Theory Comput. Systems **39**, 347–383 (2006)
DOI: 10.1007/s00224-004-1170-6

Theory of Computing Systems

Reachability Problems on Regular Ground Tree Rewriting Graphs

Christof Löding

Lehrstuhl für Informatik VII, RWTH Aachen,
52056 Aachen, Germany
loeding@informatik.rwth-aachen.de

Abstract. We consider the transition graphs of regular ground tree (or term) rewriting systems. The vertex set of such a graph is a (possibly infinite) set of trees. Thus, with a finite tree automaton one can represent a regular set of vertices. It is known that the backward closure of sets of vertices under the rewriting relation preserves regularity, i.e., for a regular set T of vertices the set of vertices from which one can reach T can be accepted by a tree automaton. The main contribution of this paper is to lift this result to the recurrence problem, i.e., we show that the set of vertices from which one can reach infinitely often a regular set T is regular, too. Since this result is effective, it implies that the problem whether, given a tree t and a regular set T, there is a path starting in t that infinitely often reaches T, is decidable. Furthermore, it is shown that the problems whether all paths starting in t eventually (respectively, infinitely often) reach T, are undecidable. Based on the decidability result we define a fragment of temporal logic with a decidable model-checking problem for the class of regular ground tree rewriting graphs.

1. Introduction

An important subject in theoretical computer science is the automated verification of programs and processes. In this context graphs play an important role for the formal description of state-based systems. The use of theoretically unbounded data structures (such as stacks, queues, etc.) in these systems requires the use of infinite (transition-)graphs. In order to allow algorithmic applications, these systems have to be given effectively, i.e., by some finite object. In recent years many classes of finitely representable infinite graphs have been studied. A starting point for this research on infinite graphs was an analysis of the configuration graphs of pushdown automata by Muller and Schupp in [21]. The vertices of pushdown graphs are words (the control state followed by the stack

content) and the edges are defined via the transitions which are prefix rewriting rules on these words. In [21] it is shown that pushdown graphs have a decidable monadic second-order (MSO) theory. Later, more efficient algorithms were developed for solving reachability problems, model-checking temporal logics [12], and synthesizing strategies for games [16], [22] on pushdown graphs.

Using rewriting as the basic process for generation of infinite graphs, a variety of possibilities arises to define classes of graphs different from pushdown graphs. We briefly review some of the approaches. The result on the decidability of the MSO theory generalizes from pushdown graphs to prefix recognizable graphs [3], which are also generated by prefix rewriting on words; but the rewriting rules refer to regular languages instead of single words. Even more general classes of graphs can be obtained when using finite word transducers to define the edge relations of graphs. This leads to automatic graphs in the case of synchronous transducers (see, e.g., [1]) and to rational graphs [20] in the case of asynchronous transducers. The formalisms to generate automatic and rational graphs are very strong and even simple reachability problems on these graphs are undecidable. In [19] model-checking problems for process rewriting graphs are studied (in process rewriting parallel composition of words is allowed in addition to the usual sequential composition).

In this paper we use another generalization of word rewriting, namely tree rewriting, as already considered in [2]. In tree (or term) rewriting the basic objects in the rewriting systems are trees. For ground tree rewrite systems (GTRS) confluence [8], the first-order theory [9] and several reachability problems [7] have been shown to be decidable. Here we study a more general form of ground tree rewriting, namely regular ground tree rewriting (RGTR), as already considered in [11] and in a slightly different way in [4]. In RGTR the rewriting rules are of the form $T_1 \hookrightarrow T_2$, where T_1 and T_2 are regular sets of trees. Such a rule can be applied to any tree that has a subtree from T_1 by replacing this subtree by an arbitrary tree from T_2. For the transition graphs of these systems, we address the following problems, where the sets T and T' denote regular sets of vertices, specified by tree automata:

(1) Compute the set of all vertices from where one can reach T.
(2) Compute the set of all vertices from where one can infinitely often visit T.
(3) Given a single tree t, do all paths starting in t eventually (respectively, infinitely often) visit T?
(4) Given a single tree t, does there exist a path starting in t that remains in T' until it eventually reaches T?

In the context of term rewriting (see [5]–[7]) it has already been shown that the set from (1) is regular and that one can effectively construct a finite tree automaton accepting this set. The main contribution of this paper is to show that the set of vertices from (2) is regular, too, and to provide an algorithm for computing a tree automaton accepting this set. From now on we call the problem described in (2) the recurrence problem. Sometimes we also refer to the corresponding decision problem, i.e., given a tree t and a regular set T of trees, does there exist a path starting in t that infinitely often visits T? Since we give an algorithm to compute a tree automaton representing the set from (2), this also allows us to solve the decision problem because it can easily be tested whether the computed tree automaton accepts the given tree t.

Problems (3) and (4) are shown to be undecidable and hence mark a boundary in the design of a fragment of temporal logic with a decidable model-checking problem for RGTR graphs.

This paper is a complete and extended version of [17]. We provide full proofs for the facts stated in [17], extend the algorithm for the recurrence problem to RGTR instead of simple ground tree rewriting and provide a complexity analysis for this algorithm.

The remainder of this paper is organized as follows. In Section 2 we introduce the basic terminology and definitions. In Section 3 we introduce the reachability problems under consideration and briefly discuss simple reachability and one step reachability. In Section 4 we develop the algorithm to solve the recurrence problem. The undecidable reachability problems are considered in Section 5. In Section 6 we use the results from Sections 3 and 4 to define a temporal logic that has a decidable model-checking problem for the class of RGTR graphs.

2. Regular Ground Tree Rewriting Graphs

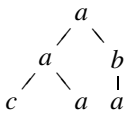
We start with the basic terminology concerning graphs, trees, tree automata, and tree rewriting.

Graphs. A directed edge labeled graph G is a tuple $G = (V, E, \Sigma)$, where V is the set of vertices, Σ is the finite set of edge labels, and $E \subseteq V \times \Sigma \times V$ is the set of edges. We only consider countable graphs, i.e., the set V is countable. Whenever we use the notion of a graph without specifying the type of the graph in more detail, then we mean a countable directed edge labeled graph. If we are not interested in the edge labels or if we consider graphs without edge labels, we omit Σ and assume that $E \subseteq V \times V$.

For a vertex $u \in V$, a successor of v is a vertex $v \in V$ such that $(u, v) \in E$. A path π in G is a nonempty sequence of vertices $\pi = v_0 \cdots v_n$ such that $(v_i, v_{i+1}) \in E$ for all $i \in \{0, \ldots, n-1\}$. The length of a path $\pi = v_0 \cdots v_n$ is n. For two vertices $u, v \in V$ we say that there is a path from u to v if there is a path $\pi = v_0 \cdots v_n$ in G with $v_0 = u$ and $v_n = v$. Infinite paths are defined in the same way as finite paths but with an infinite sequence of vertices. A path is maximal if it is infinite or if it is finite and cannot be extended. For a path π we denote by $\pi(i)$ the ith vertex on π and by $\pi[i, \infty)$ we denote the suffix of π starting in $\pi(i)$.

Ranked Trees. For a set X we denote by X^* the set of all finite sequences over X and by X^+ all finite nonempty sequences over X. For $w \in X^*$ the length of w is denoted by $|w|$ and the empty sequence is denoted by ε. By $\mathbb{N}$ we denote the set of natural numbers, i.e., the set of non-negative integers. By $\sqsubseteq$ we denote the prefix ordering on $\mathbb{N}^*$ and by $\sqsubset$ the strict prefix ordering. That is, $x \sqsubseteq y$ for $x, y \in \mathbb{N}^*$ if there is $z \in \mathbb{N}^*$ such that $y = xz$, and $x \sqsubset y$ if $x \sqsubseteq y$ and $x \neq y$.

A ranked alphabet A is a finite family of finite sets $(A_i)_{i \in [k]}$, where $[k] = \{0, \ldots, k\}$. In the following, k always denotes the maximum rank in A. For simplicity we identify A with the set $\bigcup_{i=0}^{k} A_i$. To specify a ranked alphabet one can list all the sets A_i or simply specify the set of all symbols together with their ranks.



$a(a(c, a), b(a))$

graphical notation term notation

Fig. 1. The tree from Example 1.

A ranked tree t over A is a mapping $t : D_t \to A$ with $D_t \subseteq [k-1]^*$ such that

— D_t is finite and prefix closed,
— $D_t \neq \emptyset$,
— for each $x \in \mathbb{N}^*$ and $i \in \mathbb{N}$: if $xi \in D_t$, then $xj \in D_t$ for all $j \leq i$, and
— if $x0, \ldots, x(i-1) \in D_t$ and $xi \notin D_t$, then $t(x) \in A_i$.

The set D_t is called the domain of t and the elements of D_t are called the locations of t (we do not call them vertices to separate them clearly from vertices of a graph). For $x, y \in D_t$ we call x the predecessor of y and y a successor of x if there is $i \in \mathbb{N}$ such that $y = xi$. The set of all ranked trees over A is denoted by T_A.

We use the usual term notation and graphical notation for trees as shown in the following example.

