Transforming Derivational Complexity of Term Rewriting to Runtime Complexity

Carsten Fuhs

Department of Computer Science and Information Systems,
Birkbeck, University of London, United Kingdom

Abstract. Derivational complexity of term rewriting considers the length of the longest rewrite sequence for arbitrary start terms, whereas runtime complexity restricts start terms to basic terms. Recently, there has been notable progress in automatic inference of upper and lower bounds for runtime complexity. We propose a novel transformation that allows an off-the-shelf tool for inference of upper or lower bounds for runtime complexity to be used to determine upper or lower bounds for derivational complexity as well. Our approach is applicable to derivational complexity problems for innermost rewriting and for full rewriting. We have implemented the transformation in the tool AProVE and conducted an extensive experimental evaluation. Our results indicate that bounds for derivational complexity can now be inferred for rewrite systems that have been out of reach for automated analysis thus far.

1 Introduction

Term rewrite systems (TRSs) are a classic computational model both for equational reasoning and for evaluation of programs with user-defined data structures and recursion [5]. A widely studied question for TRSs is that of their *complexity*, i.e., the length of their longest derivation (i.e., rewrite sequence) as a function of the size of the start term of the derivation. From a program analysis perspective, this corresponds to the worst-case time complexity of the TRS.

In the literature, commonly two distinct notions are considered for the set of start terms. On the one hand, the *derivational complexity* [21] of a term rewrite system considers arbitrary terms as start terms that need to be regarded, including terms with several (possibly nested) function calls. This notion is inspired by the notion of termination of a rewrite system, which also considers whether all rewrite sequences from arbitrary start terms terminate. Derivational complexity is a suitable measure for the number of rewrite steps needed for deciding the word problem in first-order equational reasoning with the help of a terminating and confluent term rewrite system to rewrite both sides of the conjectured equality to normal form.

On the other hand, the notion of *runtime complexity* [18] of a term rewrite system restricts the set of start terms that are regarded to what is known as *basic terms*: intuitively, these are terms where a single function call is performed on constructor terms (i.e., data objects) as arguments. The motivation for this

restriction comes from program analysis, where one is usually interested in the running time of a function when it is invoked on data objects.

These notions have been particularly studied for term rewriting with arbitrary rewrite strategies ("full rewriting") and for term rewriting restricted to innermost rewrite strategies ("innermost rewriting"). The latter notion is closely related to call-by-value evaluation in programming languages and λ-calculi.

Both for the derivational complexity and the runtime complexity of rewriting, and both for innermost and full rewriting, fully automatic push-button tools have been devised to determine asymptotic upper and lower bounds on the derivational complexity and the runtime complexity of term rewriting. Examples include the tools AProVE [14], CaT [23], Matchbox [34], and TcT [4].

However, as far as the author of this paper is aware, the two strands of research on derivational and on runtime complexity have essentially stayed separate thus far. While an upper bound on derivational complexity also implies an upper bound on runtime complexity and a lower bound on runtime complexity also implies a lower bound on derivational complexity, these implied bounds are seldom tight. A translation that would allow for applying tools for analysis of runtime complexity to analysis of derivational complexity (or vice versa) to infer potentially tight bounds is still missing. This paper aims to close this gap. We propose a transformation between rewrite systems such that the runtime complexity of the transformed rewrite system is the same as the derivational complexity of the original rewrite system. This transformation is applicable both for innermost rewriting and for full rewriting.

This paper is organized as follows. In Sec. 2, we give preliminaries on term rewriting and on notions of complexity. Sec. 3 proposes our novel transformation (Def. 5) and proves its correctness (Thm. 12 and Thm. 14). In Sec. 4, we discuss related work for complexity analysis of rewriting and for transformational approaches to analysis of rewrite systems. Sec. 5 provides a detailed experimental evaluation of our contributions on a large standard benchmark set. Sec. 6 concludes.

2 Preliminaries

In the following, we assume basic knowledge of term rewriting [5]. We recapitulate (relative) term rewriting as well as the notions of derivational complexity and runtime complexity.

Definition 1 (Signature, term, term rewriting, defined symbol, constructor symbol, basic term). *We write $\mathcal{T}(\Sigma, \mathcal{V})$ for the set of* terms *over a finite signature Σ and the set of variables $\mathcal{V}$. For a term t, $\mathcal{V}(t)$ denotes the set of variables occurring in t, and if t has the form $f(t_1, \ldots, t_n)$, we write* $\mathrm{root}(t) = f$.

A term rewrite system (TRS) *$\mathcal{R}$ is a set of rules $\{\ell_1 \to r_1, \ldots, \ell_n \to r_n\}$ with $\ell_i, r_i \in \mathcal{T}(\Sigma, \mathcal{V})$, $\ell_i \notin \mathcal{V}$, and $\mathcal{V}(r_i) \subseteq \mathcal{V}(\ell_i)$ for all $1 \leq i \leq n$. Its* rewrite relation *is given by $s \to_{\mathcal{R}} t$ iff there is a rule $\ell \to r \in \mathcal{R}$, a position $\pi \in \mathcal{P}os(s)$, and a substitution σ such that $s = s[\ell\sigma]_\pi$ and $t = s[r\sigma]_\pi$. Here, the term $\ell\sigma$ is called the* redex *of the rewrite step.*

For two TRSs $\mathcal{R}$ and $\mathcal{S}$, $\mathcal{R}/\mathcal{S}$ is a relative TRS*, and its* rewrite relation $\to_{\mathcal{R}/\mathcal{S}}$ *is $\to_{\mathcal{S}}^* \circ \to_{\mathcal{R}} \circ \to_{\mathcal{S}}^*$, i.e., rewriting with $\mathcal{S}$ is allowed before and after each $\mathcal{R}$-step. We define the* innermost *rewrite relation by $s \xrightarrow{\mathrm{i}}_{\mathcal{R}/\mathcal{S}} t$ iff $s \to_{\mathcal{S}}^* s' \to_{\mathcal{R}} s'' \to_{\mathcal{S}}^* t$ for some terms s', s'' such that the proper subterms of the redexes of each step with $\to_{\mathcal{S}}$ or $\to_{\mathcal{R}}$ are in normal form w.r.t. $\mathcal{R} \cup \mathcal{S}$. We may write $\to_{\mathcal{R}}$ instead of $\to_{\mathcal{R}/\emptyset}$ and $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ instead of $\xrightarrow{\mathrm{i}}_{\mathcal{R}/\emptyset}$.*

For a relative TRS $\mathcal{R}/\mathcal{S}$, $\Sigma_d^{\mathcal{R}\cup\mathcal{S}} = \{\mathrm{root}(\ell) \mid \ell \to r \in \mathcal{R} \cup \mathcal{S}\}$ and $\Sigma_c^{\mathcal{R}\cup\mathcal{S}} = \{f \mid f \in \Sigma$ occurs in some rule $\ell \to r \in \mathcal{R} \cup \mathcal{S}\} \setminus \Sigma_d^{\mathcal{R}\cup\mathcal{S}}$ are the defined *(and* constructor*, respectively) symbols of $\mathcal{R}/\mathcal{S}$. We write $\Sigma^{\mathcal{R}\cup\mathcal{S}} = \Sigma_d^{\mathcal{R}\cup\mathcal{S}} \uplus \Sigma_c^{\mathcal{R}\cup\mathcal{S}}$. A term $f(t_1, \ldots, t_k)$ is* basic (for a given relative TRS $\mathcal{R}/\mathcal{S}$) *iff $f \in \Sigma_d^{\mathcal{R}\cup\mathcal{S}}$ and $t_1, \ldots, t_k \in \mathcal{T}(\Sigma_c^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$. We write $\mathcal{T}_{\mathrm{basic}}^{\mathcal{R}/\mathcal{S}}$ for the set of basic terms for a relative TRS $\mathcal{R}/\mathcal{S}$.*

In the following, ω is the smallest infinite ordinal, i.e., $\omega > n$ holds for all $n \in \mathbb{N}$, and for any $P \subseteq \mathbb{N} \cup \{\omega\}$, $\sup\ P$ is the least upper bound of P, where $\sup\ \emptyset = 0$.

