Ground Reachability and Joinability in Linear Term Rewriting Systems are Fixed Parameter Tractable with Respect to Depth

Mateus de Oliveira Oliveira*

Department of Informatics, University of Bergen, Norway
mateus.oliveira@uib.no

Abstract

The ground term reachability problem consists in determining whether a given variable-free term t can be transformed into a given variable-free term t' by the application of rules from a term rewriting system $\mathfrak{R}$. The joinability problem, on the other hand, consists in determining whether there exists a variable-free term t'' which is reachable both from t and from t'. Both problems have proven to be of fundamental importance for several subfields of computer science. Nevertheless, these problems are undecidable even when restricted to linear term rewriting systems. In this work, we approach reachability and joinability in linear term rewriting systems from the perspective of parameterized complexity theory, and show that these problems are fixed parameter tractable with respect to the depth of derivations. More precisely, we consider a notion of parallel rewriting, in which an unbounded number of rules can be applied simultaneously to a term as long as these rules do not interfere with each other. A term t_1 can reach a term t_2 in depth d if t_2 can be obtained from t_1 by the application of d parallel rewriting steps. Our main result states that for some function $f(\mathfrak{R}, d)$, and for any linear term rewriting system $\mathfrak{R}$, one can determine in time $f(\mathfrak{R}, d) \cdot |t_1| \cdot |t_2|$ whether a ground term t_2 can be reached from a ground term t_1 in depth at most d by the application of rules from $\mathfrak{R}$. Additionally, one can determine in time $f(\mathfrak{R}, d)^2 \cdot |t_1| \cdot |t_2|$ whether there exists a ground term u, such that u can be reached from both t_1 and t_2 in depth at most d. Our algorithms improve exponentially on exhaustive search, which terminates in time $2^{|t_1| \cdot 2^{O(d)}} \cdot |t_2|$, and can be applied with regard to any linear term rewriting system, irrespective of whether the rewriting system in question is terminating or confluent.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems, F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Linear Term Rewriting Systems, Ground Reachability, Ground Joinability, Fixed Parameter Tractability

Digital Object Identifier 10.4230/LIPIcs.IPEC.2016.25

1 Introduction

Term rewriting systems have played a major role in several fields of computer science, such as, functional programming languages, specification of abstract data types, symbolic computation and automated theorem proving [16, 1, 2]. Many practical and theoretical aspects of the theory of term rewriting system revolve around two fundamental problems: ground reachability, and ground joinability. In the former, given a rewriting system $\mathfrak{R}$ and

* The author is currently supported by the Bergen Research Foundation. This work was concluded while the author was at the Czech Academy of Sciences, supported by the European Research Council (grant number 339691).

two ground terms t and t' one is asked to determine whether t' can be reached from t by the application of a sequence of rewriting rules from $\mathfrak{R}$. In the latter, joinability, one is asked whether there is a ground term u which can be reached from both t and t'. Both problems are known to be decidable for several restricted classes of rewriting systems, such as ground rewriting systems, right-ground systems, shallow right-linear systems, and left-linear growing systems [13, 14, 15, 18]. On the other hand, ground reachability and joinability on linear term rewriting systems are known to be undecidable if no other restriction is imposed [20]. Indeed, it can be shown that for each Turing machine M, there exists a linear term rewriting system $\mathfrak{R}_M$ such that the halting problem for M can be reduced to ground reachability (joinability) in $\mathfrak{R}_M$ [20].

A rewriting system $\mathfrak{R}$ is said to be linear if it contains only rules of the form $l \to r$ where $var(r) \subseteq var(l)$ and each variable occurs at most once in l, and at most once in r. For instance, the associativity rule $x \cdot (y \cdot z) \to (x \cdot y) \cdot z$ is linear. In this work, we show that despite the undecidability of ground state reachability and joinability for linear term rewriting systems, both problems are fixed parameter tractable with respect to the depth of derivations. In this context, we consider a notion of parallel rewriting in which an unbounded number of rules can be applied simultaneously to a term, as long as these rules do not interfere with each other [19, 4]. Such a simultaneous application of independent rewriting rules is known in term-rewriting theory literature as *multi-step*. We say that a term t can reach a term t' in depth at most d if t' can be obtained from t by the application of d multi-steps.

▶ **Theorem 1** (Main Theorem). *There is a function $f(\mathfrak{R}, d)$ such that for any set of linear term rewriting rules $\mathfrak{R}$, and any ground terms t and t' over Σ,*

- *one can determine in time $f(\mathfrak{R}, d) \cdot |t| \cdot |t'|$ whether t' can be reached from t in depth at most d.*
- *one can determine in time $f(\mathfrak{R}, d)^2 \cdot |t| \cdot |t'|$ whether there exists a ground term u such that u is reachable in depth at most d from both t and t'.*

Our algorithms improve substantially on the running time of exhaustive search. We note that given a term t of size $|t|$, there may be up to $2^{O(|t|)}$ possible ways of applying simultaneous rewriting rules to t. Additionally, if a term t' is obtained from t in depth d, then the size of t' may be as large as $|t| \cdot 2^{O(d)}$. Therefore, as many as $2^{|t| \cdot 2^{O(d)}}$ distinct terms may be derived from a term t in depth at most d. Indeed this upper bound is asymptotically tight and can be matched even in ground rewriting systems. Consider for instance, a ranked alphabet $\Sigma = \{g, a\}$ consisting of one binary function symbol g, one constant symbol a, and a term rewriting system $\mathfrak{R}$ consisting of a single rewriting rule $a \to g(a, a)$. Then for any term t over Σ, one can derive at least $2^{|t| \cdot 2^{\Omega(d)}}$ distinct terms from t in depth at most d. In other words, even when d is a constant, determining whether a term t' can be reached from a term t in depth at most d by exhaustive search takes time exponential in t in the worst case, while using our approach, this problem can be solved in time $f(\mathfrak{R}, d) \cdot |t| \cdot |t'|$. We also should note that it is straightforward to define infinite families of pairs of terms (t_n, t'_n) such that t'_n can be reached from t_n in a single multi-step, but which require the application of an unbounded number of sequential individual rewriting steps. For instance, consider the term $t_n = a_1 + b_1 + a_2 + b_2 + ... + a_n + b_n$ where $a_i = 1$, $b_i = 2$ and $+$ is an associative commutative binary function symbol. Then in one multi-step one can reach the term $t'_n = b_1 + a_1 + b_2 + a_2 + ... + b_n + a_n$, whereas one would need to use n individual rewriting steps to derive t' from t. Note that the larger the n, the larger is the number of individual rewriting rules necessary to reach t' from t, while a single multi-step is sufficient for any n.

2 Term Rewriting and Tree Automata

In this section we define standard notions from term rewriting systems and tree automata. Extensive treatments of term rewriting theory can be found in [1, 7] and on tree-automata theory can be found in [5, 12].

