On the Complexity of the Small Term Reachability Problem for Terminating Term Rewriting Systems

Franz Baader
Theoretical Computer Science, TU Dresden, Germany
SCADS.AI Dresden/Leipzig, Germany

Jürgen Giesl
RWTH Aachen University, Aachen, Germany

Abstract

Motivated by an application where we try to make proofs for Description Logic inferences smaller by rewriting, we consider the following decision problem, which we call the small term reachability problem: given a term rewriting system R, a term s, and a natural number n, decide whether there is a term t of size $\leq n$ reachable from s using the rules of R. We investigate the complexity of this problem depending on how termination of R can be established. We show that the problem is NP-complete for length-reducing term rewriting systems. Its complexity increases to N2ExpTime-complete (NExpTime-complete) if termination is proved using a (linear) polynomial order and to PSpace-complete for systems whose termination can be shown using a restricted class of Knuth-Bendix orders. Confluence reduces the complexity to P for the length-reducing case, but has no effect on the worst-case complexity in the other two cases.

2012 ACM Subject Classification Theory of computation → Equational logic and rewriting; Theory of computation → Automated reasoning; Theory of computation → Complexity theory and logic

Keywords and phrases Rewriting, Termination, Confluence, Creating small terms, Derivational complexity, Description Logics, Proof rewriting

Digital Object Identifier 10.4230/LIPIcs.FSCD.2024.16

Funding *Franz Baader*: Partially supported by DFG, Grant 389792660, within TRR 248 "Center for Perspicuous Computing", and by the German Federal Ministry of Education and Research (BMBF, SCADS22B) and the Saxon State Ministry for Science, Culture and Tourism (SMWK) by funding the competence center for Big Data and AI "ScaDS.AI Dresden/Leipzig".
Jürgen Giesl: Partially supported by DFG, Grant 235950644 (Project GI 274/6-2).

1 Introduction

Term rewriting [7, 28] is a well-investigated formalism, which can be used both for computation and deduction. A term rewriting system R consists of rules, which describe how a term s can be transformed into a new term t, in which case one writes $s \to_R t$. In the computation setting, where term rewriting is akin to functional programming [12], a given term (the input) is iteratively rewritten into a normal form (the output), which is a term that cannot be further rewritten. Termination of R prevents infinite rewrite chains, and thus ensures that a normal form can always be reached, whereas confluence guarantees that the output is unique, despite the nondeterminism inherent to the rewriting process (which rule to apply when and where). In the deduction setting, which is, e.g., relevant for first-order theorem proving with equality [25], one is interested in whether a term s can be rewritten into a term t by iteratively applying the rules of R in both directions. If R is confluent and terminating, this problem can be solved by computing normal forms of s and t, and then checking whether they are equal. In the present paper, we want to employ rewriting for a different purpose: given a term s, we are interested in finding a term t of minimal size that can be reached from s by rewriting (written $s \xrightarrow{*}_R t$), but this term need not be in normal form. To assess the

R1 $\dfrac{A \sqsubseteq B \quad B \sqsubseteq C}{A \sqsubseteq C}$ R2 $\dfrac{A \sqsubseteq B}{\exists r.A \sqsubseteq \exists r.B}$ R3 $\dfrac{A \sqsubseteq \exists r.A_1 \quad A_1 \sqsubseteq B_1 \quad \exists r.B_1 \sqsubseteq B}{A \sqsubseteq B}$

Figure 1 Three proof rules for $\mathcal{EL}$.

complexity of this computation problem, we investigate the corresponding decision problem: given a term rewriting system R, a term s, and a natural number n, decide whether there is a term t of size $\leq n$ such that $s \xrightarrow{*}_R t$. We call this the *small term reachability problem.*

Our interest in this problem stems from the work on finding small proofs [3, 4] for Description Logic (DL) inferences [6], which are then visualized in an interactive explanation tool [2]. For the DL $\mathcal{EL}$ [5], we employ the highly-efficient reasoner ELK [20] to compute proofs. However, the proof calculus employed by ELK is rather fine-grained, and thus produces relatively large proofs. Our idea was thus to generate smaller proofs by rewriting several proof steps into a single step. As a (simplified) example, consider the three proof rules in Figure 1. It is easy to see that one needs one application of R2 followed by two of R1 to produce the same consequence as a single application of R3. Thus, if one looks for patterns in a proof that use R1 and R2 in this way, and replaces them by the corresponding applications of R3, then one can reduce the size of a given proof. Given finitely many such proof rewriting rules and a proof, the question is then how to use the rules to rewrite the given proof into one of minimal size. Since tree-shaped proofs as well as DL concept descriptions can be represented as terms, this question can be seen as an instance of the small term reachability problem introduced above.

In this paper, we investigate the complexity of the small term reachability problem on the general level of term rewriting systems (TRSs). It turns out that this complexity depends on how termination of the given TRS can be shown. The paper contains the following main contributions:

1. Small term reachability for length-reducing TRSs

If the introduced rewrite rules are *length-reducing*, i.e., each rewrite step decreases the size of the term (proof), like the rule in our example, then termination of all rewrite sequences is guaranteed. In general, it may nevertheless be the case that one can generate two normal forms of different sizes. Confluence prevents this situation, i.e., then it is sufficient to generate only one rewrite sequence to produce a term (proof) of minimal size. In Section 4 we show that the small term reachability problem for length-reducing term rewriting systems is NP-complete in general, but becomes solvable in polynomial time if we restrict ourselves to confluent systems.

2. Small term reachability for TRSs whose termination is shown by polynomial orders

It also makes sense to consider sets of rules where not every rule is length-reducing, e.g., if one first needs to reshape a proof before a length-reducing rule can be applied, or if one translates between different proof calculi. In this extended setting, termination is no longer trivially given, and thus one first needs to show that the introduced set of rules is terminating, which can be achieved with the help of a reduction order [7, 28]. We show in this paper that the complexity of the small term reachability problem depends on which reduction order is used for this purpose. More precisely, in Section 5 we consider term rewriting systems that can be proved terminating using a polynomial order [22], and show that in this case the small term reachability problem is N2ExpTime-complete, both in the general and the confluent case. If the definition of the polynomial order employs only linear polynomials, then the complexity of the problem is reduced to NExpTime, where again hardness already holds for confluent systems. Here, as usual, NExpTime (N2ExpTime) is the class of all decision problems solvable by a nondeterministic Turing machine in $O(2^{p(n)})$ ($O(2^{2^{p(n)}})$) steps, where n is the size of the problem and $p(n)$ is a polynomial in n.

3. Small term reachability for TRSs whose termination is shown by KBO

In Section 6, we investigate the impact that using a Knuth-Bendix order (KBO) [21] for the termination proof has on the complexity of the small term reachability problem. In the restricted setting without unary function symbols of weight zero, the problem is PSpace-complete, again both in the general and the confluent case. The complexity class PSpace consists of all decision problems solvable by a deterministic Turing machine in $O(p(n))$ space, where n is the size of the problem and $p(n)$ is a polynomial in n.

Our proofs of the results mentioned above strongly depend on work on the derivational complexity of term rewriting systems, which links the reduction order employed for the termination proof with the maximal length of reduction sequences as a function of the size of the start term (see e.g., [14, 15, 16, 23]). To obtain reasonable complexity classes, we restricted ourselves to reduction orders where the resulting bound on the derivational complexity is not "too high". In particular, we use the results of the seminal paper by Hofbauer and Lautemann [16], which show that termination proofs with a (linear) polynomial order yield a double-exponential (exponential) upper bound on the length of derivation sequences whereas termination proofs with a KBO without unary function symbols of weight zero yield an exponential such bound. We also make use of the term rewriting systems employed in the proofs showing that these bounds are tight. A connection between the derivational complexity of term rewriting systems and complexity classes has been established in [9] for polynomial orders and in [10] for Knuth-Bendix orders. While this work considers a different problem since it views term rewriting systems as devices for computing functions by generating a normal form, and uses them to characterize complexity classes, the constructions utilized in the proofs in [9, 10] are similar to the ones we use in our hardness proofs. A notable difference between the two problems is the impact that confluence has on the obtained complexity class: while in our setting confluence only reduces the complexity in the case of length-reducing systems, in [9] it also reduces the complexity (from the nondeterministic to the respective deterministic class) for the case of systems shown terminating with a (linear) polynomial order.