Example 1. Let A be the ranked alphabet given by $A_0 = \{a, c\}$, $A_1 = \{b\}$, $A_2 = \{a\}$. Then $t : \{\varepsilon, 0, 1, 00, 01, 10\} \to A$ with $t(\varepsilon) = t(0) = t(01) = t(10) = a$, $t(1) = b$, and $t(00) = c$ is a ranked tree over A. The graphical and term notations are shown in Figure 1.

Given a tree t and a location x of t, the subtree of t at location x is the tree obtained by taking all locations of t that have x as prefix, removing the prefix x from all these locations, and keeping the labels. This is formalized as follows. For $x \in \mathbb{N}^*$ we define $xD_t = \{xy \in \mathbb{N}^* \mid y \in D_t\}$ and $x^{-1}D_t = \{y \in \mathbb{N}^* \mid xy \in D_t\}$. For $x \in D_t$ the subtree $t^{\downarrow x}$ of t at x is the tree with domain $D_{t^{\downarrow x}} = x^{-1}D_t$ and $t^{\downarrow x}(y) = t(xy)$.

To formalize the concept of replacing a subtree of a given tree by another tree we introduce the notion of substitution. A substitution is a pair consisting of a location $x \in \mathbb{N}^*$ and a tree $s \in T_A$, written as $[x/s]$. A substitution $[x/s]$ can be applied to a tree t if $x \in D_t$. The result $t[x/s]$ of a substitution applied to t is the tree t' with domain $D_{t'} = (D_t \setminus xD_{t^{\downarrow x}}) \cup xD_s$ and

$$t'(y) = \begin{cases} t(y) & \text{if} \quad y \in D_t \setminus xD_{t^{\downarrow x}}, \\ s(z) & \text{if} \quad y = xz \quad \text{with} \quad z \in D_s. \end{cases}$$

This means we replace the subtree $t^{\downarrow x}$ in t by s.

Ground Tree Rewriting. A ground tree rewriting system (GTRS) is a tuple $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$, where $A = (A_i)_{i \in [k]}$ is a ranked alphabet, Σ is an alphabet, R is a finite set of rules of the form $s \overset{\sigma}{\hookrightarrow} s'$ with $s, s' \in T_A$, $\sigma \in \Sigma$, and $t_{\text{in}} \in T_A$ is the initial

tree. The set of rules defines what kind of substitutions are compatible with $\mathcal{R}$ as follows. A substitution $[x/s']$ is $(\mathcal{R}, \sigma)$-applicable to a tree $t \in T_A$ if $x \in D_t$ and if there is a rule $s \overset{\sigma}{\hookrightarrow} s' \in R$ with $s = t^{\downarrow x}$. It is $\mathcal{R}$-applicable to t if it is $(\mathcal{R}, \sigma)$-applicable to t for some $\sigma \in \Sigma$. We write

— $t \to^{\sigma}_{\mathcal{R}} t'$ if there is an $(\mathcal{R}, \sigma)$-applicable substitution $[x/s']$ such that $t[x/s'] = t'$,
— $t \to_{\mathcal{R}} t'$ if there is $\sigma \in \Sigma$ with $t \to^{\sigma}_{\mathcal{R}} t'$, and
— $\to^{*}_{\mathcal{R}}$ for the transitive and reflexive closure of $\to_{\mathcal{R}}$.

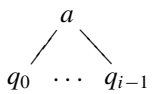
Pushdown automata (see [15]) are simple examples for GTRSs if we consider the symbols of the stack alphabet as unary symbols and the control states as constants. The rewrite rules correspond to the transitions of the pushdown automaton and the initial tree to the initial configuration of the pushdown automaton.

We want to consider a generalization of ground tree rewriting that does not just use single trees on each side of the rules but sets of trees instead. For this purpose we introduce tree automata.

Tree Automata. Finite automata over finite words are a formalism to represent certain kinds of infinite sets of words, the so-called regular sets or regular languages. The theory of finite automata and regular word languages can be transferred to finite ranked trees by means of tree automata. Tree automata are devices with finite memory that read input trees and accept or reject them. In the following we introduce the models of nondeterministic bottom-up automata with and without ε-transitions. For a more comprehensive introduction to tree automata see, e.g., [14] or [5].

A nondeterministic tree automaton (NTA) is a tuple $\mathcal{A} = (Q, A, \Delta, F)$, where Q is a finite set of states, $A = (A_i)_{i \in [k]}$ is a ranked alphabet, $F \subseteq Q$ is a set of final states, and $\Delta \subseteq (\bigcup_{i=0}^{k} Q^i \times A_i) \times Q$ is the transition relation.

Tree automata can be viewed as special GTRSs over the alphabet A with A_0 augmented by the elements of Q. The elements of the transition relation are interpreted as rewriting rules as follows. Let $q_0, \ldots, q_{i-1}, q \in Q$ and $a \in A_i$. The transition $(q_0, \ldots, q_{i-1}, a, q)$ corresponds to the (unlabeled) rewriting rule



$\hookrightarrow \; q.$

This allows us to use the terminology of GTRSs from the previous section to define the set $T(\mathcal{A})$ of trees accepted by the NTA $\mathcal{A}$:

$$T(\mathcal{A}) = \{t \in T_A \mid \exists q \in F \; : \; t \to^{*}_{\mathcal{A}} q\}.$$

So $T(\mathcal{A})$ is the set of all trees that can be transformed into a final state using the transitions as rewriting rules. A set of trees that can be accepted by an NTA is called regular.

Another way of defining acceptance of an NTA is to use the notion of run. A run ρ of $\mathcal{A}$ on a tree t is a mapping $\rho : D_t \to Q$ such that for each $x \in D_t$, if x has i successors in t, then $(\rho(x0), \ldots, \rho(x(i-1)), t(x), \rho(x)) \in \Delta$. We call this transition the transition used in ρ at location x. A run ρ is accepting if $\rho(\varepsilon) \in F$. One can easily verify that $t \to^{*}_{\mathcal{A}} q$ iff there is a run of $\mathcal{A}$ on t with $\rho(\varepsilon) = q$.

In Section 4 we also need partial runs. A partial run of $\mathcal{A}$ on a tree t is a partial function $\rho : D_t \to Q$ such that for each $x \in D_t$, if $\rho(x)$ is defined and x has i successors in t, then $\rho(x0), \ldots, \rho(x(i-1))$ are defined and $(\rho(x0), \ldots, \rho(x(i-1)), t(x), \rho(x)) \in \Delta$.

In many proofs, if a tree t can be reduced to a state p of an automaton $\mathcal{A}$, i.e., $t \to_{\mathcal{A}}^* p$, we are interested in the state that is used at a specific location $x \in D_t$ in this reduction. If this state is q, then we write

$$t \to_{\mathcal{A}}^* t[x/q] \to_{\mathcal{A}}^* p.$$

This means that the automaton reduces the subtree of t at location x to the state q and then reduces the remaining part of t to p. In the following remark we give two simple arguments that are used repeatedly in connection with this notation.

Remark 1. Let $\mathcal{A} = (Q, A, \Delta, F)$ be an NTA, let $p, q \in Q$, let $t, s \in T_A$, and let $x \in D_t$.

(i) $t \to_{\mathcal{A}}^* t[x/q]$ iff $t^{\downarrow x} \to_{\mathcal{A}}^* q$.
(ii) If $t[x/q] \to_{\mathcal{A}}^* p$ and $s \to_{\mathcal{A}}^* q$, then $t[x/s] \to_{\mathcal{A}}^* p$.

The proof of this remark is trivial and therefore omitted, but we think that isolating these two facts might ease the reading of several proofs.

An NTA as defined above reduces a tree t over A to a single state by removing from t one occurrence of a symbol from A in each rewriting step. To simplify constructions of automata we introduce an extension of this model by allowing the automaton to change a state without removing one of the input symbols. This leads to the model ε-NTA, which is equivalent to the model of NTA in the sense that the class of accepted languages consists of all regular languages.

An ε-NTA is a tuple $\mathcal{A} = (Q, A, \Delta, F)$, where Q, A, F are as for NTA and $\Delta \subseteq ((\bigcup_{i=0}^{k} Q^i \times A_i) \times Q) \cup (Q \times Q)$ is the transition function. The only difference to NTA is that transitions of the form (p, q) for $p, q \in Q$ are allowed. Transitions of this kind are called ε-transitions. An ε-transition (p, q) for $p, q \in Q$ corresponds to a rewriting rule $p \hookrightarrow q$. The set $T(\mathcal{A})$ accepted by an ε-NTA is defined as for NTAs. Note, that the notion of run is not defined for ε-NTAs.

An important property of the class of regular tree languages is its closure under Boolean operations. Here, we give constructions for union and intersection. For other closure properties and correctness proofs for subsequent constructions we refer the reader to [5].

For the complexity analysis of automaton constructions we need the size $|\mathcal{A}|$ of a tree automaton. Similar to [5] we define the size $|\alpha|$ of a transition $\alpha = (q_0, \ldots, q_{i-1}, a, q) \in \Delta$ as $|\alpha| = i + 2$. The size of an ε-transition α is 2. The size of $\mathcal{A}$ is the number of states plus the sum of the size of all transitions in Δ, i.e.,

$$|\mathcal{A}| = |Q| + \sum_{\alpha \in \Delta} |\alpha|.$$

To refer to the language of an automaton with a certain state as final state we define for $q \in Q$ the automaton $\mathcal{A}(q) = (Q, A, \Delta, \{q\})$. With this definition we get $t \in T(\mathcal{A}(q))$ iff $t \to_{\mathcal{A}}^* q$.