2.1 Terms

The set of natural numbers, excluding 0, is denoted by $\mathbb{N}$. We let $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. A *ranked alphabet* is a finite set Σ of function symbols together with an arity function $\mathfrak{a} : \Sigma \to \mathbb{N}_0$. Intuitively the arity $\mathfrak{a}(f)$ of a symbol $f \in \Sigma$ specifies the number of inputs of f. A function symbol of arity 0 is called a constant symbol. We let $\mathfrak{a}(\Sigma) = \max\{\mathfrak{a}(f) \mid f \in \Sigma\}$ be the maximum arity of a symbol in Σ. Let X be a finite set of variables and Σ be a ranked alphabet. The set $Ter(\Sigma \cup X)$ of all *terms* over $\Sigma \cup X$ is inductively defined as follows:

- If x is a variable in X then x is a term in $Ter(\Sigma \cup X)$
- if $f \in \Sigma$ and $t_1, ..., t_{\mathfrak{a}(f)}$ are terms in $Ter(\Sigma \cup X)$ then $f(t_1, t_2, ..., t_{\mathfrak{a}(f)})$ is a term in $Ter(\Sigma \cup X)$.

If f is a function symbol of arity 0 then we write simply f to denote the term $f()$. A position for a term t is a string over $\mathbb{N}$. The empty string is denoted by ε. The set of positions for a term t is inductively defined as follows.

- $Pos(t) = \{\varepsilon\}$ if $t \in X$.
- $Pos(f(t_1, ..., t_{\mathfrak{a}(f)})) = \{\varepsilon\} \cup \{i.p \mid 1 \leq i \leq \mathfrak{a}(f),\ p \in Pos(t_i)\}$

We note that if t is either a variable or a function symbol of arity 0, then $Pos(t) = \{\varepsilon\}$. We let $|t| = |Pos(t)|$ denote the *size* of the term t. The *subterm* $t|_p$ of t at position p is inductively defined as follows. At the base case, $t|_\varepsilon = t$. Now, if $t = f(t_1, t_2, ..., t_{\mathfrak{a}(f)})$, then for each $j \in \{1, ..., \mathfrak{a}(f)\}$ and each position $jp \in Pos(t)$, $t|_{jp} = t_j|_p$.

Let $t = f(t_1, ..., t_{\mathfrak{a}(f)})$ be a term in $Ter(\Sigma \cup X)$. We let $\boldsymbol{rs}(t) = f$ be the *root symbol* of t. If $t = x$ for a variable $x \in X$ then we set $\boldsymbol{rs}(t) = x$. For each $p \in Pos(t)$, we let $t(p) = \boldsymbol{rs}(t|_p)$ denote the root symbol of the subterm of t at position p.

We denote by $var(t)$ the set of variables occurring in t. A *ground term* is a term t such that $var(t) = \emptyset$. In other words, a term t is ground if it contains no variables. In some places we may write $t \in Ter(\Sigma)$ to indicate that t is a ground term.

A *substitution* is a function $\sigma : X \to Ter(\Sigma \cup X)$ mapping variables in X to terms in $Ter(\Sigma \cup X)$. If t is a term, and σ is a substitution, then we denote by t^σ the term that is obtained from t by replacing each variable $x \in X$ with the term $\sigma(x)$. If t and s are terms and p is a position in $Pos(t)$ then we denote by $t[s]_p$ the term that is obtained from t by replacing the subterm $t|_p$ with the term s.

2.2 Term Rewriting

A *rewriting rule* is a pair $l \to r$ where l and r are terms in $Ter(\Sigma \cup X)$ with $var(r) \subseteq var(l)$. We say that a rule $l \to r$ is *linear* if each variable occurs at most once in l and at most once in r. Note that this definition of linearity allows $var(l) \cap var(r) \neq \emptyset$. A term rewriting system is any finite set $\mathfrak{R}$ of rewriting rules. We say that $\mathfrak{R}$ is linear if each rewriting rule $l \to r$ in $\mathfrak{R}$ is linear.

Let t be a term in $Ter(\Sigma \cup X)$, p be a position in $Pos(t)$, and $l \to r$ be a rewriting rule in $\mathfrak{R}$. We say that $l \to r$ can be applied to t at position p if there is a substitution $\sigma : X \to Ter(\Sigma \cup X)$ such that $t|_p = l^\sigma$. In this case, we let $t' = t[r^\sigma]_p$ be the term that is obtained from t by the application of the rewriting rule $l \to r$ at position p. We write $t \to_{\mathfrak{R}} t'$

to denote that t' can be obtained from t by the application of some rewriting rule $l \to r \in \mathfrak{R}$ at some position p of t. We say that $\to_{\mathfrak{R}}$ is the relation induced by $\mathfrak{R}$ on $Ter(\Sigma \cup X)$. We let $\to^*_{\mathfrak{R}}$ be the transitive closure of $\to_{\mathfrak{R}}$. In other words, $t \to^*_{\mathfrak{R}} t'$ if and only if t' is obtained from t by the application of a finite number of rewriting rules from $\mathfrak{R}$.

2.3 Tree Automata

Let Q be a finite set of symbols of arity 0 called states. The elements of the set $Ter(\Sigma \cup Q)$ are called *configurations*. A *transition* is a rewriting rule of the form $f(q_1, ..., q_{\mathfrak{a}(f)}) \to q$ for some function symbol $f \in \Sigma$ and states $q_1, ..., q_{\mathfrak{a}(f)}, q \in Q$. A *(bottom-up non-deterministic) finite tree-automaton* over Σ is a tuple $\mathcal{A} = (Q, \Sigma, F, \Delta)$ where $F \subseteq Q$ is a set of final states and δ is a set of transitions. Note that Δ should be regarded as a term rewriting system acting on terms in $Ter(\Sigma \cup Q)$. We may write $\to_{\mathcal{A}}$ to denote the rewriting relation induced by the transitions Δ. Analogously, we may write $\to^*_{\mathcal{A}}$ to denote $\to^*_{\Delta}$. The tree language recognized by a state q in $\mathcal{A}$ is defined as

$$\mathcal{L}(\mathcal{A}, q) = \{t \in Ter(\Sigma) \mid t \to^*_{\mathcal{A}} q\}.$$

Intuitively, $\mathcal{L}(\mathcal{A}, q)$ is the set of all ground terms in $Ter(\Sigma)$ that can be reduced to the state q by the application of transitions (rewriting rules) in Δ. The tree language accepted by $\mathcal{A}$ is defined as $\mathcal{L}(\mathcal{A}) = \bigcup_{q \in F} \mathcal{L}(\mathcal{A}, q)$. As an abuse of notation we will often write $q \in \mathcal{A}$ and $t \to q \in \mathcal{A}$ to denote respectively that $q \in Q$ and $t \to q \in \Delta$.

The size of $\mathcal{A}$, which is defined as $|\mathcal{A}| = |Q| + |\Delta|$, measures the number of states in Q plus the number of transitions in Δ. If $f(q_1, ..., q_{\mathfrak{a}(f)}) \to q$ is a transition in Δ, then we say that q is the consequent of $f(q_1, ..., q_{\mathfrak{a}(f)}) \to q$, while each state in $\{q_1, ..., q_{\mathfrak{a}(f)}\}$ is an antecedent of $f(q_1, ..., q_{\mathfrak{a}(f)}) \to q$. We say that q is incident with a transition if it is either an antecedent or a consequent of the transition. The *in-degree* of a state q in Q, denoted by $\delta(q)$ is the number of transitions in Δ that have q as a consequent. The *maximum state in-degree* of $\mathcal{A}$, defined as $\delta(\mathcal{A}) = \max_{q \in Q} \delta(q)$, is the maximum in-degree of a state in $\mathcal{A}$. We say that $\mathcal{A}$ is reachable if for each state $q \in Q$ the language $\mathcal{L}(\mathcal{A}, q)$ is non-empty.