In the next section, we briefly recall basic notions from term rewriting, including the definitions of polynomial and Knuth-Bendix orders. In Section 3, we introduce the small term reachability problem and show that it is undecidable in general, but decidable for terminating systems. Sections 4, 5, and 6 respectively consider the length-reducing, polynomial order, and Knuth-Bendix order case. We conclude with a brief discussion of possible future work.

2 Preliminaries

We assume that the reader is familiar with basic notions and results regarding term rewriting. In this section, we briefly recall the relevant notions, but refer the reader to [7, 28] for details.

Given a finite set of *function symbols* with associated *arities* (called the *signature*) and a disjoint set of *variables*, terms are built in the usual way. Function symbols of arity 0 are also called *constant symbols*. For example, if x, y are variables, c is a constant symbol, and f a binary function symbol, then $c, f(x, c), f(f(x, c), c)$ are terms. The *size* $|t|$ of a term t is the number of occurrences of functions symbols and variables in t (e.g., $|f(f(x, c), c)| = 5$). If f is a function symbol or variable, then $|t|_f$ counts the number of occurrences of f in t (e.g., $|f(f(x, c), c)|_f = 2$). As usual, nested applications of unary function symbols are often written as words. For example, $g(g(h(h(g(x)))))$ is written as $gghhg(x)$ or $g^2h^2g(x)$.

A *rewrite rule* (or simply rule) is of the form $l \to r$ where l, r are terms such that l is not a variable and every variable occurring in r also occurs in l. In this paper, a term rewriting system (TRS) is a *finite* set of rewrite rules, and thus we do not mention finiteness explicitly when formulating our complexity results. A given TRS R induces the binary relation $\to_R$ on terms. Basically, we have $s \to_R t$ if there is a rule $l \to r$ in R such that s contains a substitution instance $\sigma(l)$ of l as subterm, and t is obtained from s by replacing this subterm with $\sigma(r)$. Recall that a *substitution* is a mapping from variables to terms, which is homomorphically extended to a mapping from terms to terms. For example, if R contains the rule $hh(x) \to g(x)$, then $f(hhh(c), c) \to_R f(gh(c), c)$ and $f(hhh(c), c) \to_R f(hg(c), c)$. The reflexive and transitive closure of $\to_R$ is denoted as $\xrightarrow{*}_R$, i.e., $s \xrightarrow{*}_R t$ holds if there are $n \geq 1$ terms $t_1, \ldots, t_n$ such that $s = t_1, t = t_n$, and $t_i \to_R t_{i+1}$ for $i = 1, \ldots, n-1$.

Two terms s_1, s_2 are *joinable* with R if there is a term t such that $s_i \xrightarrow{*}_R t$ holds for $i = 1, 2$. The relation $\to_R$ is *confluent* if $s \xrightarrow{*}_R s_i$ for $i = 1, 2$ implies that s_1 and s_2 are joinable with R. It is *terminating* if there is no infinite reduction chain $t_0 \to_R t_1 \to_R t_2 \to_R \ldots$. If $\to_R$ is confluent (terminating), then we also call R confluent (terminating). The term t is *irreducible* if there is no term t' such that $t \to_R t'$. If $s \xrightarrow{*}_R t$ and t is irreducible, then we call t a *normal form* of s. If R is confluent and terminating, then every term has a unique normal form. If R is terminating, then its confluence is decidable [21]. Termination can be proved using a *reduction order*, which is a well-founded order $\succ$ on terms such that $l \succ r$ for all $l \to r \in R$ implies $s \succ t$ for all terms s, t with $s \to_R t$. Since $\succ$ is well-founded, this then implies termination of R. If $l \succ r$ holds for all $l \to r \in R$, then we say that R can be *shown terminating* with the reduction order $\succ$. The following is a simple reduction order.

▶ **Example 1.** If we define $s \succ t$ if $|s| > |t|$ and $|s|_x \geq |t|_x$ for all variables x, then $\succ$ is a reduction order (see Exercise 5.5 in [7]). For example, $hh(x) \succ g(x)$, and thus the TRS $R = \{hh(x) \to g(x)\}$ is terminating. As illustrated in Example 5.2.2 in [7], the condition on variables is needed to obtain a reduction order.

This order can only show termination of *length-reducing* TRSs R, i.e., where $s \to_R t$ implies $|s| > |t|$. We now recapitulate the definitions of more powerful reduction orders [7, 28].

Polynomial orders

To define a polynomial order, one assigns to every n-ary function symbol f a polynomial P_f with coefficients in the natural numbers $\mathbb{N}$ and n indeterminates such that P_f depends on all these indeterminates. To ensure that this implies (strong) monotonicity of the polynomial order, we require that constant symbols c must be assigned a polynomial of degree 0 whose coefficient is > 0. Such an assignment also yields an assignment of polynomials P_t to terms t.

▶ **Example 2.** Assume that $+$ is binary, s, d, q are unary, and 0 is a constant. We assign the polynomial $P_+ = x + 2y + 1$ to $+$, $P_s = x + 2$ to s, $P_d = 3x + 1$ to d, $P_q = 3x^2 + 3x + 1$ to q, and $P_0 = 3$ to 0. For the terms $l = q(s(x))$ and $r = q(x) + s(d(x))$ we then obtain the associated polynomials $P_l = 3(x + 2)^2 + 3(x + 2) + 1 = 3x^2 + 15x + 19$ and $P_r = 3x^2 + 3x + 1 + 2(3x + 1 + 2) + 1 = 3x^2 + 9x + 8$.

The polynomial order induced by such an assignment is defined as follows: $t \succ t'$ if P_t evaluates to a larger natural number than $P_{t'}$ for every assignment of natural numbers > 0 to the indeterminates of P_t and $P_{t'}$. In our example, the evaluation of P_l is obviously always larger than the evaluation of P_r, and thus $l \succ r$. As shown, e.g., in Section 5.3 of [7], polynomial orders are reduction orders, and thus can be used to prove termination of TRSs.

Knuth-Bendix orders

To define a Knuth-Bendix order (KBO), one must assign a weight $w(f)$ to all function symbols and variables, and define a strict order $>$ on the function symbols (called *precedence*) such that the following is satisfied:

- All weights $w(f)$ are non-negative real numbers, and there is a weight $w_0 > 0$ such that $w(x) = w_0$ for all variables x and $w(c) \geq w_0$ for all constant symbols c.
- If there is a unary function symbol h with $w(h) = 0$, then h is the greatest element w.r.t. $>$, i.e., $h > f$ for all function symbols $f \neq h$. Such a unary function symbol h is then called a *special* symbol. Obviously, there can be at most one special symbol.

Since in this paper we only consider KBOs without special symbol, we restrict our definition of KBOs to this case. A given weight function w and strict order $>$ without special symbol induces the following KBO $\succ$: $s \succ t$ if $|s|_x \geq |t|_x$ for all variables x and

- $w(s) > w(t)$, where $w(u) := \sum_{f \text{ occurs in } u} w(f) \cdot |u|_f$ for all terms u, or
- $w(s) = w(t)$ and one of the following two conditions is satisfied:
  - $s = f(s_1, \ldots, s_m)$, $t = g(t_1, \ldots, t_n)$, and $f > g$.
  - $s = f(s_1, \ldots, s_m)$, $t = f(t_1, \ldots, t_m)$, and there is $i, 1 \leq i \leq m$, such that $s_1 = t_1, \ldots, s_{i-1} = t_{i-1}$, and $s_i \succ t_i$.

A proof of the fact that KBOs are reduction orders can, e.g., be found in Section 5.4.4 of [7].

