (akin to eager evaluation of functional programs). As a particular consequence, transformation techniques can be employed to boost the strength of POP^*. For LMPO this is not entirely clear, as not rewriting steps but evaluation steps within the above mentioned framework are counted. However, at least for finite semantic labeling, it is expected that the transformation technique can be employed in conjunction with LMPO.

In [41] *quasi-friendly sup-interpretations* are introduced. As LMPO, the results presented in [41] fall into the realm of implicit complexity analysis, to characterize the polytime computable functions. Here *product path orders* (a restricted form of multiset path orders) and sup-interpretations are employed. The latter can be conceived as a special program interpretation. These seem reminiscent to [18], providing upper bound to recursively computed results. Experimental results from [9] indicate that quasi-friendly sup-interpretations and POP^* are of similar power on practical results. However as indicated above for LMPO, we think that our Main Theorem implies a theoretical and practical stronger result.

We have already compared the polynomial path order to SMC polynomial interpretations in Chapter 7. Like the polynomial path order, those interpretations directly yield polynomial runtime-complexities on compatible rewrite systems. As our experiments indicate, SMC polynomial interpretations outperform POP^* as a direct termination technique. To the contrary, by its syntactic nature the polynomial path order provides a much faster automation, and by incorporating transformations the polynomial path order beats SMC polynomial interpretations both in strength and in speed.

Our work seems also to some extend related to [2]. Here, it is assumed that an *affine size-change terminating program* P (cf. [1]) is transformed into an *abstract program* A, operating like P but on value sizes. From A a set of constraint reflecting the runtime-complexity of P is extracted. When A admits an *exact* polynomial runtime-complexity, that is when for all arguments $\vec{x}$ the runtime-complexity is exactly $p(\vec{x})$ for some polynomial p, from these constraints a *precise* symbolic representation of the complexity-function can be extracted. This seems to present a severe restriction. On the other hand, all steps can be performed in an automatic fashion, provided the abstract program A is given. Moreover, the approach extends naturally to other polynomial complexity measures.

Finally, although incomparable to our approach we also want to mention [33]. In this seminal work by Hofmann a functional language is introduced, carefully restricted such that all definable functions are computable in polynomial time. Notably, he allows the definition of *higher-order* functions in his language. Polytime computability is asserted by a clever choice on recursion operators and a type system that asserts linearity of expressions. It seems possible to use ideas from [33] that circumvent the limitations of predicative recursion also in our setting. This is subject to future research.

9 Conclusion

In this thesis we have covered the polynomial path order, a tamed version of the multiset path order that certifies polynomial runtime-complexities of rewrite systems. In particular, the inclusion $\mathcal{R} \subseteq >_{\mathsf{pop*}}$ certifies that the length ℓ of every sequence

$$f(v_1, \ldots, v_n) \xrightarrow{\mathrm{i}}_{\mathcal{R}} t_1 \xrightarrow{\mathrm{i}}_{\mathcal{R}} \cdots \xrightarrow{\mathrm{i}}_{\mathcal{R}} t_\ell$$

with values v_i is polynomially bounded in the sizes of v_i. As a particular application, this allows in principle the (automated) complexity analysis of strict functional programs. The evaluation of such programs is naturally captured by the innermost rewriting relation. Moreover, the polynomial path order is complete in the sense that the class of polytime computable functions can be identified with the class of those functions computed by certain syntactically restricted rewrite systems compatible with POP^*.

In Chapter 6 we have combined the polynomial path order with two prominent transformation techniques originally invented for the termination analysis. As a preparation step, we have shown that the polynomial path order can be employed for estimating polynomial derivation lengths with respect to (innermost) relative rewrite steps. For the special case of relative top-steps, argument filterings can be additionally employed. Counter-intuitively, the extension to argument filterings is a non-trivial task. By slightly extending the results established for relative rewriting, we have shown that a pair $(\gtrsim^{\pi}_{\mathsf{pop*}}, >^{\pi}_{\mathsf{pop*}})$ can be employed as (safe) "reduction pair" in the context of [29]. Furthermore, we briefly investigated on the combination of polynomial path order combined with finite semantic labeling.

All those techniques give rise to an efficient implementation. In particular, we have shown that compatibility of polynomial path orders (together with transformations) can be reduced to SAT. All necessary steps can be performed completely automatic. Experimental results confirm the feasibility of the here presented approaches.

Bibliography

[1] Hugh Anderson and Siau-Cheng Khoo. Affine-based size-change termination. In *Proceedings of the 3th Asian Symposium on Programming Languages and Systems*, volume 2895 of *Lecture Notes in Computer Science*, pages 122–140, 2003.

[2] Hugh Anderson and Siau-Cheng Khoo. Calculating polynomial runtime properties. In *Proceedings of the 5th Asian Symposium on Programming Languages and Systems*, volume 3870 of *Lecture Notes in Computer Science*, pages 230–246, 2005.

[3] Elena Annov, Michael Codish, Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl. Implementing RPO and POLO using SAT. In *Proceedings of Deduction and Decision Procedures*, number 07401 in Dagstuhl Seminar Proceedings, 2007.

[4] Toshiyasu Arai and Georg Moser. Proofs of termination of rewrite systems for polytime functions. In *Proceedings of the 25th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 3821 of *Lecture Notes in Computer Science*, pages 529–540, 2005.

[5] Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1–2):133–178, 2000.

[6] Martin Avanzini. Scheme programs with polynomially bounded evaluation length. Bachelor Thesis, University of Innsbruck, Faculty for Computer Science.

[7] Martin Avanzini and Georg Moser. Complexity analysis by rewriting. In *Proceedings of the 9th International Symposium on Functional and Logic Programming*, volume 4989 of *Lecture Notes in Computer Science*, pages 130–146, 2008.

[8] Martin Avanzini and Georg Moser. Complexity analysis by rewriting. Technical report, University of Innsbruck, 2008. Available under `http://cl-informatik.uibk.ac.at/~georg/list.publications.html`.