▶ **Lemma 2** (Membership [10]). *Let $\mathcal{A}$ be a tree automaton over Σ, and let t be a ground term in $Ter(\Sigma)$. One can determine in time $O(|t| \cdot |\mathcal{A}|)$ whether $t \in \mathcal{L}(\mathcal{A})$.*

▶ **Lemma 3** (Emptiness of Intersection [10]). *Let $\mathcal{A}$ and $\mathcal{A}'$ be two tree automata over Σ. One can determine in time $O(|\mathcal{A}| \cdot |\mathcal{A}'|)$ whether $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}') = \emptyset$.*

2.4 Simultaneous Rewriting via Multi-Steps

In this work we will be interested in a notion of rewriting that allows for the simultaneous application of several rules to a term as long as these rules do not interfere with each other. Such a notion of simultaneous rewriting can be formalized via the notion of *multi-step* [19]. A more detailed treatment rewriting by multi-steps can be found in [4] (Chapter 4). When restricted to the setting of rewriting on ground terms the notion of multi-step can be formalized as in Definition 4.

▶ **Definition 4** (Multi-Step). Let $\mathfrak{R}$ be a term rewriting system. The multi-step relation $\multimap \subseteq Ter(\Sigma) \times Ter(\Sigma)$ induced by $\mathfrak{R}$ is inductively defined as follows.

1. $f(t_1, ..., t_{\mathfrak{a}(f)}) \multimap f(t'_1, ..., t'_{\mathfrak{a}(f)})$ if $f \in \Sigma$ and $t_i \multimap t'_i$ for each $i \in \{1, ..., \mathfrak{a}(f)\}$.
2. $l^\sigma \multimap r^\theta$ if $l \to r \in \mathfrak{R}$, and $\sigma, \theta : X \to Ter(\Sigma)$ are substitutions such that $\sigma(x) \multimap \theta(x)$ for each variable $x \in var(l)$.

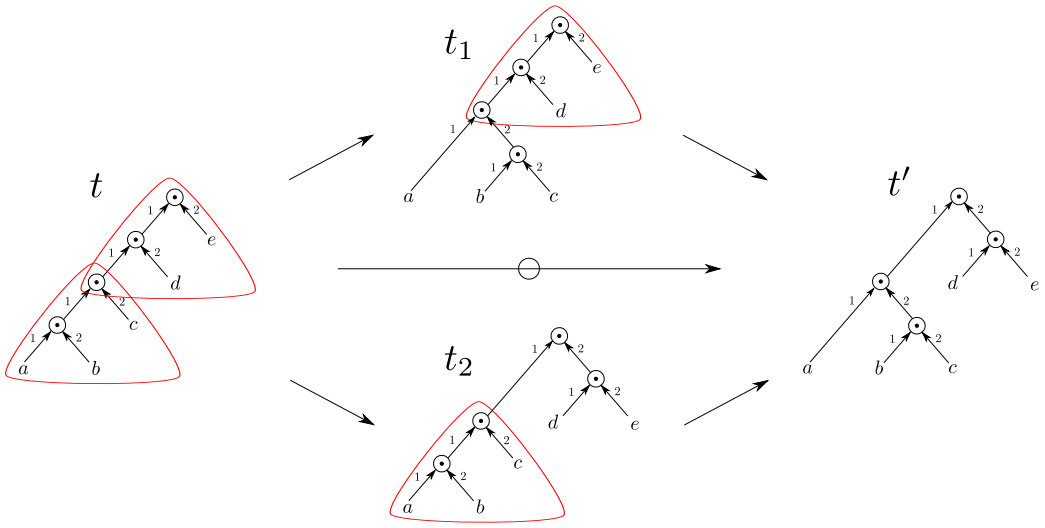

Figure 1 The application of a multi-step $t \multimap t'$ where $t = (((a \cdot b) \cdot c) \cdot d) \cdot e$ and $t' = (a \cdot (b \cdot c)) \cdot (d \cdot e)$. This multi-step $t \multimap t'$ intuitively corresponds to the simultaneous application of two instances of the associativity rule $(x \cdot y) \cdot z \to x \cdot (y \cdot z)$ to t. The regions surrounded by red curves indicate the portions of the term identifying a match for the left-hand side of the rule. The multistep can be decomposed in two ways as a sequence of individual rewriting rules: either as $t \to_{\mathfrak{R}} t_1 \to_{\mathfrak{R}} t'$ or as $t \to_{\mathfrak{R}} t_2 \to_{\mathfrak{R}} t'$.

Note that the base case of the inductive definition is embedded in Condition 1. More precisely, for each function symbol $f \in \Sigma$ of arity 0, we have that $f \multimap f$. We note that a multi-step $t \multimap t'$ may be decomposed in several different ways as sequences of applications of individual rewriting rules from $\mathfrak{R}$. For instance, in Figure 1 we depict the application of a multi-step to a term t. Intuitively, this multi-step $t \multimap t'$ corresponds to the simultaneous application of two instances of the associativity rule $(x \cdot y) \cdot z \to x \cdot (y \cdot z)$. There are two different ways of decomposing $t \multimap t'$ into sequences of individual rewriting rules: $t \to_{\mathfrak{R}} t_1 \to_{\mathfrak{R}} t'$ and $t \to_{\mathfrak{R}} t_2 \to_{\mathfrak{R}} t'$.

We say that a term t' is derived from a term t in depth at most d if there is a sequence of multi-steps $t_0 \multimap t_1 \multimap \ldots \multimap t_d$ such that $t_0 = t$ and $t_d = t'$. We write $t \overset{d}{\multimap} t'$ to denote that t' can be obtained from t in depth at most d.

3 Tree Automata Completion for Multi-Steps

Tree Automata completion is a powerful set of techniques which has found many applications in the field of termination analysis of rewriting systems [9, 11, 17, 8]. In this section we show that completion techniques, which have been so far used only in the context of sequential term rewriting, can be used to characterize derivability in one multi-step on linear term rewriting systems. More precisely, given a tree-automaton $\mathcal{A}$ and a linear term rewriting system $\mathfrak{R}$, we use a special instance of the tree-automata completion algorithm introduced in [9] in the context of sequential rewriting to construct a particular tree-automaton $\mathcal{N}(\mathcal{A}, \mathfrak{R})$. Subsequently, in Lemma 7 we show that the language accepted by $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ consists precisely of the set of terms that can be obtained from terms in $\mathcal{L}(\mathcal{A})$ by the application of one multi-step. A crucial aspect of our construction is that the size of the tree automaton $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ is upper bounded by $g(\mathfrak{R}, \delta) \cdot |\mathcal{A}|$ where $g(\mathfrak{R}, \delta)$ is a function that depends only on the term rewriting system $\mathfrak{R}$ and on the maximum state in-degree δ of $\mathcal{A}$. In other words,