Definition 2 (Size, derivation height, derivational complexity dc**, runtime complexity** rc **[18, 21, 37]).** *The* size $|t|$ *of a term t is defined as $|x| = 1$ if $x \in \mathcal{V}$ and $|f(t_1, \ldots, t_k)| = 1 + \sum_{i=1}^{k} |t_i|$, otherwise.*

The derivation height *of a term t w.r.t. a relation $\to$ is the length of the longest sequence of $\to$-steps starting with t, i.e., $\mathrm{dh}(t, \to) = \sup\{e \mid \exists\, t' \in \mathcal{T}(\Sigma, \mathcal{V}).\ t \to^e t'\}$ where $\to^e$ denotes the e^{th} iterate of $\to$. If t starts an infinite $\to$-sequence, we write $\mathrm{dh}(t, \to) = \omega$.*

To define the intended complexity notions, we first introduce a generic complexity function compl *parameterized by a natural number n, a relation $\to$, and a set of start terms $\mathcal{T}$: $\mathrm{compl}(n, \to, \mathcal{T}) = \sup\{\mathrm{dh}(t, \to) \mid t \in \mathcal{T}, |t| \leq n\}$.*

The derivational complexity *function $\mathrm{dc}_{\mathcal{R}/\mathcal{S}}$ maps any $n \in \mathbb{N}$ to the length of the longest sequence of $\to_{\mathcal{R}/\mathcal{S}}$-steps starting with a term whose size is at most n, i.e., $\mathrm{dc}_{\mathcal{R}/\mathcal{S}}(n) = \mathrm{compl}(n, \to_{\mathcal{R}/\mathcal{S}}, \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V}))$. The* innermost derivational complexity *function $\mathrm{idc}_{\mathcal{R}/\mathcal{S}}$ is defined analogously for innermost rewriting: $\mathrm{idc}_{\mathcal{R}/\mathcal{S}}(n) = \mathrm{compl}(n, \xrightarrow{\mathrm{i}}_{\mathcal{R}/\mathcal{S}}, \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V}))$.*

The runtime complexity *function $\mathrm{rc}_{\mathcal{R}/\mathcal{S}}$ maps any $n \in \mathbb{N}$ to the length of the longest sequence of $\to_{\mathcal{R}/\mathcal{S}}$-steps starting with a* basic *term whose size is at most n, i.e., $\mathrm{rc}_{\mathcal{R}/\mathcal{S}}(n) = \mathrm{compl}(n, \to_{\mathcal{R}/\mathcal{S}}, \mathcal{T}_{\mathrm{basic}}^{\mathcal{R}/\mathcal{S}})$. The* innermost runtime complexity *function $\mathrm{irc}_{\mathcal{R}/\mathcal{S}}$ is defined analogously: $\mathrm{irc}_{\mathcal{R}/\mathcal{S}}(n) = \mathrm{compl}(n, \xrightarrow{\mathrm{i}}_{\mathcal{R}/\mathcal{S}}, \mathcal{T}_{\mathrm{basic}}^{\mathcal{R}/\mathcal{S}})$.*

Our transformation will preserve and reflect derivation height precisely. However, many analysis techniques for derivational complexity and runtime complexity of rewriting consider *asymptotic* behavior. The following definition is standard.

Definition 3 (Asymptotic notation, $\mathcal{O}$, Ω, Θ). *Let $f, g : \mathbb{N} \to \mathbb{N} \cup \{\omega\}$. Then $f(n) \in \mathcal{O}(g(n))$ iff there are constants $M, N \in \mathbb{N}$ such that $f(n) \leq M \cdot g(n)$ for all $n \geq N$. Moreover, $f(n) \in \Omega(g(n))$ iff $g(n) \in \mathcal{O}(f(n))$, and $f(n) \in \Theta(g(n))$ holds iff both $f(n) \in \mathcal{O}(g(n))$ and $f(n) \in \Omega(g(n))$ hold.*

Example 4 (plus*).* Consider the relative TRS $\mathcal{R}/\mathcal{S}$ with the following rules in $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{0}, x) \to x$$
$$\mathsf{plus}(\mathsf{s}(x), y) \to \mathsf{s}(\mathsf{plus}(x, y))$$

and with $\mathcal{S} = \emptyset$. Here $\mathsf{0}$ and s are constructor symbols, and plus is a defined symbol. We have $\mathrm{rc}_{\mathcal{R}/\mathcal{S}}(n) \in \Theta(n)$ (so both $\mathrm{rc}_{\mathcal{R}/\mathcal{S}}(n) \in \mathcal{O}(n)$ and $\mathrm{rc}_{\mathcal{R}/\mathcal{S}}(n) \in \Omega(n)$ hold) and $\mathrm{irc}_{\mathcal{R}/\mathcal{S}}(n) \in \Theta(n)$. Moreover, we have $\mathrm{dc}_{\mathcal{R}/\mathcal{S}}(n) \in \Theta(n^2)$ and $\mathrm{idc}_{\mathcal{R}/\mathcal{S}}(n) \in \Theta(n^2)$.

3 From Derivational Complexity to Runtime Complexity

In this section we present the main contribution of this paper, an instrumentation of a relative TRS $\mathcal{R}/\mathcal{S}$ to a relative TRS $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$ with the same (innermost or full) runtime complexity. Moreover, we provide a proof for its correctness. The idea is to encode the set of arbitrary start terms that is considered for derivational complexity into a set of corresponding basic terms of the same size that can be analyzed for runtime complexity. This is accomplished by adding further constructor symbols cons_f that represent the defined symbols f from $\mathcal{R}/\mathcal{S}$. We also add an "instrumentation" in the form of relative rewrite rules $\mathcal{G}$ that generate the original start term for $\mathcal{R}/\mathcal{S}$ from its encoding as a basic term for $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$, but do not lead to additional derivation height. The root symbol for these basic terms will be called enc_f for f a defined or a constructor symbol for $\mathcal{R}/\mathcal{S}$ (note that the root symbol of a start term for derivational complexity with maximum derivation height is not necessarily a defined symbol, e.g., consider the rewrite rule $\mathsf{a} \to \mathsf{c}(\mathsf{b}, \mathsf{b})$). We will also introduce an auxiliary function symbol argenc for recursive application of the additional rewrite rules.

For example, a start term $\mathsf{plus}(\mathsf{plus}(\mathsf{s}(\mathsf{0}), \mathsf{0}), x)$ for derivational complexity will be represented by a basic term $\mathsf{enc}_{\mathsf{plus}}(\mathsf{cons}_{\mathsf{plus}}(\mathsf{s}(\mathsf{0}), \mathsf{0}), x)$. Here $\mathsf{enc}_{\mathsf{plus}}$ will be a defined symbol and $\mathsf{cons}_{\mathsf{plus}}$ a constructor symbol. Rewriting using $\xrightarrow{\mathsf{i}}_{\mathcal{G}}$ then restores (an instance of) the original start term.