Let $\mathcal{A}_1 = (Q_1, A, \Delta_1, F_1)$, $\mathcal{A}_2 = (Q_2, A, \Delta_2, F_2)$ be two ε-NTAs with $Q_1 \cap Q_2 = \emptyset$. The automaton $\mathcal{A}_1 \cup \mathcal{A}_2$ recognizing the union of $T(\mathcal{A}_1)$ and $T(\mathcal{A}_2)$ is defined as

$$\mathcal{A}_1 \cup \mathcal{A}_2 = (Q_1 \cup Q_2, A, \Delta_1 \cup \Delta_2, F_1 \cup F_2).$$

The automaton $\mathcal{A}_1 \times \mathcal{A}_2$ for the intersection of $T(\mathcal{A}_1)$ and $T(\mathcal{A}_2)$ is defined as

$$\mathcal{A}_1 \times \mathcal{A}_2 = (Q_1 \times Q_2, A, \Delta_\times, F_1 \times F_2),$$

where $\Delta_\times$ contains the transitions of the form

— $((p_0, q_0), \ldots, (p_{i-1}, q_{i-1}), a, (p, q))$ for $(p_0, \ldots, p_{i-1}, a, p) \in \Delta_1$ and $(q_0, \ldots, q_{i-1}, a, q) \in \Delta_2$,
— $((p, q), (p', q))$ for $(p, p') \in \Delta_1$, and
— $((p, q), (p, q'))$ for $(q, q') \in \Delta_2$.

Proposition 1. *Let $\mathcal{A}_1, \mathcal{A}_2$ be two ε-NTAs. Then*

(i) $T(\mathcal{A}_1 \cup \mathcal{A}_2) = T(\mathcal{A}_1) \cup T(\mathcal{A}_2)$ *with* $|\mathcal{A}_1 \cup \mathcal{A}_2| \in \mathcal{O}(|\mathcal{A}_1| + |\mathcal{A}_2|)$, *and*
(ii) $T(\mathcal{A}_1 \times \mathcal{A}_2) = T(\mathcal{A}_1) \cap T(\mathcal{A}_2)$ *with* $|\mathcal{A}_1 \times \mathcal{A}_2| \in \mathcal{O}(|\mathcal{A}_1| \cdot |\mathcal{A}_2|)$.

As for automata on finite words, complementation of NTAs uses the subset construction for determinization.

Proposition 2. *For each ε-NTA $\mathcal{A}$ there is an ε-NTA $\overline{\mathcal{A}}$ with $T(\overline{\mathcal{A}}) = T_A \backslash T(\mathcal{A})$ and $|\overline{\mathcal{A}}| \in \mathcal{O}(2^{|\mathcal{A}|})$.*

For later use we need an operation that transforms an ε-NTA $\mathcal{A}$ into an ε-NTA $\mathcal{A}_{!\varepsilon}$ accepting exactly those trees that are accepted by $\mathcal{A}$ with a reduction that uses exactly one ε-transition. This can easily be achieved by a construction that redirects ε-transitions to a copy of the state set of $\mathcal{A}$.

Proposition 3. *Let $\mathcal{A} = (Q, A, \Delta, F)$ be an ε-NTA and denote by $\mathcal{A}'$ the NTA that is obtained from $\mathcal{A}$ by removing the ε-transitions. One can construct in time $\mathcal{O}(|\mathcal{A}|)$ an ε-NTA $\mathcal{A}_{!\varepsilon}$ of size $\mathcal{O}(|\mathcal{A}|)$ that accepts a tree t iff there are $q_1, q_2 \in Q$ and $q \in F$ such that $t \to^*_{\mathcal{A}'} t[x/q_1] \to_{\mathcal{A}} t[x/q_2] \to^*_{\mathcal{A}'} q$.*

An important operation on tree automata is to compute the set of all reachable states, where a state q is reachable in $\mathcal{A}$ if there is a tree t such that $t \to^*_{\mathcal{A}} q$. This can be done efficiently (see, e.g., [5]).

Proposition 4. *The set of all reachable states of an ε-NTA $\mathcal{A}$ can be computed in time $\mathcal{O}(|\mathcal{A}|)$.*

Regular Ground Tree Rewriting. Regular ground tree rewriting systems are defined in the same way as GTRS but with regular sets instead of single trees in the rewriting rules. A regular ground tree rewriting system (RGTRS) is a tuple $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$, where A, Σ, and t_{in} are the same as for GTRS and R is a finite set of rewriting rules of the form

$T \overset{\sigma}{\hookrightarrow} T'$ for regular sets $T, T' \subseteq T_A$ of trees. For algorithmic applications we assume that these regular sets are given by NTAs.

A substitution $[x/s']$ is $(\mathcal{R}, \sigma)$-applicable to a tree t if $x \in D_t$ and if there is a rule $T \overset{\sigma}{\hookrightarrow} T' \in R$ with $t^{\downarrow x} \in T$ and $s' \in T'$. With this adapted definition of $(\mathcal{R}, \sigma)$-applicable substitution the definitions of $\mathcal{R}$-applicable, $\to^{\sigma}_{\mathcal{R}}$, $\to_{\mathcal{R}}$, and $\to^{*}_{\mathcal{R}}$ are the same as for GTRS. The tree language that is generated by $\mathcal{R}$ is $T(\mathcal{R}) = \{t \in T_A \mid t_{\text{in}} \to^{*}_{\mathcal{R}} t\}$. We are interested in the graph structure that is naturally induced on $T(\mathcal{R})$ by the rewriting relation. The directed edge labeled graph $G_{\mathcal{R}} = (V_{\mathcal{R}}, E_{\mathcal{R}}, \Sigma)$ generated by $\mathcal{R}$ is defined by $V_{\mathcal{R}} = T(\mathcal{R})$ and $(t, \sigma, t') \in E_{\mathcal{R}}$ if $t \to^{\sigma}_{\mathcal{R}} t'$. Graphs that are isomorphic to $G_{\mathcal{R}}$ for some RGTRS $\mathcal{R}$ are called regular ground tree rewriting graphs or RGTR graphs for short.

Example 2. We define an RGTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ as follows. The ranked alphabet is $A = (A_i)_{i \in [2]}$ with $A_0 = \{a, b\}$, $A_1 = \{c\}$, and $A_2 = \{d\}$, the alphabet for the edge labels is $\Sigma = \{0, 1\}$, and the initial tree is $t_{\text{in}} = d(a, b)$. One rule contains an infinite set of trees on the right-hand side. For this purpose we define the NTA $\mathcal{A} = (\{q_0, q_1\}, A, \Delta, \{q_1\})$ with $\Delta = \{(b, q_0), (q_0, c, q_1), (q_1, c, q_1)\}$. The set $T(\mathcal{A})$ accepted by $\mathcal{A}$ contains all the unary trees of the form $c(\cdots c(b) \cdots)$. The set R of rewriting rules is defined as $R = \{\{a\} \overset{1}{\hookrightarrow} \{c(a)\}, \{b\} \overset{0}{\hookrightarrow} T(\mathcal{A})\}$.

The graph $G_{\mathcal{R}}$ is shown in Figure 2, where the horizontal edges have label 0 and the vertical edges have label 1. Basically, it is the infinite grid, but in each row each vertex has infinitely many edges to the right.

An RGTR graph only consists of the trees reachable from the initial tree, but substitutions can also be applied to trees outside of $T(\mathcal{R})$. This leads to the notion of an

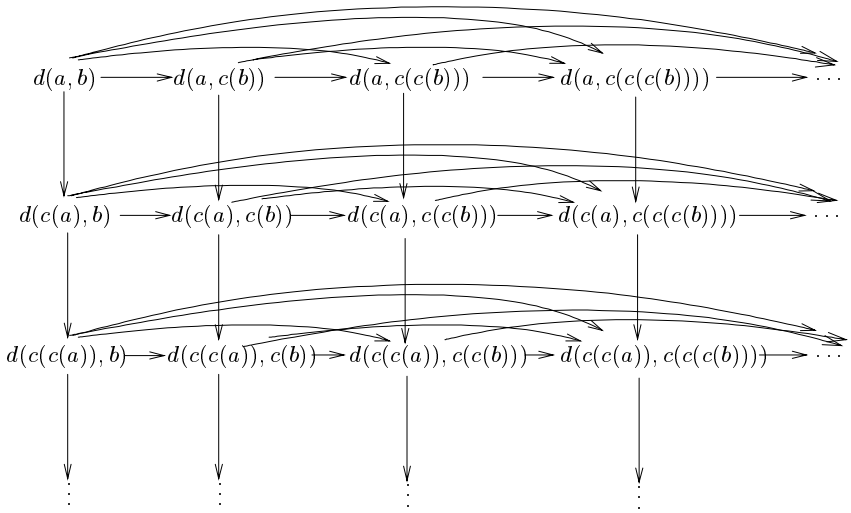


Fig. 2. The graph of the RGTRS from Example 2.