▶ **Example 3.** Let $0, 1, 1'$ be unary function symbols and c a constant symbol, and consider the following TRS, which is similar to the one introduced in the proof of Lemma 7 in [10]:

$$R = \{1(c) \to 0(c), 0(c) \to 1'(c), 0(1'(x)) \to 1'(1(x)), 1(1'(x)) \to 0(1(x))\}.$$

Basically, this TRS realizes a binary down counter, and thus it is easy to see that, starting with the binary representation $10^n(c)$ of the number 2^n, the TRS R can make $\geq 2^n$ reduction steps to arrive at the term $0^{n+1}(c)$. For example, $100(c) \to_R 101'(c) \to_R 11'1(c) \to_R 011(c) \to_R 010(c) \to_R 011'(c) \to_R 001(c) \to_R 000(c)$. Termination of R can be shown using the following KBO: assign weight 1 to all function symbols and variables, and use the precedence order $1 > 0 > 1'$.

3 Problem definition and (un)decidability results

In this paper, we investigate the complexity of the following decision problem.

▶ **Definition 4.** *Given a TRS R, a term s, and a natural number n, the* small term reachability problem *asks whether there exists a term t such that $s \xrightarrow{*}_R t$ and $|t| \leq n$.*

The name "small term reachability problem" is motivated by the fact that we want to use the TRS R to turn a given term s into a term whose size is as small as possible. The introduced problem is the decision variant of this computation problem. A solution to the computation problem, which computes a term t of minimal size reachable with R from s, of course also solves the decision variant of the problem. Thus, complexity lower bounds for the decision problem transfer to the computation problem.

It is easy to see that this problem is in general undecidable, but decidable for terminating TRSs. For non-terminating systems, confluence is not sufficient to obtain decidability.

▶ **Proposition 5.** *The small term reachability problem is in general undecidable for confluent TRSs, but is decidable for systems that are terminating.*

Proof. Undecidability in the general case follows, e.g., from the fact that TRSs can simulate Turing machines (TMs) [17]. (We will also use Turing machines for the proofs of the hardness results in the remainder of the paper.) More precisely, the reduction introduced in Section 5.1.1 of [7] transforms a given TM $\mathcal{M}$ into a TRS $R_{\mathcal{M}}$ such that (among other things) the following holds: there is an infinite run of $\mathcal{M}$ on the empty input iff there is an infinite reduction sequence of $R_{\mathcal{M}}$ starting with the term s_0 that encodes the initial configuration of $\mathcal{M}$ for the empty input. In addition, if $\mathcal{M}$ is deterministic, then $R_{\mathcal{M}}$ is confluent. We can now add rules to $R_{\mathcal{M}}$ that apply to all terms encoding a halting configuration of $\mathcal{M}$, and trigger further rules that reduce such a term to one of size 1. Since the term s_0 has size larger than one and the rules of $R_{\mathcal{M}}$ never decrease the size of a term, this yields a reduction of the (undecidable) halting problem for deterministic TMs to the small term reachability problem for confluent TRSs.

Given a terminating TRS R and a term s, we can systematically generate all terms reachable from s by iteratively applying $\to_R$. Since R is finite, $\to_R$ is finitely branching. Together with termination, this means (by König's Lemma) that there are only finitely many terms reachable with R from s (see Lemma 2.2.4 in [7]). We can then check whether, among them, there is a term of size at most n. ◀

In the following, we study the *complexity* of the small term reachability problem for terminating TRSs, depending on how their termination can be shown.

4 Length-reducing term rewriting systems

In this section, we investigate the complexity of the small term reachability problem for length-reducing TRSs, i.e., TRSs where each rewrite step decreases the size of the term.

We start with showing an *NP upper bound.* Let R, s, n be an instance of the small term reachability problem, where R is assumed to be length-reducing. This assumption implies that the length k of any rewrite sequence $s \to_R s_1 \to_R s_2 \to_R \ldots \to_R s_k$ issuing from s is bounded by $|s|$. In addition, for each term s_i there are only polynomially many terms s' (in the size of s and R) such that $s_i \to_R s'$. Thus, the following yields an NP procedure for deciding the small term reachability problem:

- guess a rewrite sequence $s \to_R s_1 \to_R s_2 \to_R \ldots \to_R s_k$ of length $k \leq |s|$;
- check whether $|s_k| \leq n$ holds. If the answer is "yes" then accept, and reject otherwise.

▶ **Lemma 6.** *The small term reachability problem is in NP for length-reducing TRSs.*

If the length-reducing system R is confluent, then it is sufficient to generate an arbitrary *terminating* (i.e., maximal) rewrite sequence starting in s, i.e., a sequence $s \to_R s_1 \to_R s_2 \to_R \ldots \to_R s_k$ such that s_k is irreducible. Obviously, we have $k \leq |s|$, and thus such a sequence can be generated in polynomial time. We claim that there is a term t of size $\leq n$ reachable from s iff $|s_k| \leq n$. Otherwise, the smallest term t reachable from s is different from s_k. But then t and s_k are both reachable from s, and thus must be joinable due to the confluence of R. As s_k is irreducible, this implies $t \to_R^* s_k$ and thus, $|t| \geq |s_k|$, i.e., t is not smaller than s_k.

▶ **Proposition 7.** *For confluent length-reducing TRSs, the small term reachability problem can be decided in deterministic polynomial time.*

In general, however, the problem is NP-hard. We prove *NP-hardness* by showing that any polynomially time bounded nondeterministic Turing machine can be simulated by a length-reducing TRS. Thus, assume that $\mathcal{M}$ is such a TM and that its time-bound is given by the polynomial p. As in [7] we assume that in every step $\mathcal{M}$ either moves to the left or to the right, where the tape of the TM is infinite in both directions. In addition, we assume without loss of generality that $\mathcal{M}$ has exactly one accepting state $\widehat{q}$. We view the tape symbols of $\mathcal{M}$ as unary function symbols and the states of $\mathcal{M}$ as binary function symbols. We assume that q_0 is the initial state of $\mathcal{M}$ and that b is the blank symbol. Furthermore, let $\#$ be a constant symbol and f be a unary function symbol different from the tape symbols.

Given an input word $w = a_1 \dots a_\ell$ for $\mathcal{M}$, we construct the term

$$t(w) := b^{p(\ell)}(q_0(a_1 \dots a_\ell b^{p(\ell)-\ell}(\#), f^{p(\ell)}(\#))).$$

Intuitively, the starting b symbols together with the first argument of q_0 in $t(w)$ provide a tape that is large enough for a $p(\ell)$-time bounded TM to run on for the given input w of length ℓ. The first argument of a state symbol represents the part of the tape that starts at the position of the head. Thus, in $t(w)$, a_1 is the tape symbol at the position of the head and $a_2 \dots a_\ell$ are the symbols to the right of it. The second argument of a state symbol is a unary down counter from which one f is removed in every step that $\mathcal{M}$ makes. This is needed to ensure that the constructed TRS is length-reducing. This counter is large enough to allow $\mathcal{M}$ to make the maximal possible number of $p(\ell)$ steps.

Basically, we now express the transitions of $\mathcal{M}$ as usual by rewrite rules (as, e.g., done in Definition 5.1.3 of [7]), but with three differences:

- since the term $t(w)$ provides enough tape for a TM that can make at most $p(\ell)$ steps, the special cases that treat a situation where the end of the represented tape is reached and one has to add a blank are not needed;
- since we fix as start term $t(w)$ a configuration term (i.e., a term that encodes a configuration of the TM), the additional effort expended in [7] to deal with non-configuration terms (by using copies of symbols with arrows to the left or right) is not needed;
- we have the additional counter in the second argument, which removes one f in every step, and thus ensures that rule application is length-reducing.

The TRS $R^{\mathcal{M}}$ that simulates $\mathcal{M}$ has the following rewriting rules:

- For each transition (q, a, q', a', r) of $\mathcal{M}$ it has the rule $q(a(x), f(y)) \to a'(q'(x, y))$. Thus, the tape symbol a is replaced by a' and the head of the TM is now at the position to the right of it.
- For each transition (q, a, q', a', l) of $\mathcal{M}$ it has the rule $c(q(a(x), f(y))) \to q'(ca'(x), y)$ for every tape symbol c of $\mathcal{M}$. Thus, a is replaced by a' and the head of the TM is now at the position to the left of it.

Note that the blank symbol b is also considered as a tape symbol of $\mathcal{M}$.