Definition 5 (Generator rules $\mathcal{G}$, runtime instrumentation). *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS. We define the* generator rules $\mathcal{G}$ *of $\mathcal{R}/\mathcal{S}$ as the set of rules*

$$\begin{aligned}\mathcal{G} = {} & \{\mathsf{enc}_f(x_1, \ldots, x_n) \to f(\mathsf{argenc}(x_1), \ldots, \mathsf{argenc}(x_n)) \mid f \in \Sigma^{\mathcal{R}\cup\mathcal{S}}\} \\ & \cup \{\mathsf{argenc}(\mathsf{cons}_f(x_1, \ldots, x_n)) \to f(\mathsf{argenc}(x_1), \ldots, \mathsf{argenc}(x_n)) \mid f \in \Sigma_d^{\mathcal{R}\cup\mathcal{S}}\} \\ & \cup \{\mathsf{argenc}(f(x_1, \ldots, x_n)) \to f(\mathsf{argenc}(x_1), \ldots, \mathsf{argenc}(x_n)) \mid f \in \Sigma_c^{\mathcal{R}\cup\mathcal{S}}\}\end{aligned}$$

where $x_1, \ldots, x_n$ are variables and where all function symbols argenc*,* cons_f*, and* enc_f *are fresh (i.e., they do not occur in $\mathcal{R} \cup \mathcal{S}$). We call the relative TRS $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$ the* runtime instrumentation *of $\mathcal{R}/\mathcal{S}$. Moreover, we call terms over the signature $\{\mathsf{enc}_f \mid f \in \Sigma^{\mathcal{R}\cup\mathcal{S}}\} \cup \{\mathsf{cons}_f \mid f \in \Sigma_d^{\mathcal{R}\cup\mathcal{S}}\} \cup \Sigma_c^{\mathcal{R}\cup\mathcal{S}}$* generator terms *for $\mathcal{R}/\mathcal{S}$ (they are the intended start terms for $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$).*

Example 6 (Example 4 continued). Continuing Example 4, we obtain the following generator rules $\mathcal{G}$:

$$\begin{aligned}
\mathsf{enc}_{\mathsf{plus}}(x, y) &\to \mathsf{plus}(\mathsf{argenc}(x), \mathsf{argenc}(y)) \\
\mathsf{enc}_{\mathsf{0}} &\to \mathsf{0} \\
\mathsf{enc}_{\mathsf{s}}(x) &\to \mathsf{s}(\mathsf{argenc}(x)) \\
\mathsf{argenc}(\mathsf{cons}_{\mathsf{plus}}(x, y)) &\to \mathsf{plus}(\mathsf{argenc}(x), \mathsf{argenc}(y)) \\
\mathsf{argenc}(\mathsf{0}) &\to \mathsf{0} \\
\mathsf{argenc}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{argenc}(x))
\end{aligned}$$

To reason about our transformation, we introduce several helper functions to encode and decode arbitrary terms for $\mathcal{R}/\mathcal{S}$ as basic terms for $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$.

Definition 7 (Constructor variant, basic variant, decoded variant). *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS and let $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$ be its runtime instrumentation.*

For a term $t \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$, we define its constructor variant $\mathrm{cv}(t)$ *inductively as follows:*

- $\mathrm{cv}(x) = x$ *for* $x \in \mathcal{V}$
- $\mathrm{cv}(f(t_1, \ldots, t_n)) = f(\mathrm{cv}(t_1), \ldots, \mathrm{cv}(t_n))$ *for* $f \in \Sigma_c^{\mathcal{R}\cup\mathcal{S}}$
- $\mathrm{cv}(f(t_1, \ldots, t_n)) = \mathsf{cons}_f(\mathrm{cv}(t_1), \ldots, \mathrm{cv}(t_n))$ *for* $f \in \Sigma_d^{\mathcal{R}\cup\mathcal{S}}$

For a term $t = f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$, we define its basic variant $\mathrm{bv}(f(t_1, \ldots, t_n)) = \mathsf{enc}_f(\mathrm{cv}(t_1), \ldots, \mathrm{cv}(t_n))$.

Finally, for a term $t \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}}, \mathcal{V})$, we define its decoded variant $\mathrm{dv}(t) \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$ *as follows:*

- $\mathrm{dv}(x) = x$ *for* $x \in \mathcal{V}$
- $\mathrm{dv}(f(t_1, \ldots, t_n)) = g(\mathrm{dv}(t_1), \ldots, \mathrm{dv}(t_n))$ *for* $f \in \{g, \mathsf{cons}_g, \mathsf{enc}_g\}$ *with* $g \in \Sigma_d^{\mathcal{R}\cup\mathcal{S}}$
- $\mathrm{dv}(f(t_1, \ldots, t_n)) = f(\mathrm{dv}(t_1), \ldots, \mathrm{dv}(t_n))$ *for* $f \in \Sigma_c^{\mathcal{R}\cup\mathcal{S}}$

The following lemmas address properties of our helper functions that we will use in the proofs of our theorems.

Lemma 8 (Basic variants of function applications are basic terms). *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS, let t be a term from $\mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$.*

(a) We have $\mathrm{cv}(t) \in \mathcal{T}(\Sigma_c^{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}}, \mathcal{V})$.
(b) The term $\mathrm{bv}(t)$ *is a basic term for the runtime instrumentation* $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$.
(c) $|\mathrm{bv}(t)| = |t|$.

Proof. Claims (a) and (b) follow directly from the definitions of cv, of bv, and of generator rules, and claim (c) follows by induction over the definitions of bv and cv. □

In the following lemmas and proofs, we will need a particular substitution σ_t that maps each variable x of a term t to $\mathsf{argenc}(x)$, so we introduce corresponding notation.

Definition 9 (σ_t). *For a term t, we define the substitution σ_t by $\sigma_t(x) = \mathsf{argenc}(x)$ for $x \in \mathcal{V}(t)$ and $\sigma_t(x) = x$ otherwise.*

Lemma 10 (argenc reduces constructor variants innermost to instances of their originals). *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS and $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$ its runtime instrumentation. Then for all $t \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$, $\mathsf{argenc}(\mathrm{cv}(t)) \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} t\sigma_t$ is an innermost rewrite sequence that moreover uses only rules from $\mathcal{G}$.*

Proof. By induction over the structure of t. Let $t \in \mathcal{V}$. Then $\mathrm{cv}(t) = t$, and $\mathsf{argenc}(\mathrm{cv}(t)) \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} t\sigma_t$ in zero steps.

Now let $t = f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$. By induction hypothesis, we have $\mathsf{argenc}(\mathrm{cv}(t_i)) \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} t_i\sigma_{t_i}$.