$\mathcal{R}$-path, as defined below. Furthermore, we have to introduce some terminology to refer to the substitutions generating the edges of paths through RGTR graphs.

The graph induced by $\mathcal{R}$ on the set of all trees over A is $\bar{G}_{\mathcal{R}} = (T_A, \bar{E}_{\mathcal{R}}, \Sigma)$ with $(t, \sigma, t') \in \bar{E}_{\mathcal{R}}$ if $t \to^{\sigma}_{\mathcal{R}} t'$. An $\mathcal{R}$-path (or just path if $\mathcal{R}$ is clear from the context) is a path in $\bar{G}_{\mathcal{R}}$, i.e., it is a finite or infinite sequence $t_0 t_1 t_2 \cdots$ of trees such that $t_i \to_{\mathcal{R}} t_{i+1}$ for all t_i, t_{i+1} in this sequence. For each such path there is a sequence of substitutions $[x_0/s_0], [x_1/s_1], [x_2/s_2], \ldots$ such that

— $x_i \in D_{t_i}$ and $t_{i+1} = t_i[x_i/s_i]$,
— $t_i^{\downarrow x_i} \in T$ and $s_i \in T'$ for some rewriting rule $T \overset{\sigma}{\hookrightarrow} T'$ of $\mathcal{R}$.

Such a sequence of substitutions is called an $\mathcal{R}$-derivation of the path $t_0 t_1 t_2 \cdots$ or just derivation if the rewriting system $\mathcal{R}$ is clear from the context. It might happen that there are two different substitutions $[x_i/s_i]$ and $[x_i'/s_i']$ such that $t_{i+1} = t_i[x_i/s_i]$ and $t_{i+1} = t_i[x_i'/s_i']$. Therefore, there may be more than one derivation for a sequence of trees. Sometimes it is desirable to speak of *the* derivation of a sequence of trees. In this case we take the unique derivation $[x_0/s_0], [x_1/s_1], [x_2/s_2], \ldots$ with maximal x_i, i.e., if $t_{i+1} = t_i[x/s]$, then either $x \sqsubseteq x_i$ or there is no rule $T \overset{\sigma}{\hookrightarrow} T'$ with $t_i^{\downarrow x} \in T$ and $s \in T'$. Note that this is just a convention to make things precise in some proofs. It would also be possible to take the minimal x_i but the idea behind this convention is that as few locations of the tree as possible should be involved in the rewritings.

If π is a finite $\mathcal{R}$-path, then $\pi : t \to^*_{\mathcal{R}} t'$ means that π is an $\mathcal{R}$-path from t to t'. For trees t, t' with $t \to^*_{\mathcal{R}} t'$, a derivation of $t \to^*_{\mathcal{R}} t'$ is a derivation of an $\mathcal{R}$-path from t to t'. This derivation is not unique since there may be different $\mathcal{R}$-paths from t to t'.

We need several notations in connection with the rewriting relation:

— For trees t, t' we write $t \to^+_{\mathcal{R}} t'$ if t' is reachable from t with at least one substitution, i.e., if there is a tree t'' such that $t \to_{\mathcal{R}} t'' \to^*_{\mathcal{R}} t'$.
— For sets $T_1, \ldots, T_n$ of trees we write

$$T_1 \to^*_{\mathcal{R}} T_2 \to^*_{\mathcal{R}} \cdots \to^*_{\mathcal{R}} T_n$$

if there are $t_i \in T_i$ for each $i \in \{1, \ldots, n\}$ such that $t_1 \to^*_{\mathcal{R}} t_2 \to^*_{\mathcal{R}} \cdots \to^*_{\mathcal{R}} t_n$. We also use this notation for $\to^+_{\mathcal{R}}$ and combinations of $\to^*_{\mathcal{R}}$ and $\to^+_{\mathcal{R}}$.
— For a tree t and a set T of trees we write $t \to^{\omega}_{\mathcal{R}} T$ if there is an infinite sequence $t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots$ with $t = t_0$ such that infinitely many of the t_i are in T. Furthermore, for sets T_1, T_2 of trees, we write $T_1 \to^{\omega}_{\mathcal{R}} T_2$ if there exists $t \in T_1$ with $t \to^{\omega}_{\mathcal{R}} T_2$.
— If π is an infinite $\mathcal{R}$-path starting in a tree t and visiting a set T of trees infinitely often, then we denote this by $\pi : t \to^{\omega}_{\mathcal{R}} T$.

We also use these notations for single trees instead of sets of trees and write, e.g., $t \to^{\omega}_{\mathcal{R}} t'$ instead of $t \to^{\omega}_{\mathcal{R}} \{t'\}$. Please note that in the above definitions we always ask for the existence of a tree. So $T_1 \to^*_{\mathcal{R}} T_2$ is true if there exists a tree in T_1 from where we can reach T_2. We do not require that from all trees in T_1 we can reach T_2 as the notation might suggest at first glance.

Another relation derived from the rewriting relation is the inverse of the rewriting relation. This relation is useful for the analysis of backward reachability problems. The

inverse $\mathcal{R}^{-1}$ of an RGTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ is obtained from $\mathcal{R}$ by reversing the rules in R, i.e., $\mathcal{R}^{-1} = (A, \Sigma, R^{-1}, t_{\text{in}})$ with $T' \overset{\sigma}{\hookrightarrow} T \in R^{-1}$ iff $T \overset{\sigma}{\hookrightarrow} T' \in R$.

As for automata, we need to define the size $|\mathcal{R}|$ of an RGTRS $\mathcal{R}$ to be able to estimate the complexity of algorithms on RGTRSs. Let $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ be an RGTRS with $R = \{T_1 \overset{\sigma_1}{\hookrightarrow} T_1', \ldots, T_m \overset{\sigma_m}{\hookrightarrow} T_m'\}$ such that $T_i = T(\mathcal{A}_i)$ and $T_i' = T(\mathcal{A}_i')$ for NTAs $\mathcal{A}_i$ and $\mathcal{A}_i'$. We define the size $|\mathcal{R}|$ of $\mathcal{R}$ as

$$|\mathcal{R}| = \sum_{i=1}^{m} (|\mathcal{A}_i| + |\mathcal{A}_i'|).$$

Note that t_{in} is not taken into account in this definition because the algorithms and constructions presented in the next two sections do not depend on it. The initial tree is used to obtain graphs with a designated source vertex to define the model-checking problem in Section 6.

3. Basic Reachability Problems

Our goal is to analyze the model-checking problem for RGTR graphs and temporal logics. More precisely, we analyze the question, what kind of temporal specification logic $\mathcal{L}$ can we use such that the problem of deciding whether $(G_{\mathcal{R}}, t_{\text{in}}) \models \varphi$ can be solved by an algorithm for all formulas φ from $\mathcal{L}$? To answer this question we introduce the following reachability problems that correspond to the semantics of basic temporal operators:

- One step reachability: Given an RGTRS $\mathcal{R}$, a vertex t, and a set T of vertices of $G_{\mathcal{R}}$, does there exist a successor of t that is in T?
- Reachability: Given an RGTRS $\mathcal{R}$, a vertex t, and a set T of vertices of $G_{\mathcal{R}}$, does there exist a path from t to a vertex in T?
- Constrained reachability: Given an RGTRS $\mathcal{R}$, a vertex t, and sets T_1, T_2 of vertices of $G_{\mathcal{R}}$, does there exist a path from t that remains in T_2 until it eventually reaches a vertex in T_1?
- Recurrence: Given an RGTRS $\mathcal{R}$, a vertex t, and a set T of vertices of $G_{\mathcal{R}}$, does there exist a path from t that infinitely often visits T?

All of these questions can also be posed in their "universal versions." These versions do not ask for the existence of a path but whether all paths have the respective property:

- Universal one step reachability: Given an RGTRS $\mathcal{R}$, a vertex t, and a set T of vertices of $G_{\mathcal{R}}$, are all successors of t in T?
- Universal reachability: Given an RGTRS $\mathcal{R}$, a vertex t, and a set T of vertices of $G_{\mathcal{R}}$, do all paths from t eventually reach a vertex in T?
- Universal constrained reachability: Given an RGTRS $\mathcal{R}$, a vertex t, and sets T_1, T_2 of vertices of $G_{\mathcal{R}}$, do all paths from t remain in T_2 until they eventually reach a vertex from T_1?
- Universal recurrence: Given an RGTRS $\mathcal{R}$, a vertex t, and a set T of vertices of $G_{\mathcal{R}}$, do all infinite paths from t infinitely often visit T?

Table 1. Overview of decidability results.

Problem	Temporal operator	
One step reachability	EX	Decidable
Reachability	EF	Decidable
Constrained reachability	EU	Undecidable
Recurrence	EGF	Decidable
Universal one step reachability	AX	Decidable
Universal reachability	AF	Undecidable
Universal constrained reachability	AU	Undecidable
Universal recurrence	AGF	Undecidable

For the sets of vertices used in the specification of the problems we use regular sets of trees. This allows us to use finitely represented infinite sets, which can serve as input for an algorithm.