In addition, we add rules to $R^{\mathcal{M}}$ that can be used to generate the term $\#$, which has size 1, whenever $\widehat{q}$ is reached:

- $a(\widehat{q}(x, y)) \to \widehat{q}(x, y)$ for every tape symbol a of $\mathcal{M}$,
- $\widehat{q}(x, y) \to \#$.

The following is now easy to see.

▶ **Lemma 8.** *The term $t(w)$ can be rewritten with $R^{\mathcal{M}}$ to a term of size 1 iff $\mathcal{M}$ accepts the word w.*

Proof. It is easy to see that $R^{\mathcal{M}}$ simulates $\mathcal{M}$ in the sense that there is a run of $\mathcal{M}$ on input $w = a_1 \ldots a_\ell$ that reaches the accepting state $\widehat{q}$ iff there is a rewrite sequence of $R^{\mathcal{M}}$ starting with $t(w)$ that reaches a term of the form $u(\widehat{q}(t, t'))$, where u is a word over the tape symbols of $\mathcal{M}$ and t, t' are terms. Note that the assumption that $\mathcal{M}$ is $p(\ell)$-time bounded together with the construction of $t(w)$ ensures that there is enough tape space and the counter is large enough for the simulation of $\mathcal{M}$ to run through completely.

Thus, if $\mathcal{M}$ accepts $w = a_1 \ldots a_\ell$, then we can rewrite $t(w)$ with $R^{\mathcal{M}}$ into a term of the form $u(\widehat{q}(t, t'))$, and this term can then be further rewritten into #, which has size 1. If $\mathcal{M}$ does not accept $w = a_1 \ldots a_\ell$, then the state $\widehat{q}$ cannot be reached by any run of $\mathcal{M}$ starting with this word. Thus, all terms reachable from $t(w)$ with the rules of $R^{\mathcal{M}}$ that simulate $\mathcal{M}$ are of the form $u(q(t, t'))$ for states q different from $\widehat{q}$. The rules of $R^{\mathcal{M}}$ of the second kind are thus not applicable, and the terms of the form $u(q(t, t'))$ clearly have size > 1. ◀

We are now ready to show the corresponding complexity lower bound.

▶ **Lemma 9.** *The small term reachability problem for length-reducing TRSs is NP-hard.*

Proof. We show that every problem Π in NP can be reduced in polynomial time to our problem. Let $\mathcal{M}$ be the nondeterministic Turing machine that is an NP decision procedure for Π, and let p be the polynomial that bounds the length of runs of $\mathcal{M}$. We can construct the length-reducing TRS $R^{\mathcal{M}}$ as described above. Given a word $w = a_1 \ldots a_\ell$, we can compute the term $t(w)$ in polynomial time, and Lemma 8 implies that this yields a reduction function from Π to the small term reachability problem for the length-reducing TRS $R^{\mathcal{M}}$. ◀

Combining the obtained upper and lower bounds, we thus have determined the exact complexity of the problem under consideration.

▶ **Theorem 10.** *The small term reachability problem is NP-complete for length-reducing TRSs.*

To show that a given TRS R is length-reducing, one can, for example, use the reduction order of Example 1. This order also applies to the TRS $R^{\mathcal{M}}$ introduced above.

5 Term rewriting systems shown terminating with a polynomial order

An interesting question is whether similar results can be obtained for TRSs whose termination can be shown using a reduction order from a class of such orders that provides an upper bound on the length of reduction sequences. For example, it is known that a proof of termination using a polynomial order yields a double-exponential upper bound on the length of reduction sequences [16]. One possible conjecture could now be that, for TRSs whose termination can be shown using a polynomial order, the small term reachability problem is N2ExpTime-complete.

The *upper bound* is easy to show since, again, one just needs to guess a reduction sequence, but now of double-exponential length, and then check the size of the obtained term. This yields a nondeterministic double-exponential time procedure for solving the small term reachability problem for TRSs whose termination can be shown using a polynomial order.

▶ **Lemma 11.** *The small term reachability problem is in N2ExpTime for TRSs whose termination can be shown using a polynomial order.*

Regarding the *lower bound*, the idea is now to use basically the same approach as employed in Section 4, but generate a double-exponentially large tape and a double-exponentially large counter with the help of a TRS whose termination can be shown using a polynomial order.

For this, we want to re-use the original system introduced by Hofbauer and Lautemann showing that the double-exponential upper bound is tight (see Example 5.3.12 in [7]).

▶ **Example 12.** Let R_{HL} be the TRS consisting of the following rules:

$$\begin{array}{llll} x+0\to x, & x+s(y)\to s(x+y), & d(0)\to 0, & d(s(x))\to s(s(d(x))), \\ q(0)\to 0, & q(s(x))\to q(x)+s(d(x)). & & \end{array}$$

The TRS R_{HL} intuitively defines the arithmetic functions addition ($+$), double (d), and square (q) on non-negative integers. Thus, it is easy to see that the term $t_n := q^n(s^2(0))$ can be reduced to $s^{2^{2^n}}(0)$. The polynomial order in Example 2 shows termination of R_{HL}.

Now, assume that $\mathcal{M}$ is a double-exponentially time bounded nondeterministic TM and that its time-bound is $2^{2^{p(\ell)}}$ for a polynomial p, where ℓ is the length of the input word. Given an input word $w = a_1 \dots a_\ell$ for $\mathcal{M}$, we construct the term

$$t(w) := q_1^{p(\ell)}(bb(q_0(a_1 \dots a_\ell q_2^{p(\ell)}(bb(\#)), q_3^{p(\ell)}(ff(\#))))).$$

The idea underlying this definition is that the term $q_1^{p(\ell)}(bb(q_0(\cdot))$ can be used to generate a tape segment before the read-write head of the TM (marked by the state q_0) with $2^{2^{p(\ell)}}$ blanks using the following modified version of R_{HL}:

$$\begin{array}{lll} R_1 := \{\, q_0(y_1,y_2) +_1 q_0(z_1,z_2) \to q_0(y_1,y_2), & b(x) +_1 q_0(z_1,z_2) \to b(x), & x +_1 b(y) \to b(x +_1 y), \\ \quad d_1(q_0(z_1,z_2)) \to q_0(z_1,z_2), & d_1(b(x)) \to b(b(d_1(x))), & \\ \quad q_1(q_0(z_1,z_2)) \to q_0(z_1,z_2), & q_1(b(x)) \to q_1(x) +_1 b(d_1(x))\}. & \end{array}$$

Here b plays the rôle of the successor function s in R_{HL}, terms of the form $q_0(\cdot)$ play the rôle of the zero 0 in R_{HL}, and $+_1$, d_1, and q_1 correspond to addition, double, and square. Instead of the rule $x +_1 q_0(z_1, z_2) \to x$ we considered two rules for the case where x is built with q_0 or with b, respectively. The reason will become clear later when we consider the restriction to confluent TRSs. Lemma 13 is an easy consequence of our observations regarding R_{HL}.

▶ **Lemma 13.** *For any two terms t_1, t_2, we can rewrite the term $q_1^{p(\ell)}(bb(q_0(t_1,t_2)))$ with R_1 into the term $b^{2^{2^{p(\ell)}}}(q_0(t_1,t_2))$.*

Next, we define a copy of R_{HL} that allows us to create a tape segment with $2^{2^{p(n)}}$ blanks to the right of the input word:

$$\begin{array}{lllll} R_2 & := & \{\# +_2 \# \to \#, & b(y) +_2 \# \to b(y), & x +_2 b(y) \to b(x +_2 y), \\ & & d_2(\#) \to \#, & d_2(b(x)) \to b(b(d_2(x))), & \\ & & q_2(\#) \to \# & q_2(b(x)) \to q_2(x) +_2 b(d_2(x))\}. & \end{array}$$

▶ **Lemma 14.** *The term $q_2^{p(\ell)}(bb(\#))$ rewrites with R_2 to the term $b^{2^{2^{p(\ell)}}}(\#)$.*

The double-exponentially large counter can be generated by the following copy of R_{HL}:

$$\begin{array}{lllll} R_3 & := & \{\# +_3 \# \to \#, & f(y) +_3 \# \to f(y), & x +_3 f(y) \to f(x +_3 y), \\ & & d_3(\#) \to \#, & d_3(f(x)) \to f(f(d_3(x))), & \\ & & q_3(\#) \to \# & q_3(f(x)) \to q_3(x) +_3 f(d_3(x))\}. & \end{array}$$

▶ **Lemma 15.** *The term $q_3^{p(\ell)}(ff(\#))$ rewrites with R_3 to the term $f^{2^{2^{p(\ell)}}}(\#)$.*

We now add to these three TRSs the system $R^{\mathcal{M}}$, which can simulate $\mathcal{M}$ and then make the term small in case the accepting state $\widehat{q}$ is reached. For the following lemma we assume, as before, that $\widehat{q}$ is the only accepting state. In addition, we assume without loss of generality that the initial state q_0 is not reachable, i.e., as soon as the machine has made a transition, it is in a state different from q_0 and cannot reach state q_0 again.