[9] Martin Avanzini, Georg Moser, and Andreas Schnabl. Automated implicit computational complexity analysis (system description). In *Proceedings of the 4th International Joint Conference on Automated Reasoning*, volume 5195 of *Lecture Notes in Computer Science*, pages 132–138, 2008.

[10] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[11] Arnold Beckmann and Andreas Weiermann. A term rewriting characterization of the polytime functions and related complexity classes. *Archive for Mathematical Logic*, 36:11–30, 1996.

[12] Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2(2):97–110, 1992.

[13] Guillaume Bonfante, Adam Cichon, Jean-Yves Marion, and Hélène Touzet. Algorithms with polynomial interpretation termination proof. *Journal of Functional Programming*, 11(1):33–53, 2001.

[14] Guillaume Bonfante, Jean-Yves Marion, and Jean-Yves Moyen. Quasi-interpretations and Small Space Bounds. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications*, volume 3467 of *Lecture Notes in Computer Science*, pages 150–164, 2005.

[15] Wilfried Buchholz. Proof-theoretic analysis of termination proofs. *Anals of Pure and Applied Logic*, 75(1-2):57–65, 1995.

[16] Vuokko-Helena Caseiro. *Equations for defining poly-time functions*. PhD thesis, University of Oslo, Faculty of Mathematics and Natural Sciences, 1997.

[17] Adam Cichon and Andreas Weiermann. Term rewriting theory for the primitive recursive functions. *Anals of Pure and Applied Logic*, 83(3):199–223, 1997.

[18] Alan Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 International Congress for Logic, Methodology and the Philosophy of Science*, pages 24–30, 1964.

[19] Michael Codish, Vitaly Lagoon, and Peter J. Stuckey. Solving partial order constraints for LPO termination. In *Proceedings of the 17th International Conference on Rewriting Techniques and Applications*, volume 4098 of *Lecture Notes in Computer Science*, pages 4–18, 2006.

[20] Nachum Dershowitz. Orderings for term-rewriting systems. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 123–131, 1979.

[21] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518, 2003.

[22] Jörg Endrullis, Johannes Waldmann, and Hans Zantema. Matrix interpretations for proving termination of term rewriting. *Jornal of Automated Reasoning*, 40(2–3):195–220, 2008.

[23] M.C. Fernández Ferreira. *Termination of Term Rewriting. Well-foundedness, Totality and Transformations.* PhD thesis, University of Utrecht, Faculty for Computer Science, 1995.

[24] Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl. SAT solving for termination analysis with polynomial interpretations. In *Proceedings of the 10th International Conference on Theory and Applications of Satisfiability Testing*, volume 4501 of *Lecture Notes in Computer Science*, pages 340–354, 2007.

[25] Alfons Geser. *Relative Termination.* PhD thesis, University of Passau, Faculty for Mathematics and Computer Science, 1990.

[26] Alfons Geser, Dieter Hofbauer, Johannes Waldmann, and Hans Zantema. On tree automata that certify termination of left-linear term rewriting systems. *Information and Computation*, 205(4):512–534, 2007.

[27] Jürgen Giesl, Stephan Swiderski, Peter Schneider-Kamp, and René Thiemann. Automated termination analysis for Haskell: From term rewriting to programming languages. In *Proceedings of the 17th International Conference on Rewriting Techniques and Application*, volume 4098 of *Lecture Notes in Computer Science*, pages 297–312, 2006.

[28] Nao Hirokawa and Aart Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1–2):172–199, 2005.

[29] Nao Hirokawa and Georg Moser. Automated complexity analysis based on the dependency pair method. In *Proceedings of the 4th International Joint Conference on Automated Reasoning*, volume 5195 of *Lecture Notes in Computer Science*, pages 364–380, 2008.

[30] Nao Hirokawa and Georg Moser. Complexity, graphs, and the dependency pair method. In *Proceedings of the 15th International on Logic Programming, Artificial Intelligence, and Reasoning*, volume 5330 of *Lecture Notes in Computer Science*, pages 667–681, 2008.

[31] Dieter Hofbauer. Termination proofs by multiset path orderings imply primitive recursive derivation lengths. *Theoretical Computer Science*, 105(1):129–140, 1992.

[32] Dieter Hofbauer and Clemens Lautemann. Termination proofs and the length of derivations. In *Proceedings of the 3th International Conference on Rewriting Techniques and Applications*, volume 355 of *Lecture Notes in Computer Science*, pages 167–177, 1989.

[33] Martin Hofmann. Linear types and non-size-increasing polynomial time computation. *Information and Computation*, 138:57–85, 2003.

[34] Adam Koprowski and Aart Middeldorp. Predictive labeling with dependency pairs using SAT. In *Proceedings of the 21th International Conference on Automated Deduction*, volume 4603 of *Lecture Notes in Computer Science*, pages 410–425, 2007.

[35] Adam Koprowski and Hans Zantema. Automation of recursive path ordering for infinite labeled rewrite systems. In *Proceedings of the 2th International Joint Conference on Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 332–346, 2006.

[36] Masahito Kurihara and Hisashi Kondo. Efficient BDD encodings for partial order constraints with application to expert systems in software verification. In *Proceedinsg of the 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, volume 3029 of *Lecture Notes in Computer Science*, pages 827–837, 2004.

[37] Keiichirou Kusakari, Masaki Nakamura, and Yoshihito Toyama. Argument filtering transformation. In *Proceedings of the 1th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming*, volume 1702 of *Lecture Notes in Computer Science*, pages 47–61, 1999.

[38] Dallas. S. Lankford. On proving term rewriting systems are noetherian. Technical report, Technical University of Louisiana, 1979.

[39] Ingo Lepper. Derivation lengths and order types of Knuth-Bendix orders. *Theoretical Computer Science*, 269:433–450, 2001.

[40] Jean-Yves Marion. Analysing the implicit complexity of programs. *Information and Computation*, 183:2–18, 2003.

[41] Jean-Yves Marion and Romain Péchoux. Quasi-friendly sup-interpretations. *Computing Research Repository*, abs/cs/0608020, 2006. informal publication.