One should note that universal one step reachability can be expressed as the negation of one step reachability: all successors of t are in T if no successor of t is in the complement of T. This relation is not valid for the other problems. Furthermore, it should be mentioned that (universal) reachability is a special case of (universal) constrained reachability by setting the set T_2 to the set of all trees. The problem of universal reachability is shown to be undecidable in Section 5. Therefore, it is clear that universal constrained reachability is also undecidable.

Table 1 gives an overview of the results from the next two sections and also lists for each problem the corresponding temporal operator in CTL* (see [10]) notation.

One step reachability and reachability are already known to be decidable. These results can be found in [5] in the context of closure properties of ground tree transducers. Since we need these results in Section 6 we state them in the following without proofs. For a detailed description of the algorithms used to obtain the desired complexities, we refer the reader to [18].

In correspondence to the problems of one step reachability and reachability we define the following sets for an RGTRS $\mathcal{R}$ and a set T of trees:

$$\mathrm{pre}_{\mathcal{R}}(T) := \{t \in T_A \mid t \to_{\mathcal{R}} T\}, \qquad \mathrm{pre}^*_{\mathcal{R}}(T) := \{t \in T_A \mid t \to^*_{\mathcal{R}} T\},$$
$$\mathrm{post}_{\mathcal{R}}(T) := \{t \in T_A \mid T \to_{\mathcal{R}} t\}, \qquad \mathrm{post}^*_{\mathcal{R}}(T) := \{t \in T_A \mid T \to^*_{\mathcal{R}} t\}.$$

The subsequent results state that it is always possible to construct ε-NTAs recognizing these sets if the set T is given as an ε-NTA.

Theorem 1. *Let $\mathcal{R} = (A, \Sigma, R, t_{\mathrm{in}})$ be an RGTRS and let $\mathcal{A} = (Q, A, \Delta, F)$ be an ε-NTA. One can construct an ε-NTA $\mathcal{A}_{\mathrm{pre}_{\mathcal{R}}}$ of size $\mathcal{O}(|\mathcal{R}| \cdot |\mathcal{A}|)$ accepting the set $\mathrm{pre}_{\mathcal{R}}(T(\mathcal{A}))$.*

The regularity of $\mathrm{post}_{\mathcal{R}}(T(\mathcal{A}))$ is a direct consequence of Theorem 1 and the equality $\mathrm{post}_{\mathcal{R}}(T) = \mathrm{pre}_{\mathcal{R}^{-1}}(T)$.

Corollary 1. *Let $\mathcal{R} = (A, \Sigma, R, t_{\mathrm{in}})$ be an RGTRS and let $\mathcal{A} = (Q, A, \Delta, F)$ be an ε-NTA. One can construct an ε-NTA $\mathcal{A}_{\mathrm{post}_{\mathcal{R}}}$ of size $\mathcal{O}(|\mathcal{R}| \cdot |\mathcal{A}|)$ accepting the set $\mathrm{post}_{\mathcal{R}}(T(\mathcal{A}))$.*

For the reachability problem we have similar properties as for one step reachability.

Theorem 2. *Let $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ be an RGTRS and let $\mathcal{A} = (Q, A, \Delta, F)$ be an ε-NTA. One can construct an ε-NTA $\mathcal{A}_{\text{pre}^*_\mathcal{R}}$ of size $\mathcal{O}(|\mathcal{R}| \cdot (|\mathcal{A}| + |\mathcal{R}|))$ such that*

(i) *$\mathcal{A}_{\text{pre}^*_\mathcal{R}} = (Q \cup Q', A, \Delta \cup \Delta', F)$ for some Q' and Δ', and*
(ii) *$t \in T(\mathcal{A}_{\text{pre}^*_\mathcal{R}}(q))$ iff $t \to^*_\mathcal{R} T(\mathcal{A}(q))$ for each $q \in Q$.*

Note that (ii) of the above theorem implies that $T(\mathcal{A}_{\text{pre}^*_\mathcal{R}}) = \text{pre}^*_\mathcal{R}(T(\mathcal{A}))$. The time complexity for constructing $\mathcal{A}_{\text{pre}^*_\mathcal{R}}$ is given in the following theorem, a proof of which can be found in [18].

Theorem 3. *The automaton $\mathcal{A}_{\text{pre}^*_\mathcal{R}}$ can be constructed in time $O(|\mathcal{R}|^2 \cdot (|\mathcal{A}| + |\mathcal{R}|))$.*

As for one step reachability one can also apply Theorem 2 to the inverse rewriting system to obtain an automaton $\mathcal{A}_{\text{post}^*_\mathcal{R}}$ for the set $\text{post}^*_\mathcal{R}(T(\mathcal{A}))$.

As we will see later, for the study of the recurrence problem we have to deal with iterated reachability problems of the form $T(\mathcal{A}) \to^*_\mathcal{R} T(\mathcal{B}) \to^*_\mathcal{R} T(\mathcal{C})$. More precisely, we have to compute the set of all tuples (p, q, r), where p, q, r are states of $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, such that $T(\mathcal{A}(p)) \to^*_\mathcal{R} T(\mathcal{B}(q)) \to^*_\mathcal{R} T(\mathcal{C}(r))$. Using the fact that $T(\mathcal{A}(p)) \to^*_\mathcal{R} T(\mathcal{B}(q)) \to^*_\mathcal{R} T(\mathcal{C}(r))$ iff there is a tree $t \in T(\mathcal{B}(q))$ such that $t \in \text{post}^*_\mathcal{R}(T(\mathcal{A}(p)))$ and $t \in \text{pre}^*_\mathcal{R}(T(\mathcal{C}(r)))$, nested computations of $\text{pre}^*_\mathcal{R}$ can be avoided.

Lemma 1. *Let p, q, r be states of ε-NTAs $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, respectively. Then (p, q, r) is reachable in $\mathcal{A}_{\text{post}^*_\mathcal{R}} \times \mathcal{B} \times \mathcal{C}_{\text{pre}^*_\mathcal{R}}$ iff $T(\mathcal{A}(p)) \to^*_\mathcal{R} T(\mathcal{B}(q)) \to^*_\mathcal{R} T(\mathcal{C}(r))$.*

Proof. The state (p, q, r) is reachable in $\mathcal{A}_{\text{post}^*_\mathcal{R}} \times \mathcal{B} \times \mathcal{C}_{\text{pre}^*_\mathcal{R}}$ iff there is $t_2 \in T_A$ such that $t_2 \in T(\mathcal{A}_{\text{post}^*_\mathcal{R}}(p))$, $t_2 \in T(\mathcal{B}(q))$, and $t_2 \in T(\mathcal{C}_{\text{pre}^*_\mathcal{R}}(r))$. According to Theorem 2, this holds iff there are $t_1 \in T(\mathcal{A}(p))$ and $t_3 \in T(\mathcal{C}(r))$ such that $t_1 \to^*_\mathcal{R} t_2 \to^*_\mathcal{R} t_3$. □

Hence, we get the following complexity.

Lemma 2. *Given ε-NTAs $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ one can compute the set of all (p, q, r) with $T(\mathcal{A}(p)) \to^*_\mathcal{R} T(\mathcal{B}(q)) \to^*_\mathcal{R} T(\mathcal{C}(r))$ in time $\mathcal{O}(|\mathcal{R}|^2|\mathcal{B}|(|\mathcal{A}| + |\mathcal{R}|)(|\mathcal{C}| + |\mathcal{R}|))$.*

Proof. By Lemma 1 it is sufficient to compute the set of reachable states in $\mathcal{D} = \mathcal{A}_{\text{post}^*_\mathcal{R}} \times \mathcal{B} \times \mathcal{C}_{\text{pre}^*_\mathcal{R}}$, which can be done in linear time in the size of $\mathcal{D}$ (Proposition 4). The size of $\mathcal{D}$ is

$$|\mathcal{D}| = |\mathcal{A}_{\text{post}^*_\mathcal{R}}| \cdot |\mathcal{B}| \cdot |\mathcal{C}_{\text{pre}^*_\mathcal{R}}| \in \mathcal{O}(|\mathcal{R}| \cdot (|\mathcal{A}| + |\mathcal{R}|) \cdot |\mathcal{B}| \cdot |\mathcal{R}| \cdot (|\mathcal{C}| + |\mathcal{R}|)).$$

Constructing $\mathcal{A}_{\text{post}^*_\mathcal{R}}$ takes time $\mathcal{O}(|\mathcal{R}|^2(|\mathcal{A}| + |\mathcal{R}|))$ and constructing $\mathcal{C}_{\text{pre}^*_\mathcal{R}}$ takes time $\mathcal{O}(|\mathcal{R}|^2(|\mathcal{C}| + |\mathcal{R}|))$. Therefore, the total time is in $\mathcal{O}(|\mathcal{R}|^2|\mathcal{B}|(|\mathcal{A}| + |\mathcal{R}|)(|\mathcal{C}| + |\mathcal{R}|))$. □

4. Decidability of the Recurrence Problem

In this section we develop an algorithm to compute for an NTA $\mathcal{A}$ and an RGTRS $\mathcal{R}$ an ε-NTA accepting the set of all trees t with $t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$. A similar algorithm was already published in [17] for GTRSs but without detailed correctness proofs.