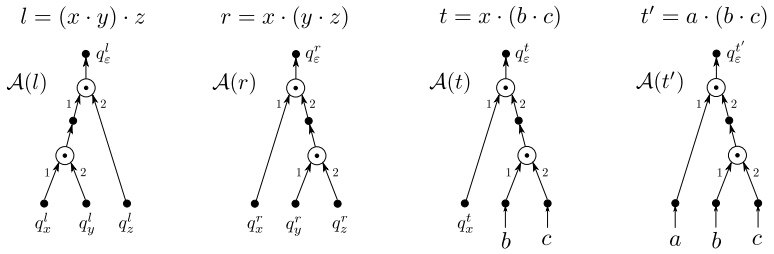

Figure 2 Graphical representation of the tree automata associated with terms l, r, t and t' respectively. The symbols x, y and z are variables. The symbol $\odot$ is a function symbol of arity 2, and a, b, c are constants (function symbols of arity 0). States are denoted by black dots.

the size of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ grows linearly with the size of $\mathcal{A}$. This linear growth will be crucial for our fixed parameter tractability results.

▶ **Definition 5** (Tree Automaton Associated with a Term)**.** Let t be a term in $Ter(\Sigma \cup X)$. The tree automaton associated with t, denoted by $\mathcal{A}(t) = (Q, \Sigma, F, \Delta)$ is defined as follows.

$$Q = \{q^t_p \mid p \in Pos(t)\} \qquad F = \{q^t_\varepsilon\}$$
$$\Delta = \{f(q^t_{p.1}, ..., q^t_{p.\mathtt{a}(f)}) \to q^t_p \mid t(p) = f\} \tag{1}$$

Intuitively, if t is a ground term, then $\mathcal{A}(t)$ is the 'simplest' (but not necessarily minimal) tree automaton that accepts t and no other term. On the other hand, if t has some variable then the language of $\mathcal{A}(t)$ is empty. Nevertheless, this is not relevant, since in this case these tree-automata will be glued to other tree-automata in order to define a meaningful tree-language. In Figure 2 we depict tree-automata associated with several terms with and without variables.

Let $\mathcal{A} = (Q, \Sigma, F, \Delta)$ be a tree automaton and let l be a term in $Ter(\Sigma \cup X)$. A *state-substitution* for l is a function $\gamma : var(l) \to Q$ that associates with each variable $x \in var(l)$ a state $\gamma(x) \in Q$. Note that the term l^γ obtained from l by replacing each variable x with the state $\gamma(x)$ is a configuration in $Ter(\Sigma \cup Q)$. Also note that if l is a ground term in $Ter(\Sigma)$, then the only state-substitution for l is the empty function $\gamma :\to Q$. In this case, $l^\gamma = l$. We say that a state-substitution $\gamma : var(l) \to Q$ is good for a pair (q, l) if $l^\gamma \to_\Delta q$. In other words, γ is good for (q, l) if the configuration l^γ can be reduced to state q by the application of transitions in Δ. We let $\mathcal{M}(\mathcal{A}, q, l)$ be the set good state-substitutions for (q, l).

Let $l \to r$ be a linear term rewriting rule over Σ. We let q^l_ε denote the unique accepting state of $\mathcal{A}(l)$, and q^r_ε denote the unique accepting state of $\mathcal{A}(r)$. Additionally, for each variable $x \in var(l) \cap var(r)$, we let q^l_x denote the unique state of $\mathcal{A}(l)$ corresponding to the variable x, and we let q^r_x denote the unique state of $\mathcal{A}(r)$ corresponding to the variable x. Now let $\mathcal{A}$ be a tree automaton over Σ, and $\gamma : var(l) \to Q$ be a state-substitution in $\mathcal{M}(\mathcal{A}, q, l)$. We denote by $\mathcal{C}(\mathcal{A}, q, l \to r, \gamma)$ the tree automaton which is obtained by creating a fresh copy of $\mathcal{A}(r)$, and by renaming the minimal and maximal states of $\mathcal{A}(r)$ as follows.

1. Rename the state q^r_ε in $\mathcal{A}(r)$ to the state q.
2. For each variable $x \in var(l) \cap var(r)$, rename the state q^r_x of $\mathcal{A}(r)$ to the state $\gamma(x)$.

Now consider the tree automaton $\mathcal{A} \cup \mathcal{C}(\mathcal{A}, q, l \to r, \gamma)$. Intuitively, this tree automaton is obtained by creating a copy of $\mathcal{A}(r)$ and subsequently, by identifying its accepting state q^r_ε with the state q of $\mathcal{A}$, and by identifying, for each variable $x \in var(l) \cap var(r)$, the state q^r_x with the state $\gamma(x)$. This process is illustrated in Figure 3.

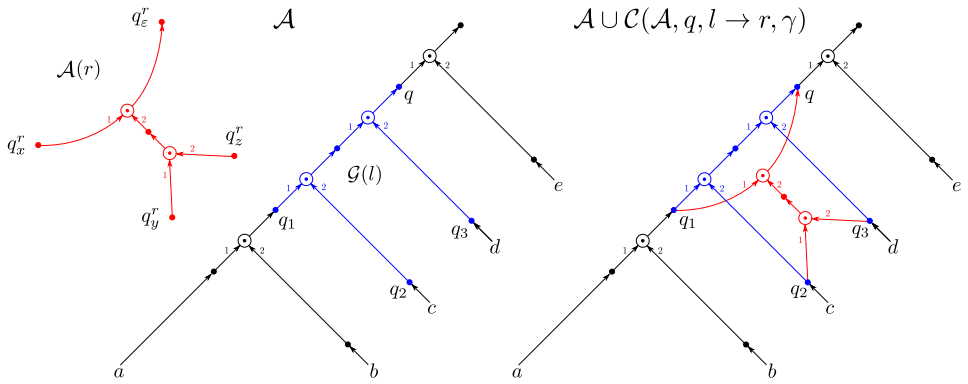

Figure 3 Let $l = (x \cdot y) \cdot z$ and $r = x \cdot (y \cdot z)$. Left: The disjoint union of a tree automaton $\mathcal{A}$ with the tree automaton $\mathcal{A}(r)$. The state-substitution γ maps the state q_x^l to q_1 the state q_y^l to q_2 and the state q_z^l to q_3. The portion of $\mathcal{A}$ in blue shows that the configuration l^γ can be reduced to the state q using transitions of $\mathcal{A}$. Right: The tree automaton $\mathcal{A} \cup \mathcal{C}(\mathcal{A}, \gamma, l \to r)$ obtained by identifying q_ε^r with q, q_x^r with q_1, q_y^r with q_2, and q_z^r with q_3. The tree automaton $\mathcal{C}(\mathcal{A}, q, l \to r, \gamma)$ is illustrated in red.