[42] Jean-Yves Marion and Romain Péchoux. Analyzing the implicit computational complexity of object-oriented programs. In *Proceedings of the 28th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 08004 of *Dagstuhl Seminar Proceedings*, 2008.

[43] Aart Middeldorp, Hitoshi Ohsaki, and Hans Zantema. Transforming termination by self-labelling. In *Proceedings of the 13th International Conference on Automated Deduction*, volume 1104 of *Lecture Notes in Computer Science*, pages 373–387, 1996.

[44] Georg Moser. Derivational complexity of knuth-bendix orders revised. In *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 4246 of *Lecture Notes in Computer Science*, pages 75–89, 2006.

[45] Enno Ohlebusch. *Advanced Topics in Term Rewriting.* Springer, 2002.

[46] Isabel Oitavem. New recursive characterizations of the elementary functions and the functions computable in polynomial space. *Revista Mathematica de la Universidad Complutense de Madrid*, 10:109–125, 1997.

[47] David A. Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2(3):293–304, 1986.

[48] Peter Schneider-Kamp, René Thiemann, Elena Annov, Michael Codish, and Jürgen Giesl. Proving termination using recursive path orders and SAT solving. In *Proceedings of the 6th International Symposium Frontiers of Combining Systems*, volume 4720 of *Lecture Notes in Computer Science*, pages 267–282, 2007.

[49] Helmut Schwichtenberg. An arithmetic for polynomial-time computation. *Theoretical Computer Science*, 357(1–3):202–214, 2006.

[50] Joshua B. Smith. *Practical OCaml.* Apress, 2006.

[51] Christian Sternagel and Aart Middeldorp. Root-labeling. In *Proceedings of the 19th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science, pages 336–350, 2008.

[52] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 2003.

[53] René Thiemann. *The DP Framework for Proving Termination of Term Rewriting.* PhD thesis, University of Aachen, Department of Computer Science, 2007.

[54] Grigori S. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic II*, pages 115–125, 1968.

[55] Andreas Weiermann. Termination proofs for term rewriting systems with lexicographic path ordering imply multiply recursive derivation lengths. *Theoretical Computer Science*, 139:355–362, 1995.

[56] Harald Zankl, Nao Hirokawa, and Aart Middeldorp. Constraints for argument filterings. In *Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science*, volume 4362 of *Lecture Notes in Computer Science*, pages 579–590, 2007.

[57] Harald Zankl and Aart Middeldorp. Satisfying KBO constraints. In *Proceedings of the 18th International Conference on Rewriting Techniques and Applications*, volume 4533 of *Lecture Notes in Computer Science*, pages 389–403, 2007.

[58] Hans Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24(1/2):89–105, 1995.

[59] Hans Zantema. Torpa: Termination of rewriting proved automatically. In *Proceedings of the 15th International Conference on Rewriting Techniques and Application*, volume 3091 of *Lecture Notes in Computer Science*, pages 257–266, 2006.

A Appendix

This chapter is a quick note on the adaption of polynomial path orders to non-constructor TRSs.

A.1 An extension of Polynomial Path Orders

Throughout the thesis, we have assumed that the input system is a constructor TRS. Indeed, this assumption is necessary for the definition of polynomial path orders as presented here. Consider the following example.

Example A.1. Consider the following TRS $\mathcal{R}_{\mathsf{e}}$ written in predicative notation:

$$\begin{aligned}
\mathsf{e}(x;\,) &\to \mathsf{f}(;\mathsf{g}(x;\,)) \\
\mathsf{f}(;\mathsf{g}(\mathsf{s}(;x);\,)) &\to \mathsf{d}(;\mathsf{g}(x;\,)) \\
\mathsf{d}(;x) &\to \mathsf{cons}_1(;x,\mathsf{f}(;x)) \\
\mathsf{cons}_1(;x,y) &\to \mathsf{cons}_2(;\mathsf{f}(;x),y) \\
\mathsf{g}(\bot;\,) &\to \bot
\end{aligned}$$

The above rewrite system is not a constructor TRS, as g is defined. Let safe be the safe mapping as indicated above, and define the precedence $\succ$ such that

$$\mathsf{e} \succ \mathsf{g} \succ \mathsf{d} \succ \mathsf{cons}_1 \succ \mathsf{cons}_2 \succ \mathsf{f} \ .$$

Then $\mathcal{R}_{\mathsf{e}} \subseteq >_{\mathsf{pop}*}$ for $>_{\mathsf{pop}*}$ as induced by $\succ$ and safe.

We proof that $\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}_{\mathsf{e}}}(n)$ is at least exponential. For this, it suffices to show

$$\mathrm{dl}(\mathsf{e}(\mathsf{s}^m(0)), \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{e}}}) > \mathrm{dl}(\mathsf{f}(\mathsf{g}(\mathsf{s}^m(0))), \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{e}}}) \geqslant 2^m \ .$$

We continue by induction on m. The base case is trivial, so assume that by induction hypothesis $\mathrm{dl}(\mathsf{f}(\mathsf{g}(\mathsf{s}^m(0))), \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{e}}}) \geqslant 2^m - 1$, and we want to show $\mathrm{dl}(\mathsf{f}(\mathsf{g}(\mathsf{s}^{m+1}(0))), \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{e}}}) \geqslant 2^{m+1} - 1$. Observe that $\mathsf{g}(\mathsf{s}^i(0)) \in \mathsf{NF}(\mathcal{R}_{\mathsf{e}})$ for $i \in \mathbb{N}$, and thus