Since RGTR graphs are infinite, in general, a path that visits a set $T(\mathcal{A})$ of vertices infinitely often either visits a single vertex from this set infinitely often, or it visits infinitely many different vertices from $T(\mathcal{A})$. To distinguish these two cases we make the following definition for an RGTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$, $t \in T_A$, and a set T of trees:

— $t \to_{\mathcal{R}}^{\omega} T$ with loop if there is $t' \in T$ such that $t \to_{\mathcal{R}}^{\omega} t'$.
— $t \to_{\mathcal{R}}^{\omega} T$ without loop if $t \to_{\mathcal{R}}^{\omega} T$ and not $t \to_{\mathcal{R}}^{\omega} T$ with loop.

In the analysis carried out in this section we will distinguish these two cases (recurrence with or without loop). We mention that the algorithm developed for recurrence without loop can also be applied to solve the case of recurrence with loop. As this algorithm and the proof of its correctness are rather involved, we have chosen to treat recurrence with loop separately (in a different way) to avoid additional case distinctions and proofs in the development of the algorithm.

For the remainder of this section we fix an NTA $\mathcal{A} = (Q, A, \Delta, F)$ and an RGTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ with $R = \{T_1 \overset{\sigma_1}{\hookrightarrow} T_1', \ldots, T_m \overset{\sigma_m}{\hookrightarrow} T_m'\}$, where the sets T_i, T_i' are accepted by NTAs $\mathcal{A}_i = (Q_i, A, \Delta_i, F_i)$ and $\mathcal{A}_i' = (Q_i', A, \Delta_i', F_i')$, respectively.

To obtain the algorithm we proceed in two steps. In the first step both cases (recurrence with and without loop) are reduced to instances of the reachability problem. This works as follows: Assume that for some $i \in \{1, \ldots, m\}$ and $q \in Q$ we have $T_i' \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$. Let $t \in T_A$ and assume that a tree s is reachable from t in $\mathcal{R}$ such that there is a location $x \in D_s$ with $s^{\downarrow x} \in T_i$ and $s[x/q] \to_{\mathcal{A}}^{*} F$. Then we can replace this subtree of s at x by an appropriate tree $t' \in T_i'$ with $t' \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$. Since $s[x/q] \to_{\mathcal{A}}^{*} F$ we get $t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$.

On the other hand, we also show the converse, i.e., if $t \in T_A$ with $t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$, then there are $i \in \{1, \ldots, m\}$ and $q \in Q$ with $T_i' \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$, and from t a tree s with the above properties is reachable.

Hence, we have reduced the problem to a reachability problem. The difficulty is that we used the condition $T_i' \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$ in the definition of the set of trees that has to be reached. This problem is shown to be decidable in the second step.

Step 1: *Reduction to Reachability.* We start with an elementary technical lemma using that substitutions that are not "synchronized" via a substitution at a common ancestor are independent.

Lemma 3. *Let $t_0, t_1, \ldots, t_n \in T_A$ such that $t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_n$ with derivation $[x_0/s_0], \ldots, [x_{n-1}/s_{n-1}]$. If $x \in D_{t_0}$ is such that $x_j \not\sqsubseteq x$ for all $j \in \{0, \ldots, n-1\}$, then*

(i) *$t_0 \to_{\mathcal{R}}^{*} t_n[x/t_0^{\downarrow x}]$ and*
(ii) *for $0 \le j \le n-1$, $t_j^{\downarrow x} \to_{\mathcal{R}} t_{j+1}^{\downarrow x}$ if $x \sqsubseteq x_j$, and $t_j^{\downarrow x} = t_{j+1}^{\downarrow x}$ otherwise.*

Proof. The idea for (i) is to omit all substitutions $[x_j/s_j]$ with $x \sqsubseteq x_j$.

The formal proof is an induction on n. For $n = 0$ claims (i) and (ii) obviously hold. If $n \geq 1$, then $t_0 \to_{\mathcal{R}}^* t_{n-1}[x/t_0^{\downarrow x}]$ by the inductive hypothesis. If $x \sqsubseteq x_{n-1}$, then $t_{n-1}[x/t_0^{\downarrow x}] = t_n[x/t_0^{\downarrow x}]$ and hence (i) holds. If $x \not\sqsubseteq x_{n-1}$, then x and x_{n-1} are incomparable and the substitution $[x_{n-1}/s_{n-1}]$ can also be applied to $t_{n-1}[x/t_0^{\downarrow x}]$, yielding $t_n[x/t_0^{\downarrow x}]$.

For (ii) we get as inductive hypothesis $t_j^{\downarrow x} \to_{\mathcal{R}} t_{j+1}^{\downarrow x}$ if $x \sqsubseteq x_j$, and $t_j^{\downarrow x} = t_{j+1}^{\downarrow x}$ otherwise, for all $0 \leq j \leq n-2$. Again we distinguish two cases. If $x \sqsubseteq x_{n-1}$, then obviously $t_{n-1}^{\downarrow x} \to_{\mathcal{R}} t_n^{\downarrow x}$. If $x \not\sqsubseteq x_{n-1}$, then x and x_{n-1} are incomparable. Hence, the substitution $[x_{n-1}/s_{n-1}]$ does not affect the subtree at x and therefore $t_{n-1}^{\downarrow x} = t_n^{\downarrow x}$. □

For the reduction to the reachability problem we define the following two sets:

$$\mathrm{Rec}_1(\mathcal{R}, \mathcal{A}) = \left\{ t \in T_A \;\middle|\; \begin{array}{c} \exists x \in D_t, q \in Q, i \in \{1, \ldots, m\} \,:\, t^{\downarrow x} \in T_i, \\ T_i' \to_{\mathcal{R}}^* T(\mathcal{A}(q)) \to_{\mathcal{R}}^* T_i, t[x/q] \to_{\mathcal{A}}^* F \end{array} \right\},$$

$$\mathrm{Rec}_2(\mathcal{R}, \mathcal{A}) = \left\{ t \in T_A \;\middle|\; \begin{array}{c} \exists x \in D_t, q \in Q, i \in \{1, \ldots, m\} \,:\, t^{\downarrow x} \in T_i, \\ T_i' \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q)) \text{ without loop}, t[x/q] \to_{\mathcal{A}}^* F \end{array} \right\}.$$

The following lemma shows that an automaton for $\{t \in T_A \mid t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})\}$ can be constructed by first constructing an automaton for the union of $\mathrm{Rec}_1(\mathcal{R}, \mathcal{A})$ and $\mathrm{Rec}_2(\mathcal{R}, \mathcal{A})$ and then applying Theorem 2. The conditions for $\mathrm{Rec}_1(\mathcal{R}, \mathcal{A})$ can be checked by an ε-NTA since the problem $T_i' \to_{\mathcal{R}}^* T(\mathcal{A}(q)) \to_{\mathcal{R}}^* T_i$ is decidable for each pair of $i \in \{1, \ldots, m\}$ and $q \in Q$ by Lemma 2. Similarly, we can construct an automaton for $\mathrm{Rec}_2(\mathcal{R}, \mathcal{A})$ if we can decide the problem $T_i' \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$ without loop for each pair of $i \in \{1, \ldots, m\}$ and $q \in Q$.

Lemma 4. *Let $t \in T_A$.*

(i) $t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$ *with loop iff* $t \in \mathrm{pre}_{\mathcal{R}}^*(\mathrm{Rec}_1(\mathcal{R}, \mathcal{A}))$.
(ii) $t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$ *without loop iff* $t \in \mathrm{pre}_{\mathcal{R}}^*(\mathrm{Rec}_2(\mathcal{R}, \mathcal{A})) \setminus \mathrm{pre}_{\mathcal{R}}^*(\mathrm{Rec}_1(\mathcal{R}, \mathcal{A}))$.

Proof. (i) ($\Rightarrow$) If $t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$ with loop, then there are $t_0, \ldots, t_n \in T_A$ with $n \in \mathbb{N}$ such that $t_0 \in T(\mathcal{A})$, $t \to_{\mathcal{R}}^* t_0$, and

$$t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_n \to_{\mathcal{R}} t_0$$

with derivation $[x_0/s_0], \ldots, [x_n/s_n]$. We choose $j \in \{0, \ldots, n\}$ such that x_j is the minimal location from $\{x_0, \ldots, x_n\}$, with respect to $\sqsubseteq$. This minimality implies that $x_j \in D_{t_l}$ for all $l \in \{0, \ldots, n\}$ because $x_j \in D_{t_j}$ and no substitutions are made above x_j. Let $i \in \{1, \ldots, m\}$ be such that $t_j^{\downarrow x_j} \in T_i$ and $s_j \in T_i'$, and let $q \in Q$ with $t_0 \to_{\mathcal{A}}^* t_0[x_j/q] \to_{\mathcal{A}}^* F$. Define the tree $t' = t_0[x_j/t_j^{\downarrow x_j}]$. By applying Lemma 3(i) to the sequence $t_j \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_0$ we know $t_j \to_{\mathcal{R}}^* t'$ and thus $t \to_{\mathcal{R}}^* t'$.