If $f \in \Sigma_c^{\mathcal{R}\cup\mathcal{S}}$, we have $\mathrm{cv}(f(t_1, \ldots, t_n)) = f(\mathrm{cv}(t_1), \ldots, \mathrm{cv}(t_n))$. By applying the induction hypothesis, we get the desired innermost rewrite sequence:

$$\begin{aligned}
& \mathsf{argenc}(\mathrm{cv}(t)) \\
= \ & \mathsf{argenc}(f(\mathrm{cv}(t_1), \ldots, \mathrm{cv}(t_n))) \\
\xrightarrow{\mathrm{i}}_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} \ & f(\mathsf{argenc}(\mathrm{cv}(t_1)), \ldots, \mathsf{argenc}(\mathrm{cv}(t_n))) \\
\xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} \ & f(t_1\sigma_{t_1}, \ldots, t_n\sigma_{t_n}) \\
= \ & t\sigma_t
\end{aligned}$$

If $f \in \Sigma_d^{\mathcal{R}\cup\mathcal{S}}$, we have $\mathrm{cv}(f(t_1, \ldots, t_n)) = \mathsf{cons}_f(\mathrm{cv}(t_1), \ldots, \mathrm{cv}(t_n))$. By applying the induction hypothesis, we get the desired innermost rewrite sequence:

$$\begin{aligned}
& \mathsf{argenc}(\mathrm{cv}(t)) \\
= \ & \mathsf{argenc}(\mathsf{cons}_f(\mathrm{cv}(t_1), \ldots, \mathrm{cv}(t_n))) \\
\xrightarrow{\mathrm{i}}_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} \ & f(\mathsf{argenc}(\mathrm{cv}(t_1)), \ldots, \mathsf{argenc}(\mathrm{cv}(t_n))) \\
\xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} \ & f(t_1\sigma_{t_1}, \ldots, t_n\sigma_{t_n}) \\
= \ & t\sigma_t
\end{aligned}$$

□

Lemma 11 (Basic variants reduce innermost to instances of their originals). *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS and $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$ its runtime instrumentation. Then for all $t \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$ of the form $f(t_1, \ldots, t_n)$, $\mathrm{bv}(t) \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} t\sigma_t$ is an innermost rewrite sequence that moreover uses only rules from $\mathcal{G}$.*

Proof. Let $t \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$ of the form $f(t_1, \ldots, t_n)$. Then we have $\mathrm{bv}(t) = \mathsf{enc}_f(\mathrm{cv}(t_1), \ldots, \mathrm{cv}(t_n))$. The only possible rewrite step uses the rewrite rule $\mathsf{enc}_f(x_1, \ldots, x_n) \to f(\mathsf{argenc}(x_1), \ldots, \mathsf{argenc}(x_n))$ in $\mathcal{G}$ for $\mathrm{bv}(t) \xrightarrow{\mathrm{i}}_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} f(\mathsf{argenc}(\mathrm{cv}(t_1)), \ldots, \mathsf{argenc}(\mathrm{cv}(t_n)))$. By Lemma 10, we have:

$f(\mathsf{argenc}(\mathrm{cv}(t_1)), \ldots, \mathsf{argenc}(\mathrm{cv}(t_n))) \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} f(t_1\sigma_{t_1}, \ldots, t_n\sigma_{t_n}) = t\sigma_t$ □

Now we are ready to prove the first theorem:

Theorem 12 (Derivational complexity via runtime complexity). *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS and let $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$ be its runtime instrumentation. Then for all $n \in \mathbb{N}$, we have $\mathrm{dc}_{\mathcal{R}/\mathcal{S}}(n) = \mathrm{rc}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}(n)$.*

Proof. We show the two directions of the theorem separately.

(1) $\mathrm{dc}_{\mathcal{R}/\mathcal{S}}(n) \leq \mathrm{rc}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}(n)$.

For $n = 0$, there are no terms of size $\leq n$. Thus, let $n > 0$ and let $t \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$ be an arbitrary term with $|t| \leq n$ starting a $\to_{\mathcal{R}/\mathcal{S}}$-rewrite sequence

$$t = t_0 \ \to_{\mathcal{R}/\mathcal{S}} \ t_1 \ \to_{\mathcal{R}/\mathcal{S}} \ t_2 \ \to_{\mathcal{R}/\mathcal{S}} \ \cdots$$

of maximal length for all terms of size at most n, i.e., (i) $\mathrm{dh}(t, \to_{\mathcal{R}/\mathcal{S}}) = \mathrm{dc}_{\mathcal{R}/\mathcal{S}}(n)$.

By Lemma 11, we have $\mathrm{bv}(t) \to_{\mathcal{G}}^* t\sigma_t$. Since $\to_{\mathcal{R}/\mathcal{S}} \subseteq \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$ and since rewriting is closed under substitutions, we have

$$\mathrm{bv}(t) \ \to_{\mathcal{G}}^* \ t\sigma_t \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} \ t_1\sigma_t \ \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} \ t_2\sigma_t \ \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} \ \cdots$$

which yields

$$\mathrm{bv}(t) \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} \ t_1\sigma_t \ \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} \ t_2\sigma_t \ \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} \ \cdots$$

and thus (ii) $\mathrm{dh}(t, \to_{\mathcal{R}/\mathcal{S}}) \leq \mathrm{dh}(\mathrm{bv}(t), \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})})$.

By Lemma 8, $\mathrm{bv}(t)$ is a basic term for $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$ with $|\mathrm{bv}(t)| = |t| \leq n$. Thus, $\mathrm{dh}(\mathrm{bv}(t), \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}) \leq \mathrm{rc}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}(n)$. Using equality (i) and inequality (ii), we can conclude that the claim indeed holds.

(2) $\mathrm{dc}_{\mathcal{R}/\mathcal{S}}(n) \geq \mathrm{rc}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}(n)$.

For $n = 0$, there are no terms of size $\leq n$. Thus, let $n > 0$ and let $t \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}}, \mathcal{V})$ be an arbitrary basic term for $\mathcal{R}/(\mathcal{S} \uplus \mathcal{G})$ with $|t| \leq n$ starting a $\to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$-rewrite sequence

$$t = t_0 \ \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} \ t_1 \ \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} \ t_2 \ \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} \ \cdots$$

of maximal length for all terms of size at most n, i.e., $\mathrm{dh}(t, \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}) = \mathrm{rc}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}(n)$.

We will now show that there exists a term $s \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$ of size at most n that has at least the same derivation height, witnessed by a "simulation" of the above $\to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$-derivation using $\to_{\mathcal{R}/\mathcal{S}}$.

If $\mathrm{root}(t) \in \Sigma_d^{\mathcal{R}\cup\mathcal{S}}$, t does not contain argenc or enc_f for any f since t is a basic term. Moreover, no rewrite sequence starting with t can use the rules in $\mathcal{G}$ since none of the rules reachable from t introduce any of the symbols argenc or enc_f for some f. Therefore, the above $\to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$-sequence starting from t is an $\to_{\mathcal{R}/\mathcal{S}}$-sequence of the same length. To get a term s over the original signature $\Sigma^{\mathcal{R}\cup\mathcal{S}}$, we replace all occurrences of function symbols cons_f by the corresponding $f \in \Sigma_d^{\mathcal{R}\cup\mathcal{S}}$, i.e., we set $s = \mathrm{dv}(t)$. Thus, we have $\mathrm{dh}(t, \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}) \leq \mathrm{dh}(s, \to_{\mathcal{R}/\mathcal{S}}) \leq \mathrm{dc}_{\mathcal{R}/\mathcal{S}}(n)$.