$$\begin{aligned}
\mathsf{f}(\mathsf{g}(\mathsf{s}^{m+1}(0))) &\xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{e}}} \mathsf{d}(\mathsf{g}(\mathsf{s}^m(0))) \\
&\xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{e}}} \mathsf{cons}_1(\mathsf{g}(\mathsf{s}^m(0)), \mathsf{f}(\mathsf{g}(\mathsf{s}^m(0)))) \\
&\xrightarrow{\mathsf{i}}{}^{\geq\, 2^m-1}_{\mathcal{R}_{\mathsf{e}}} \mathsf{cons}_1(\mathsf{g}(\mathsf{s}^m(0)), \mathsf{f}(\mathsf{g}(\mathsf{s}^m(0)))\!\downarrow_{\mathcal{R}_{\mathsf{e}}}) \\
&\xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{e}}} \mathsf{cons}_2(\mathsf{f}(\mathsf{g}(\mathsf{s}^m(0))), \mathsf{f}(\mathsf{g}(\mathsf{s}^m(0)))\!\downarrow_{\mathcal{R}_{\mathsf{e}}}) \\
&\xrightarrow{\mathsf{i}}{}^{\geq\, 2^m-1}_{\mathcal{R}_{\mathsf{e}}} \mathsf{cons}_2(\mathsf{f}(\mathsf{g}(\mathsf{s}^m(0)))\!\downarrow_{\mathcal{R}_{\mathsf{e}}}, \mathsf{f}(\mathsf{g}(\mathsf{s}^m(0)))\!\downarrow_{\mathcal{R}_{\mathsf{e}}}) \ .
\end{aligned}$$

The claim follows.

However, we can rectify the situation by slightly altering the definition of polynomial path orders. Here, instead of separating defined symbols and constructors, we partition the signature into two sets $\mathcal{G}_{\mathcal{C}}$ and $\mathcal{G}_{\mathcal{D}}$. Intuitively, the set $\mathcal{G}_{\mathcal{C}}$ plays the rõle of constructors, whereas $\mathcal{G}_{\mathcal{D}}$ plays the rõle of the set of defined symbols. Throughout the following, we fix an arbitrary TRS $(\mathcal{R}, \mathcal{F})$. Furthermore, we use $\mathcal{C}$ to refer to the constructors from $\mathcal{F}$, and assume that $\{\mathrm{root}(\ell) \mid \ell \to r \in \mathcal{R}\} \cap \mathcal{C} = \varnothing$.

Definition A.2. We define the set $\mathcal{G}_{\mathcal{C}}(\mathcal{R})$ as the least set such that

1. $\mathcal{C} \subseteq \mathcal{G}_{\mathcal{C}}(\mathcal{R})$, and
2. $\bigcup \mathsf{Fun}(\ell_i) \subseteq \mathcal{G}_{\mathcal{C}}(\mathcal{R})$ for every rule $f(\ell_1, \ldots, \ell_n) \to r \in \mathcal{R}$.

Furthermore, we define $\mathcal{G}_{\mathcal{D}}(\mathcal{R}) := \mathcal{F} \setminus \mathcal{G}_{\mathcal{C}}(\mathcal{R})$.

For brevity, we write $\mathcal{G}_{\mathcal{D}}$ and $\mathcal{G}_{\mathcal{C}}$ instead of $\mathcal{G}_{\mathcal{D}}(\mathcal{R})$ and $\mathcal{G}_{\mathcal{C}}(\mathcal{R})$ respectively.

Definition A.3. We define $\sqsupset^{\pi}_{\mathsf{pop}}$ and $\sqsupset^{\pi}_{\mathsf{pop*}}$ in correspondence to $>^{\pi}_{\mathsf{pop}}$ from Definition 5.8 and $>^{\pi}_{\mathsf{pop*}}$ from Definition 5.9 respectively, where we replace the set of defined symbols $\mathcal{D}$ by $\mathcal{G}_{\mathcal{D}}$. Moreover, we call the precedence $\succsim$ *admissible* if it respects the separation $\mathcal{F} = \mathcal{G}_{\mathcal{C}} \uplus \mathcal{G}_{\mathcal{D}}$, cf. Definition 4.2. We write $\sqsupseteq^{\pi}_{\mathsf{pop*}}$ for $\sqsupset^{\pi}_{\mathsf{pop*}} \cup \overset{\mathsf{s}}{\sim}_{\pi}$.

In the following, we suppose $\sqsupset^{\pi}_{\mathsf{pop*}}$ is defined in terms of an admissible precedence in the sense of Definition A.3.

Definition A.4. In correspondence to the set of values Val, we define $\mathsf{Val}' = \mathcal{T}(\mathcal{G}_{\mathcal{C}}, \mathcal{V})$. Moreover, we set $\mathcal{T}'_{\mathsf{b}} = \{f(v_1, \ldots, v_n) \mid f \in \mathcal{G}_{\mathcal{D}} \land v_i \in \mathsf{Val}'\}$.

Example A.5. Reconsider Example A.1 from the beginning of this chapter. We have $\mathcal{G}_{\mathcal{C}} = \{\mathsf{g}, \mathsf{s}\}$. We cannot conclude $\mathcal{R}_{\mathsf{e}} \subseteq \sqsupset_{\mathsf{pop*}}$ since we fail to orient the second rule independent on the precedence.

Similar to Chapter 5, we now embed innermost rewrite steps into $\blacktriangleright_k$ for sufficiently large k. We adapt the definition of predicative interpretations by replacing Val by Val':

Definition A.6. A *predicative interpretation* is a pair $(\mathsf{S}', \mathsf{N}')$ of mappings

$$\mathsf{S}', \mathsf{N}' : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{S}\mathrm{eq}(\mathcal{F}^{\mathsf{n}} \cup \{\mathsf{s}\}, \mathcal{V})$$

defined as follows:

$$\mathsf{S}'(t) = \begin{cases} \varnothing & \text{if } t \in \mathsf{Val}' \\ (f^{\mathsf{n}}(\mathsf{N}'(t_{j_1}), \ldots, \mathsf{N}'(t_{j_p}))\ \mathsf{S}'(t_{i_1})\ \cdots\ \mathsf{S}'(t_{i_q})) & \text{if } t = f(t_1, \ldots, t_n). \end{cases}$$
$$\mathsf{N}'(t) = (\mathsf{S}'(t)) \mathbin{@} \mathsf{BN}(t)$$

Here $\mathsf{safe}(f) = \{i_1, \ldots, i_q\}$ and $\mathsf{nrm}(f) = \{j_1, \ldots, j_p\}$.