To show that $t' \in \mathrm{Rec}_1(\mathcal{R}, \mathcal{A})$, and hence $t \in \mathrm{pre}_{\mathcal{R}}^*(\mathrm{Rec}_1(\mathcal{R}, \mathcal{A}))$, first note that, by the definitions of t', i, and q, we have already established $(t')^{\downarrow x_j} \in T_i$ and $t'[x_j/q] \to_{\mathcal{A}}^* F$. Furthermore, since $x_l \not\sqsubseteq x_j$ for all $l \in \{0, \ldots, n\}$, we know that $t_j^{\downarrow x_j} \to_{\mathcal{R}} t_{j+1}^{\downarrow x_j} \to_{\mathcal{R}}^* t_0^{\downarrow x_j} \to_{\mathcal{R}}^* t_j^{\downarrow x_j}$ by Lemma 3(ii) applied to $t_j \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_0 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_j$. Hence, we

get $T_i' \to_{\mathcal{R}}^* T(\mathcal{A}(q)) \to_{\mathcal{R}}^* T_i$ because $t_{j+1}^{\downarrow x_j} = s_j \in T_i'$, $t_0^{\downarrow x_j} \in T(\mathcal{A}(q))$, and $t_j^{\downarrow x_j} \in T_i$. Therefore, all conditions are satisfied and $t' \in \mathrm{Rec}_1(\mathcal{R}, \mathcal{A})$.

($\Leftarrow$) Assume that $t \to_{\mathcal{R}}^* t' \in \mathrm{Rec}_1(\mathcal{R}, \mathcal{A})$ and let $x \in D_{t'}$, $q \in Q$, $i \in \{1, \ldots, m\}$ be such that $(t')^{\downarrow x} \in T_i$, $T_i' \to_{\mathcal{R}}^* T(\mathcal{A}(q)) \to_{\mathcal{R}}^* T_i$, and $t'[x/q] \to_{\mathcal{A}}^* F$. Furthermore, let $s \in T_i$, $s' \in T_i'$, and $s'' \in T(\mathcal{A}(q))$ be such that $s' \to_{\mathcal{R}}^* s'' \to_{\mathcal{R}}^* s$. From $t'[x/q] \to_{\mathcal{A}}^* F$ and $s'' \to_{\mathcal{A}}^* q$ we get $t'[x/s''] \to_{\mathcal{A}}^* F$. Then $t \to_{\mathcal{R}}^\omega T(\mathcal{A})$ with loop as follows:

$$t \to_{\mathcal{R}}^* t' \to_{\mathcal{R}} t'[x/s'] \to_{\mathcal{R}}^* \underbrace{t'[x/s'']}_{\in T(\mathcal{A})} \to_{\mathcal{R}}^* t'[x/s] \to_{\mathcal{R}} t'[x/s'].$$

(ii) ($\Rightarrow$) For $t_0 \to_{\mathcal{R}}^\omega T(\mathcal{A})$ without loop let $t_1, t_2, \ldots$ be an infinite sequence of trees such that

$$t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} t_2 \cdots$$

with derivation $[x_0/s_0], [x_1/s_1], \ldots$ and $t_l \in T(\mathcal{A})$ for infinitely many $l \in \mathbb{N}$. First note that there are only finitely many minimal locations in the derivation, with respect to $\sqsubseteq$. This is because all minimal locations must be in D_{t_0}. Among these finitely many minimal locations from the derivation there must be at least one location x such that infinitely many substitutions are made below x. Therefore, we can choose $j \in \mathbb{N}$ such that $x_l \not\sqsubset x_j$ for all l and $x_j \sqsubseteq x_l$ for infinitely many l. Note that $x_l \not\sqsubset x_j$ for all l implies that $x_j \in D_{t_l}$ for all l. Since infinitely many trees along the sequence are accepted by $\mathcal{A}$ and since $\mathcal{A}$ has finitely many states, there is $q \in Q$ such that $t_l \to_{\mathcal{A}}^* t_l[x_j/q] \to_{\mathcal{A}}^* F$ for infinitely many l.

Let $i \in \{1, \ldots, m\}$ with $t_j^{\downarrow x_j} \in T_i$ and $s_j \in T_i'$, and let $k > j$ be such that $t_k \to_{\mathcal{A}}^* t_k[x_j/q] \to_{\mathcal{A}}^* F$. Similar to the proof of (i) we define $t' = t_k[x_j/t_j^{\downarrow x_j}]$ and get $t_0 \to_{\mathcal{R}}^* t'$ by Lemma 3(i) applied to $t_j \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_k$. Furthermore, we have $(t')^{\downarrow x_j} = t_j^{\downarrow x_j} \in T_i$ and $t'[x_j/q] = t_k[x_j/q] \to_{\mathcal{A}}^* F$.

It remains to show that $T_i' \to_{\mathcal{R}}^\omega T(\mathcal{A}(q))$ without loop. Since $x_j \sqsubseteq x_l$ for infinitely many l, and $t_l \to_{\mathcal{A}}^* t_l[x_j/q] \to_{\mathcal{A}}^* F$ for infinitely many l, we can choose a strictly increasing sequence $l_1, l_2, l_3, \ldots$ such that for all m, $t_{l_m} \to_{\mathcal{A}}^* t_{l_m}[x_j/q] \to_{\mathcal{A}}^* F$ and there exists $l_m \le l_m' \le l_{m+1}$ with $x_j \sqsubseteq x_{l_m'}$.

Since $s_j \in T_i'$, we can conclude that $T_i' \to_{\mathcal{R}}^\omega T(\mathcal{A}(q))$ because $s_j = t_{j+1}^{\downarrow x_j} \to_{\mathcal{R}}^+ t_{l_1}^{\downarrow x_j} \to_{\mathcal{R}}^+ t_{l_2}^{\downarrow x_j} \to_{\mathcal{R}}^+ \cdots$ by Lemma 3(ii) and the choice of l_m. Assume that $T_i' \to_{\mathcal{R}}^\omega T(\mathcal{A}(q))$ with loop. Then there is $s' \in T_i'$ and $s'' \in T(\mathcal{A}(q))$ such that $s' \to_{\mathcal{R}}^\omega s''$. Hence, $t'[x_j/s'] \to_{\mathcal{R}}^\omega t'[x_j/s'']$. From $s'' \in T(\mathcal{A}(q))$ and $t'[x_j/q] \to_{\mathcal{A}}^* F$ we get $t'[x_j/s''] \in T(\mathcal{A})$ and thus $t'[x_j/s'] \to_{\mathcal{R}}^\omega T(\mathcal{A})$ with loop. From $t_0 \to_{\mathcal{R}}^* t'$, $(t')^{\downarrow x_j} \in T_i$, and $s' \in T_i'$ we get $t_0 \to_{\mathcal{R}}^* t'[x_j/s']$ and therefore $t_0 \to_{\mathcal{R}}^\omega T(\mathcal{A})$ with loop, contradicting the assumption that $t_0 \to_{\mathcal{R}}^\omega T(\mathcal{A})$ without loop.

($\Leftarrow$) If $t \in \mathrm{pre}_{\mathcal{R}}^*(\mathrm{Rec}_2(\mathcal{R}, \mathcal{A})) \backslash \mathrm{pre}_{\mathcal{R}}^*(\mathrm{Rec}_1(\mathcal{R}, \mathcal{A}))$, then we know from (i) that not $t \to_{\mathcal{R}}^\omega T(\mathcal{A})$ with loop. Let $t' \in \mathrm{Rec}_2(\mathcal{R}, \mathcal{A})$ with $t \to_{\mathcal{R}}^* t'$. Let $x \in D_{t'}$, $q \in Q$, $i \in \{1, \ldots, m\}$, and $s' \in T_i'$ be such that $(t')^{\downarrow x} \in T_i$, $s' \to_{\mathcal{R}}^\omega T(\mathcal{A}(q))$ without loop, and $t'[x/q] \to_{\mathcal{A}}^* F$. Then obviously $t' \to_{\mathcal{R}} t'[x/s'] \to_{\mathcal{R}}^\omega T(\mathcal{A})$ and therefore $t \to_{\mathcal{R}}^\omega T(\mathcal{A})$ without loop. □

A simple consequence of the previous lemma is

Lemma 5. *For $t \in T_A$, $t \to^{\omega}_{\mathcal{R}} T(\mathcal{A})$ iff $t \in \mathrm{pre}^*_{\mathcal{R}}(\mathrm{Rec}_1(\mathcal{R}, \mathcal{A}) \cup \mathrm{Rec}_2(\mathcal{R}, \mathcal{A}))$.*

In what follows we develop a procedure to decide for $i \in \{1, \ldots, m\}$ and $q \in Q$ if $T'_i \to^{\omega}_{\mathcal{R}} T(\mathcal{A}(q))$ without loop. This, in turn, enables us to construct an ε-NTA for $\mathrm{Rec}_1(\mathcal{R}, \mathcal{A}) \cup \mathrm{Rec}_2(\mathcal{R}, \mathcal{A})$.

Step 2: *Recurrence without Loop*. We start by a general analysis of sequences of trees that infinitely often visit a regular set of trees. So, in the beginning we abstract from the more specific question "$T'_i \to^{\omega}_{\mathcal{R}} T(\mathcal{A}(q))$ without loop" and study general sequences of trees starting from some tree t and visiting some regular set $T(\mathcal{A})$ infinitely often without loop. The goal is to gather some information on how the trees in such sequences and the corresponding accepting runs of $\mathcal{A}$ evolve. Later, we use this information to solve our original problem "$T'_i \to^{\omega}_{\mathcal{R}} T(\mathcal{A}(q))$ without loop". A first tool is the limit of a sequence of trees. It contains all the locations of the trees in this sequence that eventually stay fixed.