▶ **Lemma 16.** *The term $t(w)$ can be rewritten with $R^{\mathcal{M}} \cup R_1 \cup R_2 \cup R_3$ to a term of size 1 iff $\mathcal{M}$ accepts the word w.*

Proof. First, assume that $\mathcal{M}$ accepts the word w. Then there is a run of $\mathcal{M}$ on input w such that the accepting state $\widehat{q}$ is reached. We can simulate this run, starting with $t(w)$ by first using $R_1 \cup R_2 \cup R_3$ to generate the term

$$b^{2^{2^{p(\ell)}}}(q_0(a_1 \dots a_\ell b^{2^{2^{p(\ell)}}}(\#), f^{2^{2^{p(\ell)}}}(\#))).$$

Since the tape and counter generated this way are large enough, $R^{\mathcal{M}}$ can then simulate the accepting run of $\mathcal{M}$, and the last two rules of $R^{\mathcal{M}}$ can be used to generate the term #, which has size 1.

For the other direction, we first note that a term of size 1 can only be reached from $t(w)$ using $R^{\mathcal{M}} \cup R_1 \cup R_2 \cup R_3$ if a term is reached that contains $\widehat{q}$. This function symbol can only be generated by performing transitions of $\mathcal{M}$, starting with the input w. In fact, while the simulation of $\mathcal{M}$ can start before the system $R_1 \cup R_2 \cup R_3$ has generated the tape and the counter in full size, rules of $R^{\mathcal{M}}$ can only be applied if the TM locally sees a legal tape configuration. This means that blanks generated by R_1 and R_2 can be used even if the application of these systems has not terminated yet. But if one of the auxiliary symbols employed by these systems is encountered, then no rule simulating a transition of $\mathcal{M}$ is applicable. These systems cannot generate tape symbols other than blanks, and these blanks are also available to $\mathcal{M}$ in its run. Thus, $R^{\mathcal{M}} \cup R_1 \cup R_2 \cup R_3$ can only generate a term containing $\widehat{q}$ if there is a run of $\mathcal{M}$ on input w that reaches $\widehat{q}$. ◀

To conclude from this lemma that the small term reachability problem is N2ExpTime-hard for TRSs whose termination can be shown using a polynomial order, it is enough to prove the following result.

▶ **Lemma 17.** *Termination of $R^{\mathcal{M}} \cup R_1 \cup R_2 \cup R_3$ can be shown using a polynomial order.*

Proof. Termination of $R^{\mathcal{M}} \cup R_1 \cup R_2 \cup R_3$ can be shown using the following polynomial interpretation of the function symbols:

- $a(x)$ is mapped to $x + 2$, for all tape symbols a of the TM (where a can also be the blank symbol b),
- # is mapped to 3,
- $q(x, y)$ is mapped to $x + y + 3$, for all states q of the TM, in particular also for q_0 and $\hat{q}$,
- $f(x)$ is mapped to $x + 2$,
- $+_1(x, y)$, $+_2(x, y)$, and $+_3(x, y)$ are mapped to $x + 2y + 1$,
- $d_1(x)$, $d_2(x)$, and $d_3(x)$ are mapped to $3x + 1$,
- $q_1(x)$, $q_2(x)$, and $q_3(x)$ are mapped to $3x^2 + 3x + 1$.

It remains to show that the polynomial order $\succ$ induced by this polynomial interpretation satisfies $g \succ d$ for all rules $g \to d$ of $R^{\mathcal{M}} \cup R_1 \cup R_2 \cup R_3$. First, we consider $R^{\mathcal{M}}$:

- for the rule $\hat{q}(x, y) \to \#$, the left-hand side is mapped to $x + y + 3$, and the right-hand side to 3, which is smaller that $x + y + 3$ for all instantiations of x, y with numbers > 0,

- for rules of the form $a(\hat{q}(x,y)) \to \hat{q}(x,y)$, the left-hand side is mapped to $x+y+5$, and the right-hand side to $x+y+3$,
- for all rules of $R^{\mathcal{M}}$ of the form $q(a(x), f(y)) \to a'(q'(x,y))$, the left-hand side is mapped to $(x+2)+(y+2)+3 = x+y+7$, and the right-hand side is mapped to $(x+y+3)+2 = x+y+5$,
- for all rules of $R^{\mathcal{M}}$ of the form $c(q(a(x), f(y))) \to q'(ca'(x), y)$, the left-hand side is mapped to $((x+2)+(y+2)+3)+2 = x+y+9$, and the right-hand side is mapped to $((x+2)+2)+y+3 = x+y+7$.

Next, we consider R_1:

- for the rule $q_0(y_1,y_2) +_1 q_0(z_1,z_2) \to q_0(y_1,y_2)$ of R_1, the left-hand side is mapped to $y_1+y_2+3+2(z_1+z_2+3)+1 = y_1+y_2+2z_1+2z_2+10$, and the right-hand side to y_1+y_2+3,
- for the rule $b(y) +_1 q_0(z_1,z_2) \to b(y)$ of R_1, the left-hand side is mapped to $y+2+2(z_1+z_2+3)+1 = y+2z_1+2z_2+9$, and the right-hand side to $y+2$,
- for the rule $x +_1 b(y) \to b(x +_1 y)$ of R_1, the left-hand side is mapped to $x+2(y+2)+1 = x+2y+5$, and the right-hand side to $(x+2y+1)+2 = x+2y+3$,
- for the rule $d_1(q_0(z_1,z_2)) \to q_0(z_1,z_2)$ of R_1, the left-hand side is mapped to $3(z_1+z_2+3)+1 = 3z_1+3z_2+10$, and the right-hand side to z_1+z_2+3,
- for the rule $d_1(b(x)) \to b(b(d_1(x)))$ of R_1, the left-hand side is mapped to $3(x+2)+1 = 3x+7$, and the right-hand side to $(3x+1+2)+2 = 3x+5$,
- for the rule $q_1(q_0(z_1,z_2)) \to q_0(z_1,z_2)$ of R_1, the left-hand side is mapped to $3(z_1+z_2+3)^2+3(z_1+z_2+3)+1$, and the right-hand side to z_1+z_2+3,
- for the rule $q_1(b(x)) \to q_1(x) +_1 b(d_1(x))$ of R_1, the left-hand side is mapped to $3(x+2)^2+3(x+2)+1 = 3x^2+15x+19$, and the right-hand side to $3x^2+3x+1+2(3x+1+2)+1 = 3x^2+9x+8$, as in Example 2.

The rules of R_2 and R_3 can be treated in a similar way. ◀

Combining the results obtained so far in this section, we thus have determined the exact complexity of the small term reachability problem for the class of TRSs considered here.

▶ **Theorem 18.** *The small term reachability problem for TRSs whose termination can be shown with a polynomial order is N2ExpTime-complete.*

In the setting considered in this section, restricting the attention to confluent TRSs does not reduce the complexity. Regarding the upper bound, the argument used in the proof of Proposition 7 does not apply since it is no longer the case that normal forms are of smallest size. Thus, one cannot reduce the complexity from N2ExpTime to 2ExpTime by only looking at a single rewrite sequence that ends in a normal form. However, our N2ExpTime-hardness proof does not directly work for confluent TRSs whose termination can be shown with a polynomial order. The reason is that, for a given nondeterministic Turing machine $\mathcal{M}$, the rewrite system $R^{\mathcal{M}} \cup R_1 \cup R_2 \cup R_3$ need not be confluent. In fact, for a given input word, there may be terminating runs of the TM that reach the accepting state $\hat{q}$, but also ones that do not reach this state. Using the former runs, our rewrite system can then generate the term $\#$, whereas this is not possible if we use one of the latter runs.