Now consider $\mathrm{root}(t) = \mathsf{argenc}$, i.e., $t = \mathsf{argenc}(u)$ for a constructor term $u \in \mathcal{T}(\Sigma_c^{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}}, \mathcal{V})$. We can simulate the $\to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$-derivation starting with t from

the term $s = \mathrm{dv}(u)$. In this simulation, we omit the $\to_{\mathcal{G}}$-steps from the original $\to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$-derivation and obtain a $\to_{\mathcal{R}/\mathcal{S}}$-derivation with the same number of $\to_{\mathcal{R}}$-steps and hence the same derivation height. As $|s| \leq n$ ($|\mathrm{dv}(u)| = |u|$ can be shown analogously to Lemma 8), we have $\mathrm{dh}(t, \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}) \leq \mathrm{dh}(s, \to_{\mathcal{R}/\mathcal{S}}) \leq \mathrm{dc}_{\mathcal{R}/\mathcal{S}}(n)$.

Finally, let $\mathrm{root}(t) = \mathsf{enc}_f$ for some $f \in \Sigma_d^{\mathcal{R}\cup\mathcal{S}}$ and thus $t = \mathsf{enc}_f(u_1, \ldots, u_k)$ for some terms $u_1, \ldots, u_k \in \mathcal{T}(\Sigma_c^{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}}, \mathcal{V})$. By construction, the first rewrite step in the above $\to^*_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}}$-derivation must be $t \to_{\mathcal{G}} f(\mathsf{argenc}(u_1), \ldots, \mathsf{argenc}(u_k))$. We again obtain $s \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$ as the start term for our simulation from $f(u_1, \ldots, u_k)$ as $\mathrm{dv}(f(u_1, \ldots, u_k))$, and analogously to the case $\mathrm{root}(t) = \mathsf{argenc}$, we again have $\mathrm{dh}(t, \to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}) \leq \mathrm{dh}(s, \to_{\mathcal{R}/\mathcal{S}}) \leq \mathrm{dc}_{\mathcal{R}/\mathcal{S}}(n)$. □

To prove the corresponding theorem for *innermost* derivational complexity, we use an additional lemma which lets us simulate innermost rewrite steps $s \xrightarrow{\mathrm{i}}_{\mathcal{R}/\mathcal{S}} t$ via $s\sigma_s \xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} t\sigma_s$. (This is a priori not completely obvious since innermost rewriting is not closed under substitutions nor under addition of rewrite rules.)

Lemma 13 (Innermost simulation with generator rules). *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS and let $\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})$ be its runtime instrumentation. Let $s, t \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$. Let σ be a substitution with $s\sigma = s\sigma_s$.*

(a) If $s \xrightarrow{\mathrm{i}}_{\mathcal{R}\cup\mathcal{S}} t$, then $s\sigma \xrightarrow{\mathrm{i}}_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} t\sigma$.
(b) If $s \xrightarrow{\mathrm{i}}_{\mathcal{R}/\mathcal{S}} t$, then $s\sigma \xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} t\sigma$.

Proof. (a) Let $s, t \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}}, \mathcal{V})$ such that $s \xrightarrow{\mathrm{i}}_{\mathcal{R}\cup\mathcal{S}} t$. Then we also have $s \xrightarrow{\mathrm{i}}_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} t$ since s, t do not contain any defined symbols from $\mathcal{G}$. Moreover, we have $s\sigma \xrightarrow{\mathrm{i}}_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}} t\sigma$ since the introduced function symbol argenc does not occur in $\mathcal{R}\cup\mathcal{S}$, since argenc does not occur below the root of a left-hand side of $\mathcal{G}$, and since argenc occurs in $s\sigma$ only in subterms of the shape $\mathsf{argenc}(x)$ for variables x, whereas all argenc-rules in $\mathcal{G}$ require a function symbol below the root of a potential redex.

(b) Follows directly from (a).

Theorem 14 (Innermost derivational complexity via innermost runtime complexity). *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS and let $\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})$ be its runtime instrumentation. Then for all $n \in \mathbb{N}$, we have $\mathrm{idc}_{\mathcal{R}/\mathcal{S}}(n) = \mathrm{irc}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}(n)$.*

Proof. **(1)** $\mathrm{idc}_{\mathcal{R}/\mathcal{S}}(n) \leq \mathrm{irc}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}(n)$.

The proof for this direction of the theorem is analogous to the one for Thm. 12. The only difference is that Lemma 13 is required to show that $t_i \xrightarrow{\mathrm{i}}_{\mathcal{R}/\mathcal{S}} t_{i+1}$ implies $t_i\sigma_t \xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} t_{i+1}\sigma_t$.

(2) $\mathrm{idc}_{\mathcal{R}/\mathcal{S}}(n) \geq \mathrm{irc}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}(n)$.

Let $n \in \mathbb{N}$ and let $t \in \mathcal{T}(\Sigma^{\mathcal{R}\cup\mathcal{S}\cup\mathcal{G}}, \mathcal{V})$ be an arbitrary basic term for $\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})$ with $|t| \leq n$ starting a $\xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$-rewrite sequence

$$t = t_0 \xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} t_1 \xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} t_2 \xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})} \cdots$$

of maximal length for all terms of size at most n, i.e., (i) $\mathrm{dh}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}) = \mathrm{irc}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}(n)$.

We will now show that there exists a term $s \in \mathcal{T}^{\mathcal{R}\cup\mathcal{S}}$ that has at least the same derivation height, again witnessed by a simulation of the above $\to_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$-derivation using $\to_{\mathcal{R}/\mathcal{S}}$.

If $\mathrm{root}(t) \in \mathcal{T}_d^{\mathcal{R}\cup\mathcal{S}}$, no rewrite sequence starting with t can use the rules in $\mathcal{G}$ since none of the rules reachable from the basic term t introduce any of the symbols argenc or enc_f for some f. Therefore, the above $\xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$-sequence is an $\xrightarrow{\mathrm{i}}_{\mathcal{R}/\mathcal{S}}$-sequence of the same length.

For our simulation, we still need to obtain a start term over the original signature $\mathcal{T}^{\mathcal{R}\cup\mathcal{S}}$. Simply replacing all cons_f by f as in the proof of Thm. 12 might introduce new $\mathcal{S}$-redexes that, due to innermost rewriting, could prevent further steps in the derivation using $\mathcal{R}$ and lead to a shorter $\xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$-derivation. Thus, to obtain a term s over $\mathcal{T}^{\mathcal{R}\cup\mathcal{S}}$, we replace all maximal "alien subterms" u of t with regard to the signature $\mathcal{T}^{\mathcal{R}\cup\mathcal{S}}$ by corresponding fresh variables x_u. (An alien subterm u of a term t with regard to a (sub-)signature Σ is a subterm of t with a root symbol $h \notin \Sigma$. Here $h = \mathsf{cons}_g$ for some $g \in \mathcal{T}_d^{\mathcal{R}\cup\mathcal{S}}$ may occur.)

The obtained rewrite sequence from s has at least as many $\xrightarrow{\mathrm{i}}_{\mathcal{R}/\mathcal{S}}$-rewrite steps as the original $\xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}$-rewrite sequence had steps from t, and so we have $\mathrm{dh}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}/(\mathcal{S}\uplus\mathcal{G})}) \leq \mathrm{dh}(s, \xrightarrow{\mathrm{i}}_{\mathcal{R}/\mathcal{S}}) \leq \mathrm{idc}_{\mathcal{R}/\mathcal{S}}(n)$.