Finally, we also adapt definition 5.17 accordingly:

Definition A.7. Let $\mathcal{Q} = \{f(x_1, \dots, x_n) \to \bot \mid f \in \mathcal{G}_\mathcal{D}\}$ for some fresh constant $\bot$. We define

$$\xrightarrow{\mathsf{v}'}_\mathcal{R} = \xrightarrow{\mathcal{Q}}_\mathcal{R} \text{ and } \xrightarrow{\mathsf{v}'}{}^{\varepsilon}_{\mathcal{R}/\mathcal{S}} = \xrightarrow{\mathsf{v}'}_\mathcal{S} \cdot \xrightarrow{\mathsf{v}'}{}^{\varepsilon}_\mathcal{R} \cdot \xrightarrow{\mathsf{v}'}_\mathcal{S}$$

Similar as in Chapter 5, we can simulate innermost steps by $\xrightarrow{\mathsf{v}'}$:

Definition A.8. Without loss of generality, suppose $\bot \in \mathcal{F}$ is a constructor symbol not appearing in $\mathcal{R}$. We define the system $U'(\mathcal{R})$ as

$$U'(\mathcal{R}) = \{f(t_1, \dots, t_n) \to \bot \mid f(t_1, \dots, t_n) \in \mathsf{NF}(\mathcal{R}) \text{ and } f \in \mathcal{G}_\mathcal{D}\} \; .$$

Lemma A.9. *Let $\mathcal{R}$ and $\mathcal{S}$ be two TRSs. Then for $\mathcal{S}' = \mathcal{S} \cup U'(\mathcal{R} \cup \mathcal{S})$,*

$$s \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} t \Longrightarrow s{\downarrow}_{U'(\mathcal{R}\cup\mathcal{S})} \xrightarrow{\mathsf{v}'}_{\mathcal{R}/\mathcal{S}'} t{\downarrow}_{U'(\mathcal{R}\cup\mathcal{S})} \; .$$

Proof. The lemma follows by exactly same reasoning as in Lemma 5.20. ☐

The reformulation of values implies a partitioning of $\mathcal{R}$ as follows.

Definition A.10. We set $\mathcal{R}^{\rhd} = \{\ell \to r \mid \ell \in \mathsf{Val}' \wedge \ell \to r \in \mathcal{R}\}$ and $\mathcal{R}' = \mathcal{R} \setminus \mathcal{R}^{\rhd}$.

A crucial observation is that for $\mathcal{R}'$ it follows that for every left-hand side ℓ, $\ell \in \mathcal{T}'_{\mathsf{b}}$. Moreover, by definition every left-hand side from $\mathcal{R}^{\rhd}$ is an element from Val'. Observe that we can adopt Lemma 5.12 as follows:

Lemma A.11. *For $\pi(s) \in \mathsf{Val}'$, if $s >^{\pi}_{\mathsf{pop*}} t$ then for some position $p \neq \varepsilon$, $\pi(s)|_p \overset{s}{\sim} \pi(t)$.*

And thus, if we have $\mathcal{R} \subseteq \sqsupset^{\pi}_{\mathsf{pop*}}$ then for $\ell \to r \in \mathcal{R}^{\rhd}$ it follows that $\pi(r) \in \mathsf{Val}'$. Hence top-steps due to $\mathcal{R}^{\rhd}$ can be easily embedded into $\blacktriangleright_k$ via the interpretation N'.

Lemma A.12. *Assume $\mathcal{R} \subseteq \sqsupset^{\pi}_{\mathsf{pop*}}$, and suppose $s \xrightarrow{}{}^{\varepsilon}_{\mathcal{R}^{\rhd}} t$. Then*

(1) $\mathsf{S}'(\pi(s)) \underset{\sim}{\blacktriangleright}_1 \mathsf{S}'(\pi(t))$*, and*

(2) $\mathsf{N}'(\pi(s)) \blacktriangleright_1 \mathsf{N}'(\pi(t))$*.*

Proof. Suppose $s = \ell\sigma \xrightarrow{\mathcal{R}}{}^{\varepsilon}_{\mathcal{R}^{\rhd}} r\sigma = t$ for some rule $\ell \to r \in \mathcal{R}^{\rhd}$. From $\mathcal{R}^{\rhd} \subseteq \sqsupset^{\pi}_{\mathsf{pop*}}$ together with Lemma A.11 we see that $\pi(r) \in \mathsf{Val}'$. And hence, when $\pi(s) \in \mathsf{Val}'$, then $\pi(t) \in \mathsf{Val}'$ and $\mathsf{S}'(\pi(s)) = \varnothing = \mathsf{S}'(\pi(t))$ follows. For the case $\pi(s) \notin \mathsf{Val}'$, then either $\mathsf{S}'(\pi(t)) = \varnothing$ or $\mathsf{S}'(\pi(s)) \rhd \mathsf{S}'(\pi(t))$. For both cases it is easy to derive the desired result. Finally, as $\|\pi(s)\| > \|\pi(t)\|$, we also derive $\mathsf{N}'(\pi(s)) \blacktriangleright_1 \mathsf{N}'(\pi(t))$. ☐

Lemma A.13. *Assume $\mathcal{R} \subseteq \sqsupset^{\pi}_{\mathsf{pop*}}$. Then $s \to_{\mathcal{R}^{\rhd}} t \Longrightarrow \mathsf{Q}(\pi(s)) \underset{\sim}{\blacktriangleright}_1 \mathsf{Q}(\pi(t))$ for $\mathsf{Q} \in \{\mathsf{S}', \mathsf{N}'\}$.*

Proof. The Lemma follows by induction with the help of Lemma A.12. In correspondence to Lemma 5.29 we additionally employ $\|\pi(s)\| \geqslant \|\pi(t)\|$ here. ☐

Lemma A.14. *Assume $\mathcal{R} \subseteq \sqsupset^{\pi}_{\mathsf{pop}*}$. Then for k depending only on $\mathcal{R}'$,*

(1) $s \xrightarrow{\mathsf{v}'}{}^{\varepsilon}_{\mathcal{R}'} t \Longrightarrow \mathsf{N}'(\pi(s)) \blacktriangleright_k \mathsf{N}'(\pi(t))$, and

(2) $s \xrightarrow{\mathsf{v}'}_{\mathcal{R}'} t \Longrightarrow \mathsf{N}'(\pi(s)) \underset{\sim}{\blacktriangleright}_k \mathsf{N}'(\pi(t))$.