▶ **Definition 6** (Next Layer Operator). Let $\mathfrak{R}$ be a linear term rewriting system and $\mathcal{A}$ be a reachable tree automaton. The tree automaton $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ is defined as follows.

$$\mathcal{N}(\mathcal{A}, \mathfrak{R}) \quad = \quad \mathcal{A} \quad \cup \quad \bigcup_{\substack{q \in Q, l \to r \in \mathfrak{R} \\ \gamma \in \mathcal{M}(\mathcal{A}, q, l)}} \mathcal{C}(\mathcal{A}, q, l \to r, \gamma). \tag{2}$$

We say that each sub-automaton $\mathcal{C}(\mathcal{A}, q, l \to r, \gamma)$ of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ is a *right-component* of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$. Intuitively, $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ may be regarded as being constructed by adding one right component $\mathcal{C}(\mathcal{A}, q, l \to r, \gamma)$ to $\mathcal{A}$ at a time, in any desired order (See Figure 4).

The next lemma states that $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ accepts precisely those terms that can be reached from terms in $\mathcal{L}(\mathcal{A})$ by the application of one multi-step.

▶ **Lemma 7.** *Let $\mathfrak{R}$ be a linear term rewriting system, and let $\mathcal{A}$ be a tree automaton over Σ. The tree automaton $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ accepts the following language.*

$$\mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R})) = \{t' \mid \exists t \in \mathcal{L}(\mathcal{A}),\ t \multimap t'\}. \tag{3}$$

Proof. Let $\mathcal{A} = (Q, \Sigma, F, \Delta)$ and $\mathcal{N}(\mathcal{A}, \mathfrak{R}) = (Q \cup Q', \Sigma, F, \Delta \cup \Delta')$ where $Q \cap Q' = \emptyset$ and $\Delta \cap \Delta' = \emptyset$. Note that the final states of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ are the same as those of $\mathcal{A}$.

Completeness Proof

First we show that $\mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R})) \supseteq \{t' \mid \exists t \in \mathcal{L}(\mathcal{A}),\ t \multimap t'\}$. It is enough to show that for each state $q \in Q$ if $t \in \mathcal{L}(\mathcal{A}, q)$ and $t \multimap t'$ then $t' \in \mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R}), q)$. The proof of this claim is by induction on the structure of t, using Definition 4.

In the base case of Condition 1 of Definition 4, $t = a$ where a is a function symbol of arity 0 and $t' = a$. In this case the claim follows trivially. In the base case of Condition 2 of Definition 4, both t and t' are ground terms and $t \to t'$ belongs to $\mathfrak{R}$. Let $\gamma : \emptyset \to Q$ be the empty substitution. Since the term t' reaches the state q in $\mathcal{C}(\mathcal{A}, q, a \to t', \gamma)$ we have that $t' \in \mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R}), q)$.

For the inductive step of Condition 1, let $t = f(t_1, ..., t_{\mathfrak{a}(f)})$ and $t' = f(t'_1, ..., t'_{\mathfrak{a}(f)})$ where $t_i \multimap t'_i$ for each $i \in \{1, ..., \mathfrak{a}(f)\}$. Since $t \in \mathcal{L}(\mathcal{A}, q)$, there exists a transition

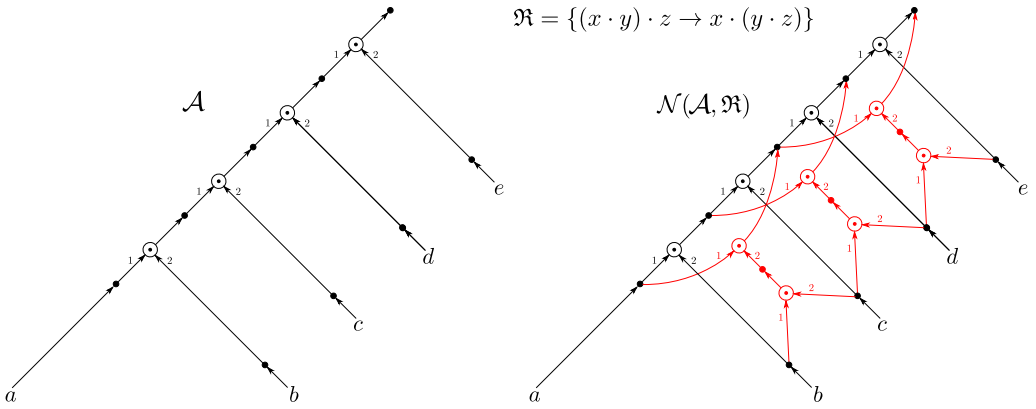

Figure 4 Left: A tree automaton $\mathcal{A}$ which recognizes the language $\mathcal{L}(\mathcal{A}) = \{(((a \cdot b) \cdot c) \cdot d) \cdot e\}$. Right: The tree automaton $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ which recognizes the language $\mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R})) = \{(((a \cdot b) \cdot c) \cdot d) \cdot e, ((a \cdot (b \cdot c)) \cdot d) \cdot e, ((a \cdot b) \cdot (c \cdot d)) \cdot e, ((a \cdot b) \cdot c) \cdot (d \cdot e), (a \cdot (b \cdot c)) \cdot (d \cdot e)\}$. The three right components of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ are depicted in red.

$(q_1, ..., q_{\mathfrak{a}(f)}, f) \to q$ in $\mathcal{A}$ such that t_i reaches q_i for each $i \in \{1, ..., \mathfrak{a}(f)\}$. By the induction hypothesis, since $t_i \multimap\!\!\to t'_i$, we have that $t'_i \in \mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R}), q_i)$. Therefore, $f(t'_1, ..., t'_{\mathfrak{a}(f)}) \in \mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R}), q)$.

For the inductive step of Condition 2, let $t = l^\sigma$ for some substitution $\sigma : X \to Ter(\Sigma)$, and let $t' = r^\theta$ where for each variable $x \in var(l)$, $\sigma(x) \multimap\!\!\to \theta(x)$. Since $t \in \mathcal{L}(\mathcal{A}, q)$, there exists a state substitution $\gamma : var(l) \to Q$ such that $l^\gamma \to^*_\Delta q$. By the induction hypothesis, $\theta(x) \in \mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R}), \gamma(x))$ for each $x \in var(l)$. Additionally, $r^\gamma \to^*_{\Delta'} q$ in the right component $\mathcal{C}(\mathcal{A}, q, l \to r, \gamma)$. This implies that $t' \in \mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R}), q)$.

Soundness Proof

Now we show that $\mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R})) \subseteq \{t' \mid \exists t \in \mathcal{L}(\mathcal{A}),\ t \multimap\!\!\to t'\}$. It is enough to show that for each state q of $\mathcal{A}$, if $t' \in \mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R}), q)$ then there is a ground term $t \in \mathcal{L}(\mathcal{A}, q)$ such that $t \multimap\!\!\to t'$. The proof is by well founded induction with terms ordered by the strict subterm relation.