The cases $\mathrm{root}(t) = \mathsf{argenc}$ and $\mathrm{root}(t) = \mathsf{enc}_f$ are analogous to Thm. 12. Note that by construction, here argenc is always applied to constructor terms as arguments. □

We finish the section by presenting an example that (to the author's knowledge) was out of reach for automated analysis tools for derivational complexity so far, but can now be handled using an off-the-shelf tool for automated inference of runtime complexity bounds. This example is taken from the *Termination Problems Data Base (TPDB)* [36], a collection of examples used at the annual *Termination and Complexity Competition* [16, 35].

Example 15 (`Derivational_Complexity_Full_Rewriting/AG01/#3.12`, TPDB). Consider the following set of rewrite rules $\mathcal{R}$:

$$\begin{aligned}
\mathsf{app}(\mathsf{nil}, y) &\to y\\
\mathsf{app}(\mathsf{add}(n, x), y) &\to \mathsf{add}(n, \mathsf{app}(x, y))\\
\mathsf{reverse}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{reverse}(\mathsf{add}(n, x)) &\to \mathsf{app}(\mathsf{reverse}(x), \mathsf{add}(n, \mathsf{nil}))\\
\mathsf{shuffle}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{shuffle}(\mathsf{add}(n, x)) &\to \mathsf{add}(n, \mathsf{shuffle}(\mathsf{reverse}(x)))
\end{aligned}$$

Using our transformation to a runtime instrumentation, AProVE adds the following generator rules $\mathcal{G}$:

$$\begin{aligned}
\mathsf{argenc}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{argenc}(\mathsf{add}(x_1, x_2)) &\to \mathsf{add}(\mathsf{argenc}(x_1), \mathsf{argenc}(x_2))\\
\mathsf{argenc}(\mathsf{cons}_{\mathsf{app}}(x_1, x_2)) &\to \mathsf{app}(\mathsf{argenc}(x_1), \mathsf{argenc}(x_2))\\
\mathsf{argenc}(\mathsf{cons}_{\mathsf{reverse}}(x_1)) &\to \mathsf{reverse}(\mathsf{argenc}(x_1))\\
\mathsf{argenc}(\mathsf{cons}_{\mathsf{shuffle}}(x_1)) &\to \mathsf{shuffle}(\mathsf{argenc}(x_1))\\
\mathsf{enc}_{\mathsf{nil}} &\to \mathsf{nil}\\
\mathsf{enc}_{\mathsf{add}}(x_1, x_2) &\to \mathsf{add}(\mathsf{argenc}(x_1), \mathsf{argenc}(x_2))\\
\mathsf{enc}_{\mathsf{app}}(x_1, x_2) &\to \mathsf{app}(\mathsf{argenc}(x_1), \mathsf{argenc}(x_2))\\
\mathsf{enc}_{\mathsf{reverse}}(x_1) &\to \mathsf{reverse}(\mathsf{argenc}(x_1))\\
\mathsf{enc}_{\mathsf{shuffle}}(x_1) &\to \mathsf{shuffle}(\mathsf{argenc}(x_1))
\end{aligned}$$

Then AProVE determines $\mathrm{dc}_{\mathcal{R}/\emptyset}(n) \in \mathcal{O}(n^4)$ and $\mathrm{dc}_{\mathcal{R}/\emptyset}(n) \in \Omega(n^3)$. (A manual analysis reveals that $\mathrm{dc}_{\mathcal{R}/\emptyset}(n) \in \Theta(n^4)$.)

For the inference of the upper bound, first a sufficient criterion [12] is used to show that this TRS belongs to a class of TRSs where runtime complexity and innermost runtime complexity coincide. To analyze innermost runtime complexity, the approach by Naaf et al. [30] is applied. Here the search for an upper bound for innermost runtime complexity is encoded as the search for an upper bound for the runtime of integer transition systems. The proof is completed using the tools CoFloCo [10, 11] and KoAT [6] as backends for complexity analysis of integer transition systems.

For the inference of the lower bound, AProVE uses a technique based on rewrite lemmas [13].

4 Related Work

Derivational complexity analysis. There is a significant body of work on automated analysis of derivational complexity [21] of term rewriting systems. Early techniques are based on observations on the induced maximal derivation height by a direct termination proof via reduction orders [20, 27, 28, 22, 31]. Later work also considers modular techniques [37].

Runtime complexity analysis. In recent years, techniques to infer bounds on the runtime complexity of rewrite systems [18] have become a subject of intensive study, both for full and for innermost rewriting strategies [18, 2, 3, 30, 13]. We can use these techniques to analyze the runtime instrumentations generated from our transformation. In this way, we can now analyze of derivational complexity indirectly, e.g., using amortized complexity analysis [29], adaptions of the dependency pair method [18, 32], and further transformational techniques discussed below (see also Ex. 15).

Transformational approaches for proving properties of TRSs. Transformational approaches for proving properties of TRSs have been introduced successfully in the literature before.

For instance, for termination, techniques like semantic labeling [38] or dependency pairs [1] transform rewrite systems in a way that preserves and reflects termination and that often makes the resulting system more amenable to (automated) termination proofs.

For termination of rewriting with different rewrite strategies, a number of transformations have been devised. For example, transformations to context-sensitive rewriting [25] have been proposed for innermost rewriting [8] (later adapted to innermost runtime complexity [19]) and for outermost rewrite strategies [7]. Here termination of the resulting rewrite system w.r.t. a context-sensitive strategy implies termination w.r.t. the original strategy. Similarly, for context-sensitive rewriting, a number of transformations to full rewriting have been proposed [24, 39, 9, 15] such that termination of the transformed TRS w.r.t. full rewriting implies termination of the original TRS w.r.t. the original context-sensitive rewrite strategy. In particular, Giesl and Middeldorp [15] propose a transformation such that termination of the transformed system is equivalent to termination of the original system w.r.t. its context-sensitive rewrite strategy. These transformations often encode aspects of the context-sensitive rewrite strategy by means of rewrite rules. We follow a related idea, but in contrast to the rewrite strategy, in this paper we encode the set of *start terms*.

For complexity analysis, dependency pairs have been adapted in the form of weak dependency pairs [18] and dependency tuples [32]. Further transformational approaches with term rewriting as target formalism for complexity analysis of programming languages have been investigated, e.g., for Prolog [17] and for Java [26]. Here upper bounds on the (innermost) runtime complexity of the obtained TRS (possibly with constraints) are used to draw conclusions on upper bounds for the worst-case time complexity of the input program. Our approach is related in that it also encodes a complexity problem for a source language (here: term rewriting) to a runtime complexity problem.

Similar to us, Frohn and Giesl [12] also relate different complexity properties. They identify a sufficient criterion to identify TRSs where runtime complexity of innermost rewriting and of full rewriting coincide.

However, to the best of the author's knowledge, so far complexity properties for *different sets of start terms* for the same TRS have not been related. This is where the present work comes in.

5 Implementation and Experimental Evaluation

Of course, to make the point that an instrumentation technique such as the present one is of practical interest, automatic analysis tools need to be able to actually prove useful statements about the output of the technique on standard examples. Thus, to assess the practical usefulness of our contributions, we implemented our transformation in the termination and complexity analysis tool

AProVE [14]. First the runtime instrumentation of the derivational complexity problems is computed, and then this generated problem is processed further by existing techniques to find upper or lower bounds for runtime complexity of innermost or full rewriting. The corresponding configurations are labeled "AProVE instrumentation irc" and "AProVE instrumentation rc" in Tables 1 – 4.