We can, however, modify the system $R^{\mathcal{M}} \cup R_1 \cup R_2 \cup R_3$ such that it becomes confluent. To this end, we introduce two new function symbols $\#_1$ and $\#_0$ of arity 1 and 0, respectively. Moreover, we add the following rules R_c:

$$\begin{array}{rcll} g(x_1,\dots,x_n) & \to & \#_1(\#_0) & \text{for all function symbols } g \text{ of arity} > 0 \text{ except } \#_1, \\ \#_1(\#_1(\#_0)) & \to & \#_1(\#_0), & \\ \# & \to & \#_1(\#_0). & \end{array}$$

Clearly, $R^{\mathcal{M}} \cup R_1 \cup R_2 \cup R_3 \cup R_c$ is confluent, because any term that is not in normal form (i.e., any term except variables, $\#_0$, $\#_1(\#_0)$, and terms of the form $\#_1(x)$ for variables x) has the only normal form $\#_1(\#_0)$ of size two. (This is the reason why we could not use a rule like $x +_1 q_0(z_1, z_2) \to x$ in R_1, because then $x +_1 q_0(z_1, z_2)$ would have the two normal forms x and $\#_1(\#_0)$.) However, the term $\#$ of size one is still only reachable from $t(w)$ if the final state of the TM is reached by a simulation of an accepting computation of $\mathcal{M}$. We extend the polynomial interpretation in the proof of Lemma 9 as follows:

- $\#_1(x)$ is mapped to $x + 1$,
- $\#_0$ is mapped to 1.

Then the polynomial order induced by this polynomial interpretation also orients the rules of R_c from left to right, i.e., termination of the resulting system can still be shown using a polynomial order.

▶ **Corollary 19.** *For confluent TRSs whose termination can be shown with a polynomial order, the small term reachability problem is N2ExpTime-complete.*

As shown in [16], if termination of a TRS can be shown with a *linear* polynomial order (i.e., where all polynomials have degree at most 1), then this implies an exponential bound on the lengths of reduction sequences. Again, this bound is tight and one can use the example showing this to obtain a TRS that generates an exponentially large tape and an exponentially large counter, similarly to what we have done in the general case.

▶ **Example 20.** Let R_d consist of just the two d-rules from Example 12. Then the term $d^\ell(s(0))$ can be reduced to $s^{2^\ell}(0)$.

▶ **Corollary 21.** *The small term reachability problem is NExpTime-complete for TRSs whose termination can be shown with a linear polynomial order. NExpTime-hardness already holds if only confluent systems are considered.*

Proof. The upper bound can be shown as before, i.e., one just needs to guess a reduction sequence (of exponential length) and then check the size of the obtained term.

For the lower bound, we proceed as in the proof of N2ExpTime-hardness for the case of general polynomial orders. Thus, we assume that $\mathcal{M}$ is an exponentially time bounded nondeterministic TM whose time-bound is $2^{p(\ell)}$ for a polynomial p, where ℓ is the length of the input word. Given such an input word $w = a_1 \ldots a_\ell$ for $\mathcal{M}$, we now construct the term

$$t'(w) = d_1^{p(\ell)}(b(q_0(a_1 \ldots a_\ell d_2^{p(\ell)}(b(\#)), d_3^{p(\ell)}(f(\#))))).$$

Instead of R_1, R_2, R_3, we now only need their rules for d_1, d_2, and d_3; let R'_d denote this system of 6 rules. As above, we can show that the term $t'(w)$ can be rewritten with $R^{\mathcal{M}} \cup R'_d$ to a term of size 1 iff $\mathcal{M}$ accepts the word w. Moreover, termination of $R^{\mathcal{M}} \cup R'_d$ can be proved by the linear polynomial order obtained from the one in the proof of Lemma 17 by removing the (non-linear) interpretations of q_1, q_2, q_3.

Similarly to the proof of Corollary 19, we can prove that NExpTime-hardness also holds for *confluent* TRSs whose termination can be shown with a linear polynomial order. The reason is that termination of the modified confluent TRS $R^{\mathcal{M}} \cup R'_d \cup R_c$ can be shown by the linear polynomial order that results from the one employed in the proof of Corollary 19 by removing the (non-linear) interpretations of q_1, q_2, q_3. ◀

6 Term rewriting systems shown terminating with a Knuth-Bendix order without special symbol

Without any restriction, there is no primitive recursive bound on the length of derivation chains for TRSs whose termination can be shown using a Knuth-Bendix order [16], but a uniform multiple recursive upper bound is shown in [15]. Here, we restrict the attention to KBOs without a *special symbol*, i.e., without a unary symbol of weight zero. For such KBOs, an exponential upper bound on the derivation length was shown in [16].[1] Given the results proven in the previous section, one could now conjecture that in this case the small term reachability problem is NExpTime-complete. However, we will show below that the complexity is actually only PSpace. In fact, the TRSs yielding the lower bounds for the derivation length considered in the previous section have not only long reduction chains (of double-exponential or exponential length), but are also able to produce large terms (of double-exponential or exponential size). For KBOs without special symbol, this is not the case. The following lemma provides us with a linear bound on the sizes of reachable terms. It will allow us to show a *PSpace upper bound* for the small term reachability problem.

▶ **Lemma 22.** *Let R be a TRS whose termination can be shown using a KBO without special symbol, and s_0, s_1 terms such that $s_0 \xrightarrow{*}_R s_1$. Then the size of s_1 is linearly bounded by the size of s_0, i.e., there is a constant c such that that $|s_1| \leq c{\cdot}|s_0|$ whenever $s_0 \xrightarrow{*}_R s_1$.*

Proof. Fix a KBO with weight function w showing termination of R such that all symbols of arity 1 have weight > 0. Let w_{min} be the minimal weight > 0 of a function symbol occurring in R or a variable,[2] i.e.,

$$w_{min} := \min\{w(f) \mid w(f) > 0 \text{ and } f \text{ is a function symbol in } R \text{ or a variable}\},$$

and let w_{max} be the maximal weight of a function symbol in R or a variable. As the weights of function symbols not occurring in R have no influence on the orientation of the rules in R with the given KBO, we can assume without loss of generality that their weight is w_{min}.

Let t be a term and $n_i(t)$ for $i = 0, \dots, k$ the number of occurrences of symbols of arity i in t, where k is the maximal arity of a symbol occurring in t.[3] Note that $|t| = n_0(t) + n_1(t) + \ldots + n_k(t)$. The following fact, which can easily be shown by induction on the structure of t, is stated in [21]:

$$n_0(t) + n_1(t) + \ldots + n_k(t) = 1 + 1{\cdot}n_1(t) + 2{\cdot}n_2(t) + \ldots + k{\cdot}n_k(t).$$

In particular, this implies that $n_0(t) \geq n_2(t) + \ldots + n_k(t)$. Since symbols of arity 0 and 1 have weights > 0, we know that

$$w(t) \geq w_{min}{\cdot}(n_0(t) + n_1(t)) \geq w_{min}{\cdot}n_0(t) \geq w_{min}{\cdot}(n_2(t) + \ldots + n_k(t)).$$

Consequently, $2{\cdot}w_{min}^{-1}{\cdot}w(t) \geq n_0(t) + n_1(t) + \ldots + n_k(t) = |t|$. This shows that the size of a term is linearly bounded by its weight. Conversely, it is easy to see that the weight of a term is linearly bounded by its size: $w(t) \leq w_{max}{\cdot}|t|$.

Now, assume that $s_0 \xrightarrow{*}_R s_1$. Since termination of R is shown with our given KBO, we know that $w(s_0) \geq w(s_1)$, and thus $w_{max}{\cdot}|s_0| \geq w(s_1) \geq 1/2{\cdot}w_{min}{\cdot}|s_1|$. This yields $|s_1| \leq 2{\cdot}w_{min}^{-1}{\cdot}w_{max}{\cdot}|s_0|$. ◀

[1] Actually, this result was shown in [16] only for KBOs using weights in $\mathbb{N}$, but it also holds for KBOs with non-negative weights in $\mathbb{R}$. This is an easy consequence of our Lemma 22.