Let $q \in Q$ and let $t \in \mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R}), q)$. In the base case, let $t' = a$ for some function symbol a of arity 0. Let $a \to q$ be a transition in $\mathcal{N}(\mathcal{A}, \mathfrak{R})$. If $a \to q$ belongs to Δ then $a \in \mathcal{L}(\mathcal{A}, q)$ and additionally, by the base case of Condition 1 of Definition 4, we have that $a \multimap\!\!\to a$. If $a \to q$ belongs to Δ', then there is some rewriting rule $l \to r$ in $\mathfrak{R}$ and some state-substitution $\gamma : var(l) \to Q$ such that $a \to q$ is a transition in $\mathcal{C}(\mathcal{A}, q, l \to r, \gamma)$. Let $\sigma : var(l) \to Ter(\Sigma)$ be a substitution that associates with each variable $x \in var(l)$ an arbitrary term in $\mathcal{L}(\mathcal{A}, \gamma(x))$. Note that such term is guaranteed to exist, since by assumption $\mathcal{A}$ is reachable. Then the term l^σ belongs to $\mathcal{L}(\mathcal{A}, q)$. Additionally, by Condition 2 of Definition 4, we have that $l^\sigma \multimap\!\!\to a^\tau$ where $\tau : \emptyset \to Ter(\Sigma)$ is the empty substitution. This verifies the claim in the base case.

Now let $t' = f(t'_1, ..., t'_{\mathfrak{a}(f)})$ where f has arity at least 1. Then there exists a transition $f(q_1, ..., q_{\mathfrak{a}(f)}) \to q$ in $\Delta \cup \Delta'$ such that $t'_i \in \mathcal{L}(\mathcal{N}(\mathcal{A}), q_i)$ for each $i \in \{1, ..., \mathfrak{a}(f)\}$. There are two cases to be analysed.

1. If $f(q_1, ..., q_{\mathfrak{a}(f)}) \to q$ belongs to Δ then all states $q_1, ..., q_{\mathfrak{a}(f)}$ belong to Q. In this case, by the induction hypothesis, there exists terms $t_1, ..., t_{\mathfrak{a}(f)}$ such that $t_i \multimap\!\!\to t'_i$ and $t_i \in \mathcal{L}(\mathcal{A}, q_i)$ for each $i \in \{1, ..., \mathfrak{a}(f)\}$. This implies that the term $t = f(t_1, ..., t_{\mathfrak{a}(f)})$ is in $\mathcal{L}(\mathcal{A}, q)$ and additionally, by Condition 1 of Definition 4, we have that $t \multimap\!\!\to t'$.

2. If $f(q_1, ..., q_{\mathfrak{a}(f)}) \to q$ belongs to Δ' then the states $q_1, ..., q_{\mathfrak{a}(f)}$ belong to some right component of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$. In other words, there is some rewriting rule $l \to r$ in $\mathfrak{R}$, some substitution $\theta : var(r) \to Ter(\Sigma)$ and some state-substitution $\gamma : var(l) \to Q$ such that $t' = r^\theta$, $l^\gamma \to^*_\Delta q$ and $r^\gamma \to^*_{\Delta'} q$ and such that $\theta(x) \in \mathcal{L}(\mathcal{N}(\mathcal{A}, \mathfrak{R}), \gamma(x))$ for each $x \in var(r)$. By the induction hypothesis, for each $x \in var(r) \subseteq var(l)$ there is a term s_x which belongs to $\mathcal{L}(\mathcal{A}, \gamma(x))$ and $s_x \multimap\!\!\to \theta(x)$. Additionally, since $\mathcal{A}$ is reachable, for each variable $y \in var(l) \backslash var(r)$ there is at least one term s_y in $\mathcal{L}(\mathcal{A}, \gamma(y))$. Let $\sigma : var(l) \to Ter(\Sigma)$ be a substitution that sets $\sigma(x) = s_x$ for each variable $x \in var(r)$, and which sets $\sigma(y) = s_y$ for each variable $y \in var(l) \backslash var(r)$. Then the term l^σ belongs to $\mathcal{L}(\mathcal{A}, q)$ and additionally, by Condition 2 of Definition 4, $l^\sigma \multimap\!\!\to r^\theta$. ◀

4 Bounding the Size of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$

In this section we will establish an upper bound for the size of the tree automaton $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ in terms of the size of $\mathcal{A}$, the maximum state in-degree of $\mathcal{A}$, and several parameters extracted from the term rewriting system $\mathfrak{R}$. Subsequently, we will use Lemma 7, together with the size upper bound mentioned above to establish the fixed parameter tractability of reachability and joinability in depth d.

A morphism from a tree automaton $\mathcal{A} = (Q, \Sigma, F, \Delta)$ to a tree automaton $\mathcal{A}' = (Q', \Sigma, F', \Delta')$ is a function $\mu : Q \to Q'$ such that for each transition $f(q_1, ..., q_{\mathfrak{a}(f)}) \to q$ in Δ, the transition $f(\mu(q_1), ..., \mu(q_{\mathfrak{a}(f)})) \to \mu(q)$ is in Δ'. As an abuse of notation we write $\mu : \mathcal{A} \to \mathcal{A}'$ to denote such a morphism.

Let $l \in Ter(\Sigma \cup X)$ and $\mathcal{A}(l)$ be the tree automaton associated with l. We say that a morphism $\mu : \mathcal{A}(l) \to \mathcal{A}$ from $\mathcal{A}(l)$ to $\mathcal{A}$ is rooted at a state $q \in \mathcal{A}$ if $\mu(q^l_\varepsilon) = q$ where q^l_ε is the unique maximal state of $\mathcal{A}(l)$. Each such morphism μ defines a state-substitution $\gamma : var(l) \to Q$ which is defined by setting $\gamma(x) = \mu(q^l_x)$ for each variable $x \in var(l)$. Intuitively, the morphism μ specifies a run of the automaton $\mathcal{A}$ (i.e. a sequence of transitions from $\mathcal{A}$) which are applied to reduce the configuration l^γ to the state q.

For each tree automaton $\mathcal{A}$, each state q of $\mathcal{A}$, and each positive integer s, we denote by $\eta(\mathcal{A}, q, s)$ the set of all pairs (l, μ) where l is a term in $Ter(\Sigma \cup X)$ of size at most s, and μ is a morphism from $\mathcal{A}(l)$ to $\mathcal{A}$ rooted at q.

In the following lemma we show an upper bound on the size of $\eta(\mathcal{A}, q, s)$ in terms of the maximum state in-degree of $\mathcal{A}$.

▶ **Lemma 8.** *Let $\mathcal{A}$ be a tree-automaton over Σ of maximum state in-degree δ, q be a state of $\mathcal{A}$, and s be a positive integer. Let $\mathfrak{a}$ be the maximum arity of a function symbol in Σ. Then $|\eta(\mathcal{A}, q, s)| \leq (e \cdot \max\{\delta, \mathfrak{a}\})^{2s+1}$.*

Proof. Let $T(\delta)$ be the rooted infinite δ-regular tree. Let $b_{k,\delta}$ be the number of rooted subtrees of $T(\delta)$ of size k. It is well known that $b_{k,\delta} \leq \binom{\delta \cdot k}{k}$ (See for instance [6, 3]). Using the inequality $\binom{n}{k} \leq \left(\frac{e \cdot n}{k}\right)^k$ (where $e \approx 2.71$ is the Euler number) we have that $b_{k,\delta} \leq (e \cdot \delta)^k$. This implies that the number $c_{k,\delta}$ of rooted subtrees of $T(\delta)$ of size *at most* k is upper bounded by $c_{k,\delta} \leq (e \cdot \delta)^{k+1}$.