As the state of the art against which to compare our contributions, we used the complexity analysis tool TcT [4] from the Termination and Complexity Competition in 2018[1] (for the competition of 2019, no tools had been submitted to analyze derivational complexity of rewriting) to analyze derivational complexity for innermost and for full rewriting. The corresponding tool configurations are labeled "TcT direct idc" and "TcT direct dc" in Tables 1 – 4. Thus far, AProVE featured only rudimentary techniques for analysis of derivational complexity. Therefore, we did not use AProVE as a reference implementation for analysis of derivational complexity, and we deactivated the existing rudimentary techniques for direct analysis of derivational complexity in our experiments.

Additionally, we wanted to assess whether our runtime instrumentation technique could be useful also for existing state-of-the-art tools like TcT for analysis of derivational complexity. To this end, we extracted the runtime instrumentations for the derivational complexity benchmarks and then conducted experiments on the resulting runtime complexity inputs for innermost and for full rewriting using TcT. The time needed for computing the runtime instrumentations themselves is negligible, so we believe that this is a fair comparison that can inform whether it might be worthwhile to add our transformation technique to the portfolio of techniques to analyze derivational complexity in an established tool like TcT.

For inferring lower bounds for dc, it is sound to use lower bounds for irc or rc for the same set of rewrite rules. Similarly, a lower bound for idc can be obtained directly from a lower bound for irc for the same set of rewrite rules. Thus, for computation of lower bounds, we ran the tools AProVE and TcT on corresponding versions of the rewrite systems (configurations "AProVE direct irc", "AProVE direct rc", "TcT direct irc", and "TcT direct rc" in Tables 2 and 4). The purpose of including these configurations was to see to what extent the addition of our generator rules facilitates the search for lower bounds.

As benchmark set, we used the derivational complexity families of the TPDB, version 10.6 [36]. For technical reasons, we restricted ourselves to the 2664 benchmarks for innermost rewriting[2] and the 1754 benchmarks for full rewriting[3] whose rewrite rules satisfy the conditions from Def. 1 that left-hand sides of rewrite rules must not be variables and right-hand sides of rewrite rules must not contain variables that do not occur in the corresponding left-hand sides.[4]

[1] Available at:
`https://www.starexec.org/starexec/secure/details/solver.jsp?id=20651`

[2] Benchmark family: `Derivational_Complexity_Innermost_Rewriting`

[3] Benchmark family: `Derivational_Complexity_Full_Rewriting`

[4] Version 10.6 of the TPDB contains 60 further examples for derivational complexity of innermost rewriting and 55 further examples for derivational complexity of full rewriting that violate these restrictions.

Tool	$\mathcal{O}(1)$	$\leq \mathcal{O}(n)$	$\leq \mathcal{O}(n^2)$	$\leq \mathcal{O}(n^3)$	$\leq \mathcal{O}(n^{\geq 4})$
TcT direct idc	1	368	468	481	501
TcT instrumentation irc	3	465	555	626	691
AProVE instrumentation irc	13	598	769	827	833

Table 1. Upper bounds for derivational complexity of innermost rewriting

Tool	$\geq \Omega(n)$	$\geq \Omega(n^2)$	$\geq \Omega(n^3)$	$\geq \Omega(n^{\geq 4})$	EXP
TcT direct idc	0	0	0	0	0
TcT direct irc	913	10	10	10	10
AProVE direct irc	1047	205	140	140	139
TcT instrumentation irc	893	10	10	10	10
AProVE instrumentation irc	1082	169	135	135	134

Table 2. Lower bounds for derivational complexity of innermost rewriting

We ran our experiments on the StarExec compute cluster [33] in the `all.q` queue with a timeout of 300 seconds per example.

Tables 1 – 4 give an overview over our experimental results. For each considered configuration, we state the number of examples for which the corresponding asymptotic complexity bound could be inferred. More precisely, a row "$\leq \mathcal{O}(n^k)$" means that the corresponding tools proved a bound $\leq \mathcal{O}(n^k)$ (e.g., in Table 1, the configuration "TcT direct idc" proved constant or linear upper bounds in 368 cases). The column "EXP" in Table 2 and Table 4 refers to an unspecified exponential.

Upper bounds for innermost rewriting. Table 1 provides our experimental data for inference of upper bounds for innermost rewriting. As evidenced by the results, both TcT and AProVE benefit significantly from using our instrumentation rather than relying on existing techniques. For example, the 2018 version of TcT inferred constant or linear upper bounds for 368 TRSs. In contrast, TcT with our instrumentation found constant or linear upper bounds for 465 TRSs, and AProVE found constant or linear upper bounds for 598 TRSs. This indicates that our technique is particularly useful for finding upper complexity bounds.

Lower bounds for innermost rewriting. In Table 2 we present our data for the inference of lower bounds for innermost rewriting. Here the analysis of innermost runtime complexity of the runtime instrumentation and the analysis of the original TRSs are roughly on par. In particular, approximately the same numbers of exponential bounds could be found.

Upper bounds for full rewriting. Table 3 presents our data for upper bounds of derivational complexity for rewriting with arbitrary strategies. Here we observe that the 2018 version of TcT scores noticeably better than our instrumentation-based approach. We conjecture that this is because a number of advanced

Tool	$\mathcal{O}(1)$	$\leq \mathcal{O}(n)$	$\leq \mathcal{O}(n^2)$	$\leq \mathcal{O}(n^3)$	$\leq \mathcal{O}(n^{\geq 4})$
TcT direct dc	1	366	466	479	499
TcT instrumentation rc	1	203	224	304	304
AProVE instrumentation rc	1	328	386	398	399

Table 3. Upper bounds for derivational complexity of full rewriting

Tool	$\geq \Omega(n)$	$\geq \Omega(n^2)$	$\geq \Omega(n^3)$	$\geq \Omega(n^{\geq 4})$	EXP
TcT direct dc	0	0	0	0	0
TcT direct rc	415	0	0	0	0
AProVE direct rc	451	73	68	68	68
TcT direct irc	345	0	0	0	0
AProVE direct irc	426	59	54	54	54
TcT instrumentation rc	378	0	0	0	0
AProVE instrumentation rc	456	68	65	65	65

Table 4. Lower bounds for derivational complexity of full rewriting

techniques (e.g., [19, 29, 30, 32]) for analysis of runtime complexity are available only for innermost rewriting. Still, Ex. 15 shows that also here bounds on derivational complexity can now be found that were out of reach before.

Lower bounds for full rewriting. Table 4 shows the results for our experiments with respect to lower bounds for arbitrary rewrite strategies. Similar to innermost rewriting, also here the precision of the analysis with and without the generator rules is roughly on par (note that for lower bounds, high bounds are better).

Overall we can conclude that in particular for upper bounds of (innermost) derivational complexity, our instrumentation-based approach provides a good addition to state-of-the-art techniques.

Our experimental data from StarExec is available at the following URL:

`http://www.dcs.bbk.ac.uk/~carsten/eval/rcdc/`

6 Reflections and Conclusion

In this article, we have introduced a transformation technique that allows one to analyze derivational complexity problems in term rewriting via an off-the-shelf analysis tool specialized for the analysis of runtime complexity. We have proved correctness of the technique, and we have performed extensive experiments to validate the practical usefulness of our approach.