Proof. This is just Lemma 5.28 and Lemma 5.29 reformulated. □

Summing up, we can proof the following:

Lemma A.15. *If $\mathcal{R} \subseteq \sqsupset^{\pi}_{\mathsf{pop}*}$, then for k depending only on $\mathcal{R}$, $s \xrightarrow{\mathsf{v}'}{}^{\varepsilon}_{\mathcal{R}} t \Longrightarrow \mathsf{N}'(\pi(s)) \blacktriangleright_k \mathsf{N}'(\pi(t))$. Moreover, if $\mathcal{R} \subseteq \sqsupseteq^{\pi}_{\mathsf{pop}*}$, then for k depending only on $\mathcal{R}$, $s \xrightarrow{\mathsf{v}'}_{\mathcal{R}} t \Longrightarrow \mathsf{N}'(\pi(s)) \underset{\sim}{\blacktriangleright}_k \mathsf{N}'(\pi(t))$.*

Proof. The Lemma is almost a consequence of Lemma A.13 and Lemma A.15. The only new case is $\ell \overset{\mathsf{s}}{\sim}_{\pi} r$ for some rule $l \to r \in \mathcal{R}$. For this case, the result follows as $\ell \overset{\mathsf{s}}{\sim}_{\pi} r$ (where $\overset{\mathsf{s}}{\sim}_{\pi}$ respect the separation of $\mathcal{F}$ into $\mathcal{G}_{\mathcal{C}}$ and $\mathcal{G}_{\mathcal{D}}$) implies $\mathsf{N}^{\mathsf{s}}(C[s\sigma]) \sim \mathsf{N}^{\mathsf{s}}(C[t\sigma])$ for any context C, substitution σ and $\mathsf{N}^{\mathsf{s}} \in \{\mathsf{S}', \mathsf{N}'\}$. □

A.2 Counting Dependency Pair Steps

The above observations suffice for an embedding of $\xrightarrow{\mathsf{i}}{}^{\varepsilon}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}$ into $\blacktriangleright^{+}_{k}$. We can strengthen the observation for weak and weak innermost dependency pairs, provided we only consider derivations starting from $t \in \mathcal{T}^{\#}_{\mathsf{b}}$. In correspondence to Section 6.1 we adapt interpretation N' for compound symbols:

Definition A.16. The *extended predicative interpretation*

$$\mathsf{N'}_{\mathsf{s}} : \mathcal{T}(\mathcal{F} \cup \mathcal{C}_{\text{COM}}, \mathcal{V}) \to \mathcal{S}\text{eq}(\mathcal{F}^{\mathsf{n}} \cup \{\mathsf{s}\}, \mathcal{V})$$

is defined as follows:

$$\mathsf{N'}_{\mathsf{s}}(t) = \begin{cases} (\mathsf{N'}_{\mathsf{s}}(t_1) \ \cdots \ \mathsf{N'}_{\mathsf{s}}(t_n)) & \text{if } t = c(t_1, \dots, t_n) \text{ and } c \text{ is a compound symbol} \\ (\mathsf{N}'(t)) & \text{otherwise.} \end{cases}$$

In the following, let $\mathcal{P}$ denote the weak or weak innermost dependency pairs of $\mathcal{R}$.

Lemma A.17. *Suppose $\mathcal{P}$ is compatible with $\sqsupset^{\pi}_{\mathsf{pop}*}$, i.e. suppose $\mathcal{P} \subseteq \sqsupset^{\pi}_{\mathsf{pop}*}$ holds. Then, for sufficiently large k depending only on $\mathcal{R}$,*

$$s \xrightarrow{\mathsf{v}'}{}^{\varepsilon}_{\mathcal{P}} t \Longrightarrow \mathsf{N'}_{\mathsf{s}}(\pi(s)) \blacktriangleright_k \mathsf{N'}_{\mathsf{s}}(\pi(t)) \ .$$

Proof. The proof equals the proof of Lemma 6.11, where we replace the appication of Lemma 5.28 by Lemma A.15. □

Lemma A.18. *Let $\mathcal{R}$ be a TRS compatible with $\sqsupseteq^{\pi}_{\mathsf{pop}*}$, i.e $\mathcal{R} \subseteq \sqsupseteq^{\pi}_{\mathsf{pop}*}$ holds. Let $\mathsf{Q} \in \{\mathsf{S}, \mathsf{N}\}$. Then for k depending only on $\mathcal{R}$,*

$$s \xrightarrow{\mathsf{v}}_{\mathcal{R}} t \Longrightarrow \mathsf{N'}_{\mathsf{s}}(\pi(s)) \underset{\sim}{\blacktriangleright}_k \mathsf{N'}_{\mathsf{s}}(\pi(t)) \ .$$

Proof. The Lemma follows by similar reasoning as in Lemma 6.13. Here we simply replace the application of Lemma 5.29 by Lemma A.15. □

Finally, for contexts build from compound symbols we derive an embedding into the strict relation $\blacktriangleright_k$:

Lemma A.19. *Suppose $\mathcal{P}$ is compatible with $\sqsupset^{\pi}_{\mathsf{pop}*}$, i.e. suppose $\mathcal{P} \subseteq \sqsupset^{\pi}_{\mathsf{pop}*}$ holds. Then for k depending only on $\mathcal{R}$,*

$$s \xrightarrow{\mathsf{v}}{}^{\varepsilon}_{\mathcal{P}} t \Longrightarrow \mathsf{N'_s}(\pi(C[u_1, \dots, s, \dots, u_n])) \blacktriangleright_k \mathsf{N'_s}(\pi(C[u_1, \dots, t, \dots, u_n]))$$

for every safe argument filtering π and context $C \in \mathcal{T}(\mathcal{C}_{\text{COM}} \cup \{\square\}, \mathcal{V})$ build from compound symbols.