Now let $\mathrm{U}(\mathcal{A}, q)$ be the unfolding of $\mathcal{A}$ rooted at q. Since the maximum in-degree of $\mathcal{A}$ is δ, we have that U is a rooted infinite tree of degree at most $\max\{\mathfrak{a}, \delta\}$. Additionally, if l is a term in $Ter(\Sigma \cup X)$ of size s, then for each pair $(l, \mu) \in \eta(\mathcal{A}, q, s)$ the image of $\mathcal{A}(l)$ under μ on $\mathcal{A}$ corresponds unequivocally to a rooted subtree of $\mathrm{U}(\mathcal{A}, q)$ of size at most $2s$. Therefore, the number of possible morphisms rooted at q from $\mathcal{A}(t)$ to $\mathcal{A}$ where t is a term of size s is upper bounded by the number of rooted subtrees of $\mathrm{U}(\mathcal{A}, q)$ of size at most $2s$. This implies that $|\eta(\mathcal{A}, q, s)| \leq (e \cdot \delta)^{2s+1}$ if $\delta \geq \mathfrak{a}$, and $|\eta(\mathcal{A}, q, s)| \leq (e \cdot \mathfrak{a})^{2s+1}$ if $\mathfrak{a} \geq \delta$. ◀

The next lemma establishes an upper-bound for the size and for the state in-degree of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ in terms of the size of $\mathcal{A}$, state in-degree of $\mathcal{A}$, and several parameters extracted from the term rewriting system $\mathfrak{R}$.

▶ **Lemma 9.** *Let $\mathcal{A}$ be a tree automaton of maximum in-degree δ. Let $\mathfrak{R}$ be a linear term rewriting system. Let s_1 be the maximum size of the left-hand side of a rule in $\mathfrak{R}$, and s_2 be the maximum size of a right-hand side of a rule in $\mathfrak{R}$. Let ρ be the maximum number of rules in $\mathfrak{R}$ with the same left-hand side.*

1. *The maximum in-degree of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ is at most $\delta + \rho \cdot (e \cdot \max\{\mathfrak{a}, \delta\})^{2s_1+1}$.*
2. *The size of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ is at most $|\mathcal{A}| + 2 \cdot s_2 \cdot \rho \cdot (e \cdot \max\{\mathfrak{a}, \delta\})^{2s_1+1} \cdot |\mathcal{A}|$.*
3. *$\mathcal{N}(\mathcal{A}, \mathfrak{R})$ can be constructed in time $O\left(s_2 \cdot \rho \cdot (e \cdot \max\{\mathfrak{a}, \delta\})^{2s_1+1} \cdot (\log |\mathfrak{R}|) \cdot |\mathcal{A}|\right)$.*

Proof.

1. Let q be a state of $\mathcal{A}$. The in-degree of q in $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ is equal to the in-degree of q in $\mathcal{A}$ plus the number of right-components $\mathcal{C}(\mathcal{A}, q, l \to r, \gamma)$ of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ rooted at q, where $l \to r$ is a rule in $\mathfrak{R}$ and γ is a good state-substitution for (q, l). By Lemma 8 there is at most $(e \cdot \max\{\mathfrak{a}, \delta\})^{2s_1+1}$ pairs of the form (l, μ) where where l is a term of size at most s_1 and $\mu : \mathcal{A}(l) \to \mathcal{A}$ is a morphism from $\mathcal{A}(l)$ to $\mathcal{A}$ rooted at q. Therefore the number of components $\mathcal{C}(\mathcal{A}, q, l \to r, \gamma)$ is upper bounded by $(e \cdot \max\{\mathfrak{a}, \delta\})^{2s_1+1}$. Since the number of rules with same left-hand side is upper bounded by ρ, we have that each pair $(l, \mu) \in \eta(\mathcal{A}, q, s_1)$ gives rise to at most ρ right components rooted at q. Therefore, the in-degree of q in $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ is at most $\delta + \rho \cdot (e \cdot \max\{\mathfrak{a}, \delta\})^{2s_1+1}$.
2. As argued in the previous item, the number of right components of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ rooted at q is upper bounded by $\rho \cdot (e \cdot \max\{\mathfrak{a}, \delta\})^{2s_1+1}$. Since each right-component $\mathcal{C}(\mathcal{A}, \gamma, l \to r)$ is isomorphic to $\mathcal{A}(r)$, we have that such component has size at most $2 \cdot |r| \leq 2 \cdot s_2$. Therefore, the size of $\mathcal{N}(\mathcal{A}, \mathfrak{R})$ is upper bounded by $|\mathcal{A}| + 2 \cdot s_2 \cdot \rho \cdot (e \cdot \max\{\mathfrak{a}, \delta\})^{2s_1+1} \cdot |\mathcal{A}|$.
3. Assume that $\mathfrak{R}$ is specified as a lexicographically ordered list of rules. For each state q of $\mathcal{A}$, we can enumerate in time $O((e \cdot \max\{\mathfrak{a}, \delta\})^{2s_1+1})$ the set of all pairs (l, γ) where $l \in \mathit{Ter}(\Sigma \cup X)$ and γ is a morphism from $\mathcal{A}(l)$ to $\mathcal{A}$. For each of these pairs, we use binary search to look up for the existence of a rule in $\mathfrak{R}$ having l as left-hand side. Each such look up takes time $O(\log |\mathfrak{R}|)$. Finally, for each rule $l \to r$ in $\mathfrak{R}$, we create the right component $\mathcal{C}(\mathcal{A}, q, l \to r, \gamma)$. The addition of each such component takes time $O(|r|) \leq O(s_2)$. Since there are at most ρ rules with right-hand side l, the total amount of time to construct the automaton is $O\left(s_2 \cdot \rho \cdot (e \cdot \max\{\mathfrak{a}, \delta\})^{2s_1+1} \cdot (\log |\mathfrak{R}|) \cdot |\mathcal{A}|\right)$. ◀

Now let $t \in \mathit{Ter}(\Sigma)$ be a ground term over Σ, and let $\mathfrak{R}$ be a linear term rewriting system. Let $\mathcal{A}(t)$ be the tree automaton associated with t. For each $d \in \mathbb{N}$ we inductively define the tree automaton $\mathcal{A}(t, \mathfrak{R}, d)$ as follows.

$$\mathcal{A}(t, \mathfrak{R}, d) = \begin{cases} \mathcal{A}(t) \text{ if } d = 0 \\ \\ \mathcal{N}(\mathcal{A}(t, \mathfrak{R}, d-1), \mathfrak{R}) \qquad \text{if } d \geq 1. \end{cases} \tag{4}$$

Note that $\mathcal{L}(\mathcal{A}(t, \mathfrak{R}, 0)) = \mathcal{L}(\mathcal{A}(t)) = \{t\}$. Additionally, from Lemma 7 it follows straightforwardly by induction on d that $\mathcal{A}(t, \mathfrak{R}, d)$ is a tree automaton recognizing precisely the ground terms in $\mathit{Ter}(\Sigma)$ that can be reached from t in depth at most d. Let $\mathfrak{a}$ be the maximum arity of a function symbol in Σ. Then the maximum in-degree of a vertex of $\mathcal{A}(t, \mathfrak{R}, 0)$ is $\mathfrak{a}$, while the size of $\mathcal{A}(t, \mathfrak{R}, 0)$ is $2 \cdot |t|$. The next proposition establishes upper bounds for the size and maximum in-degree of the tree automaton $\mathcal{A}(t, \mathfrak{R}, d)$.