We recommend that a complexity analysis tool should use this approach and existing techniques in parallel. For complexity analysis tools specialized to

(innermost) runtime complexity, our transformation can provide an avenue to broadened applicability.

In general, the approach of using instrumentations by rewrite rules to generate the set of "intended" start terms from their representation via "allowed" start terms appears to be underexplored in the analysis of properties of rewrite systems. We believe that this approach is worth investigating further, also for other properties of rewriting.

Acknowledgments. The author wishes to thank Florian Frohn and Jürgen Giesl for valuable discussions and the anonymous reviewers for suggestions and comments that helped to improve the paper.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1–2):133–178, 2000.
2. M. Avanzini, N. Eguchi, and G. Moser. A path order for rewrite systems that compute exponential time functions. In *Proc. RTA '11*, volume 10 of *LIPIcs*, pages 123–138, 2011.
3. M. Avanzini and G. Moser. A combination framework for complexity. *Information and Computation*, 248:22–55, 2016.
4. M. Avanzini, G. Moser, and M. Schaper. TcT: Tyrolean complexity tool. In *Proc. TACAS '16*, volume 9636 of *LNCS*, pages 407–423, 2016.
5. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge Univ. Press, 1998.
6. M. Brockschmidt, F. Emmes, S. Falke, C. Fuhs, and J. Giesl. Analyzing runtime and size complexity of integer programs. *ACM Transactions on Programming Languages and Systems*, 38(4):13:1–13:50, 2016.
7. J. Endrullis and D. Hendriks. Transforming outermost into context-sensitive rewriting. *Logical Methods in Computer Science*, 6(2), 2010.
8. M.-L. Fernández. Relaxing monotonicity for innermost termination. *Information Processing Letters*, 93(3):117–123, 2005.
9. M. C. F. Ferreira and A. L. Ribeiro. Context-sensitive AC-rewriting. In *Proc. RTA '99*, volume 1631 of *LNCS*, pages 286–300, 1999.
10. A. Flores-Montoya and R. Hähnle. Resource analysis of complex programs with cost equations. In *Proc. APLAS '14*, volume 8858 of *LNCS*, pages 275–295, 2014.
11. A. Flores-Montoya. Upper and lower amortized cost bounds of programs expressed as cost relations. In *Proc. FM '16*, volume 9995 of *LNCS*, pages 254–273, 2016.
12. F. Frohn and J. Giesl. Analyzing runtime complexity via innermost runtime complexity. In *Proc. LPAR '17*, volume 46 of *EPiC*, pages 249–268, 2017.
13. F. Frohn, J. Giesl, J. Hensel, C. Aschermann, and T. Ströder. Lower bounds for runtime complexity of term rewriting. *Journal of Automated Reasoning*, 59(1):121–163, 2017.
14. J. Giesl, C. Aschermann, M. Brockschmidt, F. Emmes, F. Frohn, C. Fuhs, J. Hensel, C. Otto, M. Plücker, P. Schneider-Kamp, T. Ströder, S. Swiderski, and R. Thiemann. Analyzing program termination and complexity automatically with AProVE. *Journal of Automated Reasoning*, 58:3–31, 2017.
15. J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4):379–427, 2004.

16. J. Giesl, A. Rubio, C. Sternagel, J. Waldmann, and A. Yamada. The termination and complexity competition. In *Proc. TACAS '19 (3)*, volume 11429 of *LNCS*, pages 156–166, 2019.
17. J. Giesl, T. Ströder, P. Schneider-Kamp, F. Emmes, and C. Fuhs. Symbolic evaluation graphs and term rewriting: A general methodology for analyzing logic programs. In *Proc. PPDP '12*, pages 1–12, 2012.
18. N. Hirokawa and G. Moser. Automated complexity analysis based on the dependency pair method. In *Proc. IJCAR '08*, volume 5195 of *LNAI*, pages 364–379, 2008.
19. N. Hirokawa and G. Moser. Automated complexity analysis based on context-sensitive rewriting. In *Proc. RTA-TLCA '14*, volume 8560 of *LNCS*, pages 257–271, 2014.
20. D. Hofbauer. Termination proofs by multiset path orderings imply primitive recursive derivation lengths. *Theoretical Computer Science*, 105(1):129–140, 1992.
21. D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations. In *Proc. RTA '89*, volume 355 of *LNCS*, pages 167–177, 1989.
22. A. Koprowski and J. Waldmann. Max/plus tree automata for termination of term rewriting. *Acta Cybernetica*, 19(2):357–392, 2009.
23. M. Korp, C. Sternagel, and H. Zankl. CaT (complexity and termination). `http://cl-informatik.uibk.ac.at/software/cat/`.
24. S. Lucas. Termination of context-sensitive rewriting by rewriting. In *Proc. ICALP '96*, volume 1099 of *LNCS*, pages 122–133, 1996.
25. S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1–61, 1998.
26. G. Moser and M. Schaper. From Jinja bytecode to term rewriting: A complexity reflecting transformation. *Information and Computation*, 261:116–143, 2018.
27. G. Moser and A. Schnabl. Proving quadratic derivational complexities using context dependent interpretations. In *Proc. RTA '08*, volume 5117 of *LNCS*, pages 276–290, 2008.
28. G. Moser, A. Schnabl, and J. Waldmann. Complexity analysis of term rewriting based on matrix and context dependent interpretations. In *Proc. FSTTCS '08*, volume 2 of *LIPIcs*, pages 304–315, 2008.
29. G. Moser and M. Schneckenreither. Automated amortised resource analysis for term rewrite systems. In *Proc. FLOPS '18*, volume 10818 of *LNCS*, pages 214–229, 2018.
30. M. Naaf, F. Frohn, M. Brockschmidt, C. Fuhs, and J. Giesl. Complexity analysis for term rewriting by integer transition systems. In *Proc. FroCoS '17*, volume 10483 of *LNAI*, pages 132–150, 2017.
31. F. Neurauter, H. Zankl, and A. Middeldorp. Revisiting matrix interpretations for polynomial derivational complexity of term rewriting. In *Proc. LPAR '10 (Yogyakarta)*, volume 6397 of *LNCS*, pages 550–564, 2010.
32. L. Noschinski, F. Emmes, and J. Giesl. Analyzing innermost runtime complexity of term rewriting by dependency pairs. *Journal of Automated Reasoning*, 51(1):27–56, 2013.
33. A. Stump, G. Sutcliffe, and C. Tinelli. Starexec: A cross-community infrastructure for logic solving. In *Proc. IJCAR '14*, volume 8562 of *LNAI*, pages 367–373, 2014. `https://www.starexec.org/`.
34. J. Waldmann. Matchbox: A tool for match-bounded string rewriting. In *Proc. RTA '04*, volume 3091 of *LNCS*, pages 85–94. Springer, 2004.
35. Wiki. The International Termination Competition (TermComp). `http://termination-portal.org/wiki/Termination_Competition`.

36. Wiki. Termination Problems DataBase (TPDB). `http://termination-portal.org/wiki/TPDB`.
37. H. Zankl and M. Korp. Modular complexity analysis for term rewriting. *Logical Methods in Computer Science*, 10(1), 2014.
38. H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24(1–2):89–105, 1995.
39. H. Zantema. Termination of context-sensitive rewriting. In *Proc. RTA '97*, volume 1232 of *LNCS*, pages 172–186, 1997.