Proof. Cf. Lemma 6.12. □

Theorem A.20. *If $\mathcal{P} \subseteq \sqsupset^{\pi}_{\mathsf{pop}*}$ and $\mathcal{U}(\mathcal{P}) \subseteq \sqsupseteq^{\pi}_{\mathsf{pop}*}$ then there exists a polynomial p with*

$$\mathrm{dl}(t^{\sharp}, \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}) \leqslant p(|t|)$$

for any constructor-based term $t \in \mathcal{T}_{\mathsf{b}}$. The polynomial p depends only on $\mathcal{R}$.

Proof. Let $t \in \mathcal{T}_{\mathsf{b}}$, and assume a maximal derivation

$$t^{\sharp} = t^{\sharp}_0 \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t^{\sharp}_1 \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} \cdots \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t^{\sharp}_{\ell} \ .$$

Consider a relative step $t^{\sharp}_i \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})} t^{\sharp}_{i+1}$ for $i \in \{1, \dots, \ell - 1\}$. We abbreviate $U'(\mathcal{P} \cup \mathcal{U}(\mathcal{P}))$ by $\mathcal{V}$. Define $\mathcal{U} = \mathcal{U}(\mathcal{P}) \cup \mathcal{V}$, and observe $t'_i \xrightarrow{\mathsf{v}'}{}^{\varepsilon}_{\mathcal{P}/\mathcal{U}} t'_{i+1}$ for $t'_j = t^{\sharp}_j{\downarrow_{\mathcal{V}}}$ according to Lemma A.9. Hence there exists terms u and v such that

$$t'_i \xrightarrow{\mathcal{P} \cup \mathcal{U}}{}^{*}_{\mathcal{U}} u \xrightarrow{\mathcal{P} \cup \mathcal{U}}_{\mathcal{P}} v \xrightarrow{\mathcal{P} \cup \mathcal{U}}{}^{*}_{\mathcal{U}} t'_{i+1} \ .$$

Next observe that $\mathsf{N'_s}(\pi(t'_i)) \mathrel{\underset{\sim}{\blacktriangleright}}{}^{*}_{k} \mathsf{N'_s}(\pi(u))$ and likewise $\mathsf{N'_s}(\pi(v)) \mathrel{\underset{\sim}{\blacktriangleright}}{}^{*}_{k} \mathsf{N'_s}(\pi(t'_{i+1}))$, which follows from Lemma A.18 and the shape of $\mathcal{V}$ (cf. Lemma 5.21). From the shape of $\mathcal{P}$ and the assumption $t \in \mathcal{T}_{\mathsf{b}}$ we conclude that for some context $C \in \mathcal{T}(\mathcal{C}_{\text{COM}} \cup \{\square\}, \mathcal{V})$, we have $u = C[u^{\sharp}_1, \dots, u^{\sharp}_i, \dots, u^{\sharp}_p]$ and $v = C[u^{\sharp}_1, \dots, v^{\sharp}_i, \dots, u^{\sharp}_p]$ with $u^{\sharp}_i \xrightarrow{\mathsf{v}}{}^{\varepsilon}_{\mathcal{P}} v^{\sharp}_i$. From Lemma A.19 we conclude $\mathsf{N'_s}(\pi(u)) \blacktriangleright_k \mathsf{N'_s}(\pi(v))$. Summing up, we derive

$$\mathsf{N'_s}(\pi(t'_i)) \blacktriangleright^{+}_{k} \mathsf{N'_s}(\pi(t'_{i+1}))$$

due to compatibility of $\blacktriangleright_k$ with $\sim$.

From the above, we derive $\mathrm{dl}(t^{\sharp}, \xrightarrow{\mathsf{i}}_{\mathcal{P}/\mathcal{U}(\mathcal{P})}) \leqslant \mathsf{G}_k(\mathsf{N'_s}(\pi(t^{\sharp}{\downarrow_{\mathcal{V}}})))$. It can be verified that the latter is bounded polynomially in the size of t, and we conclude the theorem. □

A.3 POP* as Direct Method

In the preceding section we have seen that $\mathcal{R}$ can be partitioned into $\mathcal{R}'$ and $\mathcal{R}^{\rhd}$. Suppose $\mathcal{R} \subseteq \sqsupset_{\mathsf{pop}*}$ (here $\sqsupset_{\mathsf{pop}*}$ refers to $\sqsupset^{\pi}_{\mathsf{pop}*}$ where $\pi = \mathsf{id}$). We have shown that $s \xrightarrow{\mathsf{i}}_{\mathcal{R}'} t$ implies $\mathsf{N}'(s) \blacktriangleright_k^+ \mathsf{N}'(t)$ (this is just Lemma 5.33), whereas for $s \xrightarrow{\mathsf{i}}_{\mathcal{R}^{\rhd}} t$ we can only show $\mathsf{N}'(s) \underset{\sim}{\blacktriangleright}_k^+ \mathsf{N}'(t)$ (cf. Lemma A.13). However, the number of $\mathcal{R}^{\rhd}$-steps in an innermost $\mathcal{R}$ derivation is tightly controlled by the number of $\mathcal{R}'$ steps. Consider the following Lemma.

Lemma A.21. *Suppose* $\mathcal{R} \subseteq \sqsupset_{\mathsf{pop}*}$.

$$\mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}}) \leqslant (\mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}'/\mathcal{R}^{\rhd}}) + 1)^2 * (\triangle_{\mathcal{R}} + 1) + |t|$$

where $\triangle_{\mathcal{R}} = \max\{|r| \mid \ell \to r \in \mathcal{R}\}$.

Proof. Let $w(t) = \mathrm{dl}(t, \xrightarrow{\mathcal{R}}_{\mathcal{R}^{\rhd}})$, and consider the following claim.

Claim. $s \xrightarrow{\mathcal{R}}_{\mathcal{R}'} t$ implies $w(t) \leqslant w(s) + \mathrm{depth}(s) + \triangle_{\mathcal{R}}$.