[2] Recall that all variables have the same weight $w_0 > 0$.

[3] Variables have arity 0.

In particular, this means that the terms encountered during a rewriting sequence starting with a term s can each be stored using only polynomial space in the size of s. Given that the length of such a sequence is exponentially bounded, we can decide the small term reachability problem by the following NPSpace algorithm:

- guess a rewrite sequence $s \to_R s_1 \to_R s_2 \to_R \dots$ and always store only the current term;
- in each step, check whether $|s_i| \leq n$ holds. If the answer is "yes" then stop and accept. Otherwise, guess the next rewriting step; if this is not possible since s_i is irreducible, then stop and reject.

This algorithm needs only polynomial space since, by Lemma 22, the size of each term s_i is linearly bounded by the size of s. It always terminates since R is terminating. If there is a term of size $\leq n$ reachable from s, then the algorithm is able to guess the sequence leading to it, and thus it has an accepting run. Otherwise, all runs are terminating and rejecting. Since, by Savitch's theorem [27], NPSpace = PSpace, we obtain the following complexity upper bound.

▶ **Lemma 23.** *The small term reachability problem is in PSpace for TRSs whose termination can be shown with a KBO without special symbol.*

It remains to prove the corresponding *lower bound.* Let $\mathcal{M}$ be a polynomial space bounded TM, and p the polynomial that yields the space bound. Then there is a polynomial q such that any run of $\mathcal{M}$ longer than $2^{q(\ell)}$ on an input word w of length ℓ is cyclic. Thus, to check whether $\mathcal{M}$ accepts w, it is sufficient to consider only runs of length at most $2^{q(\ell)}$. However, in contrast to the reduction used in the previous section, we cannot generate an exponentially large unary down counter using a TRS whose termination can be shown with a KBO without special symbol. Instead, we use a polynomially large *binary* down counter that is decremented, starting with the binary representation $10^{q(\ell)}$ of $2^{q(\ell)}$ (see Example 3). For example, if $q(\ell) = 3$, then we represent the number $2^{q(\ell)} = 2^3 = 8$ as the binary number $10^{q(\ell)} = 1000$. The construction of the TRS $R^{\mathcal{M}}_{bin}$ simulating $\mathcal{M}$ given below is very similar to the construction given in the proof of Lemma 7 in [10].

As signature for $R^{\mathcal{M}}_{bin}$ we again use the tape symbols of $\mathcal{M}$ as unary function symbols, but now also the states are treated as unary symbols. In addition, we need the unary function symbols 0 and 1 to represent the counter, as well as primed versions $a', q', 1'$ of the tape symbols a, the states q, and the symbol 1. For a given input word $w = a_1 \dots a_\ell$ of $\mathcal{M}$, we now construct a term that starts with the binary representation of $2^{q(\ell)}$ and is followed by enough tape space for a $p(\ell)$ space bounded TM to work on:

$$t(w) := 10^{q(\ell)}(b^{p(\ell)}(q_0(a_1 \dots a_\ell(b^{p(\ell)-\ell}(\#))))).$$

Clearly, $t(w)$ can be constructed in polynomial time.

The TRS $R^{\mathcal{M}}_{bin}$ is now constructed as follows. The first part decrements the counter (as in Example 3) and by doing so "sends a prime" to the right:

$$\begin{array}{lll} 1(a(x)) \to 0(a'(x)) \quad \text{and} & 0(a(x)) \to 1'(a'(x)) & \text{for all tape symbols } a, \\ 0(1'(x)) \to 1'(1(x)), & 1(1'(x)) \to 0(1(x)). & \end{array}$$

The prime can go to the right on the tape until it reaches a state, which it then turns into its primed version:

$$a'g(x) \to ag'(x) \qquad \text{for tape symbols } a \text{ and tape symbols or states } g.$$

Only primed states can perform a transition of the TM:

$q_1'(a_1(x)) \to a_2(q_2(x))$	for each transition (q_1, a_1, q_2, a_2, r) of $\mathcal{M}$,
$c(q_1'(a_1(x)) \to q_2(c(a_2(x)))$	for each transition (q_1, a_1, q_2, a_2, l) of $\mathcal{M}$ and tape symbol c.

Again, the blank symbol b is also considered as a tape symbol of $\mathcal{M}$. Note that the rôle of the counter is not to restrict the number of transition steps simulated by $R^{\mathcal{M}}_{bin}$. Instead it produces enough primes to allow the simulation of at least $2^{q(\ell)}$ steps, while termination can still be shown using a KBO without special symbol.

Once the unique final accepting state $\hat{q}$ is reached, we remove all symbols other than #:

$a(\hat{q}(x)) \to \hat{q}(x)$	where a is a tape symbol or 0 or 1,
$\hat{q}(x) \to \#.$	

▶ **Lemma 24.** *The term $t(w)$ can be rewritten with $R^{\mathcal{M}}_{bin}$ to a term of size 1 iff $\mathcal{M}$ accepts the word w.*

Proof. If $\mathcal{M}$ accepts the word w, then there is a run of $\mathcal{M}$ on input w that ends in the state $\hat{q}$, uses at most $p(\ell)$ space, and requires at most $2^{q(\ell)}$ steps. This run can be simulated by $R^{\mathcal{M}}_{bin}$ by decrementing the counter, sending a prime to the state, applying a transition, decrementing the counter, etc. Since the counter can be decremented $2^{q(\ell)}$ times, we can use this approach to simulate a run of length at most $2^{q(\ell)}$. Once the accepting state is reached, we can use the last two rules to reach the term #, which has size 1.

Conversely, we can only reach a term of size one, if these cancellation rules are applied. This is only possible if first the accepting state has been reached by simulating an accepting run of $\mathcal{M}$. ◀

To conclude from this lemma that the small term reachability problem is PSpace-hard for TRSs whose termination can be shown using a KBO without special symbol, it is enough to show the following result.

▶ **Lemma 25.** *Termination of $R^{\mathcal{M}}_{bin}$ can be shown with a KBO without special symbol.*

Proof. It is easy to see that the KBO that assigns weight 1 to all function symbols and to all variables, and uses the precedence order $1 > 0 > 1'$ and $q' > a' > a > q$ for states q and tape symbols a, orients all rules of $R^{\mathcal{M}}_{bin}$ from left to right.[4] ◀

Combining the results obtained so far in this section, we thus have determined the exact complexity of the small term reachability problem for our class of TRSs.

▶ **Theorem 26.** *The small term reachability problem is PSpace-complete for TRSs whose termination can be shown with a KBO without special symbol.*

As in the case of the TRSs considered in the previous section, confluence does not reduce the complexity of the small term reachability problem for TRSs shown terminating with a KBO without special symbol. In fact, we can again extend the TRS $R^{\mathcal{M}}_{bin}$ such that it becomes confluent. To this purpose, we add two new function symbols $\#_1$ and $\#_0$ of respective arity 1 and 0, and two new rules:

$g(x) \to \#_1(\#_0)$	for all unary function symbols g different from $\#_1$,
$\# \to \#_1(\#_0).$	

[4] This KBO is similar to the one introduced in Example 10 of [10].

With this addition, every non-variable term built using the original signature of $R_{bin}^{\mathcal{M}}$ can be reduced to $\#_1(\#_0)$, which shows confluence. To show termination of the extended TRS, we modify and extend the KBO from the proof of Lemma 25 as follows. All function symbols in the original signature of $R_{bin}^{\mathcal{M}}$ (including $\#$) now get weight 2, and the symbols $\#_1$ and $\#_0$ as well as the variables get weight 1. The precedence order is extended by setting $g > \#_1$ for all function symbols g in the original signature of $R_{bin}^{\mathcal{M}}$. It is easy to see that the KBO defined this way shows that the extended TRS is terminating.