▶ **Proposition 10.** *Let $\mathfrak{a}$ be the maximum arity of a function symbol in Σ. Let s_1 be the maximum size of the left-side of a rule in $\mathfrak{R}$, s_2 be the maximum size of the right-side of a term in $\mathfrak{R}$, and ρ be the maximum number of rules in $\mathfrak{R}$ with the same left-hand side.*

- *The maximum in-degree of $\mathcal{A}(t, \mathfrak{R}, d)$ is at most $(e \cdot \rho \cdot \mathfrak{a})^{s_1^{2 \cdot d}}$.*
- *The size of $\mathcal{A}(t, \mathfrak{R}, d)$ is at most $s_2^d \cdot (e \cdot \rho \cdot \mathfrak{a})^{s_1^{2 \cdot d}} \cdot |t|$.*
- *$\mathcal{A}(t, \mathfrak{R}, d)$ can be constructed in time $s_2^d \cdot (e \cdot \rho \cdot \mathfrak{a})^{s_1^{2 \cdot d}} \cdot (\log |\mathfrak{R}|) \cdot |t|$.*

We omit the proof of Proposition 10 since it follows straightforwardly from Lemma 9 by induction on d. Finally, we are in a position to prove Theorem 1.

Proof of Theorem 1

1. **Reachability.** Let $f(\mathfrak{R}, d) = s_2^d \cdot (e \cdot \rho \cdot \mathfrak{a})^{s_1^{2d}} \cdot (\log |\mathfrak{R}|)$. Given two ground terms t and t' we want to determine whether t' can be derived from t in depth at most d. First, we construct in time $f(\mathfrak{R}, d) \cdot |t|$ the tree automaton $\mathcal{A}(t, \mathfrak{R}, d)$, whose size is at most $f(\mathfrak{R}, d) \cdot |t|$. Then we determine whether $\mathcal{A}(t, \mathfrak{R}, d)$ accepts the term t'. By Lemma 2 this membership test can be performed in time $f(\mathfrak{R}, d) \cdot |t| \cdot |t'|$.
2. **Joinability.** Given two ground terms t and t' we want to determine whether there exists a term u such that u can be derived both from t and from t' in depth at most d. First we construct the tree automata $\mathcal{A}(t, \mathfrak{R}, d)$ and $\mathcal{A}(t', \mathfrak{R}, d)$ whose sizes are respectively upper bounded by $f(\mathfrak{R}, d) \cdot |t|$ and $f(\mathfrak{R}, d) \cdot |t'|$. We have that there exists a term u that can be reached by both t and t' in depth at most d if and only if $\mathcal{A}(t, \mathfrak{R}, d) \cap \mathcal{A}(t', \mathfrak{R}, d)$ is non-empty. By Lemma 3, this emptiness of intersection test can be realized in time $f(\mathfrak{R}, d)^2 \cdot |t| \cdot |t'|$.

5 Conclusion

In this work we have shown that reachability and joinability in linear term rewriting systems are fixed parameter tractable with respect to the depth of derivations. More precisely, we showed that given a linear term rewriting system $\mathfrak{R}$, and ground terms t and t' one can determine in time $f(\mathfrak{R}, d) \cdot |t| \cdot |t'|$ whether t' is reachable from t in depth at most d, and in time $f(\mathfrak{R}, d)^2 \cdot |t| \cdot |t'|$ whether t and t' are joinable in depth at most d. We note that the function $f(\mathfrak{R}, d)$ depends double exponentially on d. We leave open the problem of determining whether the dependence on d in the function $f(\mathfrak{R}, d)$ can be substantially improved.

References

1 Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge university press, 1999.

2 Leo Bachmair. Rewrite techniques in theorem proving. In *Proc. of the 5th International Conference on Rewriting Techniques and Applications*, LNCS, pages 1–1, 1993.

3 Andrew Beveridge, Alan Frieze, and Colin McDiarmid. Random minimum length spanning trees in regular graphs. *Combinatorica*, 18(3):311–333, 1998.

4 Marc Bezem, Jan Willem Klop, and Roel de Vrijer. Terese. term rewriting systems. *Cambridge Tracts in Theoretical Computer Science*, 55, 2003.

5 H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.

6 Colin Cooper and Alan Frieze. Component structure of the vacant set induced by a random walk on a random graph. *Random Structures & Algorithms*, 42(2):135–158, 2013.

7 N. Dershowitz and J. P. Jouannaud. Rewrite systems. *Handbook of Theoretical Computer Science (Chapter 6)*, pages 243–320, 1990.

8 Bertram Felgenhauer and René Thiemann. Reachability analysis with state-compatible automata. In *Proc. of the 8th International Conference on Language and Automata Theory and Applications*, LNCS, pages 347–359. Springer, 2014.

9 Guillaume Feuillade, Thomas Genet, and Valérie Viet Triem Tong. Reachability analysis over term rewriting systems. *Journal of Automated Reasoning*, 33(3-4):341–383, 2004.

10 Ferenc Gécseg and Magnus Steinby. Tree languages. In *Handbook of formal languages*, pages 1–68. Springer, 1997.

11 Thomas Genet. Decidable approximations of sets of descendants and sets of normal forms. In *Proc. of the 9th International Conference on Rewriting Techniques and Applications*, LNCS, pages 151–165, 1998.

12 Rémy Gilleron and Sophie Tison. Regular tree languages and rewrite systems. *Fundamenta informaticae*, 24(1, 2):157–175, 1995.

13 Guillem Godoy, Robert Nieuwenhuis, and Ashish Tiwari. Classes of term rewrite systems with polynomial confluence problems. *ACM Transactions on Computational Logic (TOCL)*, 5(2):321–331, 2004.

14 Guillem Godoy, Ashish Tiwari, and Rakesh Verma. Characterizing confluence by rewrite closure and right ground term rewrite systems. *Applicable Algebra in Engineering, Communication and Computing*, 15(1):13–36, 2004.

15 Lukasz Kaiser. Confluence of right ground term rewriting systems is decidable. In *Foundations of Software Science and Computational Structures*, pages 470–489. Springer, 2005.

16 Jan Willem Klop, Marc Bezem, and RC De Vrijer. *Term rewriting systems*. Cambridge University Press, 2001.

17 Martin Korp and Aart Middeldorp. Match-bounds revisited. *Information and Computation*, 207(11):1259–1283, 2009.

18 Takashi Nagaya and Yoshihito Toyama. Decidability for left-linear growing term rewriting systems. In *Proc. of the 10th International Conference on Rewriting Techniques and Applications*, LNCS, pages 256–270, 1999.

19 Vincent van Oostrom. Normalisation in weakly orthogonal rewriting. In *Proc. of the 10th International Conference on Rewriting Techniques and Applications*, LNCS, pages 60–74, 1999.

20 Rakesh M Verma, Michael Rusinowitch, and Denis Lugiez. Algorithms and reductions for rewriting problems. *Fundamenta Informaticae*, 46(3):257–276, 2001.