Proof of claim. Suppose $s = C[\ell\sigma]$, $t = C[r\sigma]$ and $\ell \to r \in \mathcal{R}'$ such that $\ell_i\sigma \in \mathsf{NF}(\mathcal{R})$ for $\ell = f(\ell_1, \dots, \ell_m)$. We continue by induction on C. For the base case, suppose $C = \Box$. We continue by induction on r and show the stronger result $w(r\sigma) \leqslant |r|$. The base case $r \in \mathcal{V}$ is trivial since then $r\sigma \in \mathsf{NF}(\mathcal{R})$ and $\mathcal{R}^{\rhd} \subseteq \mathcal{R}$. So assume $r\sigma = g(r_1\sigma, \dots, r_m\sigma)$. We have $w(r\sigma) \leqslant \Sigma_{j=1}^m w(r_i\sigma) + 1 \overset{\mathsf{IH}}{\leqslant} \Sigma_{j=1}^m |r_i| + 1 = |r|$ and conclude the base case of the claim.

For the inductive step, suppose $C[\Box] = g(s_1, \dots, C'[\Box], \dots, s_m)$ and thus $t = g(s_1, \dots, C'[r\sigma], \dots, s_m)$. We have $w(s) = \sum_{j \neq i} w(s_i) + w(C'[\ell\sigma])$ (as we cannot apply a rule from $\mathcal{R}^{\rhd}$ at the root), likewise $w(t) \leqslant \sum_{j \neq i} w(s_i) + w(C'[r\sigma]) + 1$. By induction hypothesis $w(C'[r\sigma]) \leqslant w(C'[\ell\sigma]) + \mathrm{depth}(C'[\ell\sigma]) + \triangle_{\mathcal{R}}$. As $\mathrm{depth}(s) > \mathrm{depth}(C'[\ell\sigma])$ we conclude the claim. $\Box$

Now consider a maximal sequence of $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ steps starting from a term t, without loss of generality of the form

$$t = t_0 \xrightarrow{\mathcal{R}}{}^{l_0}_{\mathcal{R}^{\rhd}} s_0 \xrightarrow{\mathcal{R}}_{\mathcal{R}'} t_1 \xrightarrow{\mathcal{R}}{}^{l_1}_{\mathcal{R}^{\rhd}} \dots s_{m-1} \xrightarrow{\mathcal{R}}_{\mathcal{R}'} t_m \xrightarrow{\mathcal{R}}{}^{l_m}_{\mathcal{R}^{\rhd}} s_m \ .$$

Clearly, $m \leqslant \mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}'/\mathcal{R}^{\rhd}})$, moreover notice that $l_i \leqslant w(t_i) - w(s_i)$ for all i. Observe that $s_i \xrightarrow{\mathcal{R}}_{\mathcal{R}'} t_{i+1}$ implies $\mathrm{depth}(t_{i+1}) \leqslant \mathrm{depth}(s_i) + \triangle_{\mathcal{R}}$, and moreover $t_{i+1} \xrightarrow{\mathcal{R}}{}^{l_{i+1}}_{\mathcal{R}^{\rhd}} s_{i+1}$ implies $\mathrm{depth}(s_{i+1}) \leqslant \mathrm{depth}(t_{i+1})$, we see $\mathrm{depth}(s_{i+1}) \leqslant \mathrm{depth}(s_i) + \triangle_{\mathcal{R}}$. Hence $\sum_{i=0}^{m-1} \mathrm{depth}(s_i) \leqslant m^2 * \triangle_{\mathcal{R}}$, a fact we employ below.

Summing up all observations, we derive

$$\begin{aligned}
\sum_{i=0}^{m} l_i &\leqslant \sum_{i=0}^{m-1}(w(t_{i+1}) - w(s_{i+1})) + l_0 \\
&\leqslant \sum_{i=0}^{m-1}(w(s_i) + \mathrm{depth}(s_i) + \triangle_{\mathcal{R}} - w(s_{i+1})) + l_0 \\
&= m * \triangle_{\mathcal{R}} + \sum_{i=0}^{m-1}(w(s_i) - w(s_{i+1})) + \sum_{i=0}^{m-1}\mathrm{depth}(s_i) + l_0 \\
&= m * \triangle_{\mathcal{R}} + w(s_0) + m^2 * \triangle_{\mathcal{R}} + l_0 \\
&\leqslant (m+1)^2 * \triangle_{\mathcal{R}} + w(s_0) + l_0 \\
&\leqslant (m+1)^2 * \triangle_{\mathcal{R}} + w(t) \\
&\leqslant (m+1)^2 * \triangle_{\mathcal{R}} + |t|
\end{aligned}$$

As m is bounded by $\mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}'/\mathcal{R}^{\triangleright}})$ and $\mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}}) = \sum_{i=0}^{m} l_i + m$ we derive the desired result. □

Theorem A.22. *If $\mathcal{R} \subseteq\ \sqsupset_{\mathsf{pop*}}$ then there exists a polynomial p with*

$$\mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}}) \leqslant p(|t|)$$

for any constructor-based term $t \in \mathcal{T}_{\mathsf{b}}$. The polynomial p depends only on $\mathcal{R}$.

Proof. By Lemma A.21 $\mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}}) \leqslant |t| + \mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}'/\mathcal{R}^{\triangleright}}) * \triangle_{\mathcal{R}}$ where $\triangle_{\mathcal{R}}$ depends only on $\mathcal{R}$. For $s \xrightarrow{\mathsf{i}}_{\mathcal{R}'/\mathcal{R}^{\triangleright}} t$ have $\mathsf{N}'(s) \blacktriangleright_k^{+} \mathsf{N}'(t)$ by the observation from the beginning of the section , and thus $\mathrm{dl}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}}) \leqslant |t| + (\mathsf{G}_k(t) + 1)^2 * c$ for some constant $c \in \mathbb{N}$. The latter follows from Lemma A.21. As for $t \in \mathcal{T}_{\mathsf{b}}$, $\mathsf{G}_k(t)$ is bounded by a polynomial in the size of t, we conclude the theorem. □