▶ **Corollary 27.** *For confluent TRSs whose termination can be shown with a KBO without special symbol, the small term reachability problem is PSpace-complete.*

7 Conclusion

The results of this paper show that the complexity of the small term reachability problem is closely related to the derivational complexity of the class of term rewriting systems considered. Interestingly, restricting the attention to confluent TRSs reduces the complexity only for the class of length-reducing systems, but not for the other two classes considered in this paper. The investigations in this paper were restricted to classes of TRSs defined by reduction orders (restricted form of KBO and polynomial orders) that yield relatively low bounds on the derivational complexity of the TRS. The derivational complexity of TRSs shown terminating by KBOs with a unary function symbol of weight zero or by recursive path orders is much higher [14, 15, 23, 24, 29]. From a theoretical point of view, it would be interesting to see whether using such reduction orders or other more powerful techniques [13] for showing termination also results in a very high complexity of the small term reachability problem. In fact, as we have seen in this paper, the complexity of this problem not only depends on the length of reduction sequences, but also on whether one can use long sequences to generate large terms.

On the practical side, up to now we have only used length-reducing rules to shorten DL proofs. Basically, these rules are generated by finding frequent proof patterns (currently by hand) and replacing them by a new "macro rule". The results of Section 4 show that, in this case, confluence of the rewrite system is helpful. When translating between different proof calculi, length-reducing systems will probably not be sufficient. Therefore, we will investigate with what kinds of techniques proof rewriting systems (e.g., translating between different proof calculi for $\mathcal{EL}$) can be shown terminating. Are polynomial orders or KBOs without unary function symbol of weight zero sufficient, or are more powerful approaches for showing termination needed? In this context, it might also be interesting to consider rewriting modulo equational theories [8, 18] and associated approaches for showing termination [1, 11, 19, 26]. For example, it makes sense not to distinguish between proof steps that differ only in the order of the prerequisites. Hence, rewriting such proofs could be represented via term rewriting modulo associativity and commutativity.

References

1 Beatriz Alarcón, Salvador Lucas, and José Meseguer. A dependency pair framework for $A \vee C$-termination. In Peter Csaba Ölveczky, editor, *Rewriting Logic and Its Applications - 8th International Workshop, WRLA 2010, Revised Selected Papers*, volume 6381 of *Lecture Notes in Computer Science*, pages 35–51. Springer, 2010. doi:10.1007/978-3-642-16310-4_4.

2 Christian Alrabbaa, Franz Baader, Stefan Borgwardt, Raimund Dachselt, Patrick Koopmann, and Julián Méndez. Evonne: Interactive proof visualization for description logics (system description). In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Proceedings*, volume 13385 of *Lecture Notes in Computer Science*, pages 271–280. Springer, 2022. doi:10.1007/978-3-031-10769-6_16.

3 Christian Alrabbaa, Franz Baader, Stefan Borgwardt, Patrick Koopmann, and Alisa Kovtunova. Finding small proofs for description logic entailments: Theory and practice. In Elvira Albert and Laura Kovács, editors, *LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Proceedings*, volume 73 of *EPiC Series in Computing*, pages 32–67. EasyChair, 2020. `doi:10.29007/NHPP`.

4 Christian Alrabbaa, Franz Baader, Stefan Borgwardt, Patrick Koopmann, and Alisa Kovtunova. Finding good proofs for description logic entailments using recursive quality measures. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Proceedings*, volume 12699 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2021. `doi:10.1007/978-3-030-79876-5_17`.

5 Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 364–369. Professional Book Center, 2005. URL: `http://ijcai.org/Proceedings/05/Papers/0372.pdf`.

6 Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

7 Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

8 Leo Bachmair and Nachum Dershowitz. Completion for rewriting modulo a congruence. *Theor. Comput. Sci.*, 67(2&3):173–201, 1989. `doi:10.1016/0304-3975(89)90003-0`.

9 Guillaume Bonfante, Adam Cichon, Jean-Yves Marion, and Hélène Touzet. Algorithms with polynomial interpretation termination proof. *J. Funct. Program.*, 11(1):33–53, 2001. `doi:10.1017/S0956796800003877`.

10 Guillaume Bonfante and Georg Moser. Characterising space complexity classes via Knuth-Bendix orders. In Christian G. Fermüller and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Proceedings*, volume 6397 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2010. `doi:10.1007/978-3-642-16242-8_11`.

11 Jürgen Giesl and Deepak Kapur. Dependency pairs for equational rewriting. In Aart Middeldorp, editor, *Rewriting Techniques and Applications, 12th International Conference, RTA 2001, Proceedings*, volume 2051 of *Lecture Notes in Computer Science*, pages 93–108. Springer, 2001. `doi:10.1007/3-540-45127-7_9`.

12 Jürgen Giesl, Matthias Raffelsieper, Peter Schneider-Kamp, Stephan Swiderski, and René Thiemann. Automated termination proofs for Haskell by term rewriting. *ACM Trans. Program. Lang. Syst.*, 33(2):7:1–7:39, 2011. `doi:10.1145/1890028.1890030`.

13 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Mechanizing and improving dependency pairs. *J. Autom. Reason.*, 37(3):155–203, 2006. `doi:10.1007/S10817-006-9057-7`.

14 Dieter Hofbauer. Termination proofs by multiset path orderings imply primitive recursive derivation lengths. *Theor. Comput. Sci.*, 105(1):129–140, 1992. `doi:10.1016/0304-3975(92)90289-R`.

15 Dieter Hofbauer. An upper bound on the derivational complexity of Knuth-Bendix orderings. *Inf. Comput.*, 183(1):43–56, 2003. `doi:10.1016/S0890-5401(03)00008-7`.

16 Dieter Hofbauer and Clemens Lautemann. Termination proofs and the length of derivations (preliminary version). In Nachum Dershowitz, editor, *Rewriting Techniques and Applications, 3rd International Conference, RTA-89, Proceedings*, volume 355 of *Lecture Notes in Computer Science*, pages 167–177. Springer, 1989. `doi:10.1007/3-540-51081-8_107`.

17 Gérard Huet and Dallas S. Lankford. On the uniform halting problem for term rewriting systems. INRIA Rapport de Recherche No. 283, 1978. URL: `https://www.ens-lyon.fr/LIP/REWRITING/TERMINATION/Huet_Lankford.pdf`.

18 Jean-Pierre Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM J. Comput.*, 15(4):1155–1194, 1986. `doi:10.1137/0215084`.

19 Jean-Pierre Jouannaud and Claude Marché. Termination and completion modulo associativity, commutativity and identity. *Theor. Comput. Sci.*, 104(1):29–51, 1992. doi:10.1016/0304-3975(92)90165-C.

20 Yevgeny Kazakov, Markus Krötzsch, and Frantisek Simancik. The incredible ELK - from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. *J. Autom. Reason.*, 53(1):1–61, 2014. doi:10.1007/S10817-013-9296-3.

21 Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra.* Pergamon Press, Oxford, 1970.

22 Dallas S. Lankford. On proving term rewriting systems are Noetherian. Memo MTP-3, Math. Dept., Louisiana Technical University, Ruston, LA, 1979. URL: http://www.ens-lyon.fr/LIP/REWRITING/TERMINATION/Lankford_Poly_Term.pdf.

23 Ingo Lepper. Derivation lengths and order types of Knuth-Bendix orders. *Theor. Comput. Sci.*, 269(1-2):433–450, 2001. doi:10.1016/S0304-3975(01)00015-9.

24 Ingo Lepper. Simply terminating rewrite systems with long derivations. *Arch. Math. Log.*, 43(1):1–18, 2004. doi:10.1007/S00153-003-0190-2.

25 Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning, Vol. I*, pages 371–443. Elsevier and MIT Press, 2001. doi:10.1016/B978-044450813-3/50009-6.

26 Albert Rubio. A fully syntactic AC-RPO. *Inf. Comput.*, 178(2):515–533, 2002. doi:10.1006/INCO.2002.3158.

27 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

28 TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 2003.

29 Andreas Weiermann. Termination proofs for term rewriting systems by lexicographic path orderings imply multiply recursive derivation lengths. *Theor. Comput. Sci.*, 139(1&2):355–362, 1995. doi:10.1016/0304-3975(94)00135-6.