To simplify notation, a path π always means an $\mathcal{R}$-path π for the RGTRS $\mathcal{R}$ fixed at the beginning of this section. Let $[x_0/s_0], [x_1/s_1], \ldots$ be the derivation of an infinite path π starting with a tree $t \in T_A$. A location $x \in \mathbb{N}^*$ is called stable on π if it is present in all the trees from π, and if it is never involved in any of the substitutions. More formally, $x \in \mathbb{N}^*$ is stable on π if $x \in D_{\pi(i)}$ and $x_i \not\sqsubseteq x$ for all $i \in \mathbb{N}$. Recall that $\pi(i)$ denotes the ith vertex on the path π. Since the vertices of an $\mathcal{R}$-path are trees, $\pi(i)$ denotes the ith tree on π.

The limit $\lim(\pi)$ of π is defined as

$$\lim(\pi) = \{x \in \mathbb{N}^* \mid \exists i \in \mathbb{N} \ : \ x \text{ stable on } \pi[i, \infty)\}.$$

If x is stable on $\pi[i, \infty)$ for some $i \in \mathbb{N}$, then, by the definition of stable, each prefix y of x is stable on $\pi[i, \infty)$, too. Hence, $\lim(\pi)$ is prefix closed. Furthermore, note that x being stable on $\pi[i, \infty)$ for some $i \in \mathbb{N}$ also implies that x is stable on $\pi[j, \infty)$ for all $j > i$. We use this fact implicitly in several proofs.

For a location $x \notin \lim(\pi)$ there are two possibilities: either x is involved in infinitely many substitutions or x only occurs in finitely many trees on π. We are interested in paths where the latter case holds for all locations that are not in the limit of the path. An infinite path π is called stable if for all $x \notin \lim(\pi)$ there exists an $i \in \mathbb{N}$ such that $x \notin D_{\pi(j)}$ for all $j \geq i$. Informally speaking, π is stable if there is no location that is involved in infinitely many substitutions on π. This is made precise by the following lemma.

Lemma 6. *For an infinite path π with derivation $[x_0/s_0], [x_1/s_1], \ldots$ the following statements are equivalent.*

(i) *The path π is stable.*
(ii) *For all $x \in \mathbb{N}^*$ the set $\{j \in \mathbb{N} \mid x_j = x\}$ is finite.*
(iii) *For all $x \in \mathbb{N}^*$ the set $\{j \in \mathbb{N} \mid x_j \sqsubseteq x\}$ is finite.*

Proof. Assume that π is stable and let $x \in \mathbb{N}^*$. If $x \in \lim(\pi)$, then there is $i \in \mathbb{N}$ such that x is stable on $\pi[i, \infty)$, by definition of $\lim(\pi)$. This implies that $x_j \not\sqsubseteq x$ for all $j \geq i$ and hence the set $\{j \in \mathbb{N} \mid x_j = x\}$ is finite since x is a prefix of itself. If $x \notin \lim(\pi)$, then, since π is stable, there exists $i \in \mathbb{N}$ such that $x \notin D_{\pi(j)}$ for all $j \geq i$. Thus, $x_j \neq x$ for all $j \geq i$ and the set $\{j \in \mathbb{N} \mid x_j = x\}$ is finite. Thus, we have shown the implication (i) $\Rightarrow$ (ii).

The implication (ii) $\Rightarrow$ (iii) follows from the fact that every location has only finitely many prefixes. So, if $y \in \mathbb{N}^*$ and $\{j \in \mathbb{N} \mid x_j = x\}$ is finite for all prefixes x of y, then $\{j \in \mathbb{N} \mid x_j \sqsubseteq y\}$ is finite, too.

It remains to show the implication (iii) $\Rightarrow$ (i). Assume that the set $\{j \in \mathbb{N} \mid x_j \sqsubseteq x\}$ is finite for all $x \in \mathbb{N}^*$ and let $y \notin \lim(\pi)$. To show that π is stable we have to show that there is $i \in \mathbb{N}$ such that $y \notin D_{\pi(j)}$ for all $j \geq i$. Since $\{j \in \mathbb{N} \mid x_j \sqsubseteq y\}$ is finite, there exists $i \in \mathbb{N}$ such that $x_j \not\sqsubseteq y$ for all $j \geq i$. This obviously implies that either $y \in D_{\pi(j)}$ for all $j \geq i$ or $y \notin D_{\pi(j)}$ for all $j \geq i$. In the first case y would be stable on $\pi[i, \infty)$ and so $y \in \lim(\pi)$. Hence the second case holds. □

The following lemma establishes a first connection between the recurrence problem and stable paths.

Lemma 7. *If $t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$ without loop and if π is an infinite path with $\pi : t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$, then π is stable.*

Proof. Let $\pi = t_0 t_1 t_2 \cdots$ be an infinite path with $\pi : t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$, and let $[x_0/s_0]$, $[x_1/s_1], \ldots$ be the derivation of π. Let $X = \{x \in \mathbb{N}^* \mid x = x_i \text{ for infinitely many } i\}$ be the set of locations that occur infinitely often in the sequence $x_0, x_1, \ldots$ of locations from the derivation. From Lemma 6 we know that π is stable if, and only if, X is empty. In the case that X is not empty we show that $t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$ with loop. By the assumption that $t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$ without loop we get a contradiction.

Assume that $X \neq \emptyset$ and let x be a minimal element of X with respect to $\sqsubseteq$. By the minimality of x there exists an index $k \in \mathbb{N}$ such that $x_j \not\sqsubset x$ for all $j \geq k$. Since x equals infinitely many of the x_j there must be $i \in \{1, \ldots, m\}$ such that $x = x_j$, $t_j^{\downarrow x} \in T_i$ and $s_j \in T_i'$ for infinitely many j. Then we can find j, j_1, j_2 with $k \leq j_1 < j < j_2$ such that $x = x_{j_1} = x_{j_2}$, $t_{j_1}^{\downarrow x}, t_{j_2}^{\downarrow x} \in T_i$, $s_{j_1}, s_{j_2} \in T_i'$, and $t_j \in T(\mathcal{A})$.

From Lemma 3(i) we can conclude that $t = t_0 \to_{\mathcal{R}}^* t_j[x/s_{j_1}]$, and from Lemma 3(ii) we know that $s_{j_1} \to_{\mathcal{R}}^* t_j^{\downarrow x} \to_{\mathcal{R}}^* t_{j_2}^{\downarrow x}$. Furthermore, $t_{j_2}^{\downarrow x} \to_{\mathcal{R}} s_{j_1}$ because $t_{j_2}^{\downarrow x} \in T_i$ and $s_{j_1} \in T_i'$. Therefore, we get

$$t \to_{\mathcal{R}}^* t_j[x/s_{j_1}] \to_{\mathcal{R}}^* t_j[x/t_j^{\downarrow x}] \to_{\mathcal{R}}^* t_j[x/t_{j_2}^{\downarrow x}] \to_{\mathcal{R}} t_j[x/s_{j_1}].$$

Since $t_j[x/t_j^{\downarrow x}] = t_j \in T(\mathcal{A})$ we get $t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})$ with loop. □

If an infinite path π is stable, then there is no location involved in infinitely many substitutions. Hence, the domain of the trees on π must grow. This is formalized in the following lemma.

Lemma 8. *If an infinite path π is stable, then* $\lim(\pi)$ *is infinite.*

Proof. Let π be an infinite path with derivation $[x_0/s_0], [x_1/s_1], \ldots$ such that $\lim(\pi)$ is finite. We have to show that π is not stable. Since $\lim(\pi)$ is finite, we can choose $i \in \mathbb{N}$ such that x is stable on $\pi[i, \infty)$ for all $x \in \lim(\pi)$. Let $j \geq i$ be such that $x_k \not\sqsubset x_j$ for all $k \geq i$, i.e., x_j is a minimal location in the derivation starting at position i. Since $j \geq i$ and by the choice of i, $x_j \notin \lim(\pi)$. However, $x_j \in D_{\pi(j)}$ because the jth substitution replaces the subtree at x_j. Furthermore, $x_k \not\sqsubset x_j$ for all $k \geq j$ and thus $x_j \in D_{\pi(k)}$ for all $k \geq j$. This verifies that π is not stable. □

From Lemmas 7 and 8 we can conclude that the trees on paths π with $\pi : t \to^{\omega}_{\mathcal{R}} T(\mathcal{A})$ without loop grow indefinitely. To develop an algorithm that can check whether such a path exists, we show that if such a path π exists, then it has the following "nice" property:

— For infinitely many trees on π there are accepting runs of $\mathcal{A}$. These runs do not differ too much, i.e., there are infinitely many trees on π such that there are accepting runs on these trees that agree on growing initial segments of the trees.

This property is formalized with the next definitions and the subsequent lemma.

A branch β of a prefix closed set $X \subseteq \mathbb{N}^*$ is a maximal subset of X that is linearly ordered by $\sqsubseteq$. The initial segment $\beta^{\uparrow x}$ of β up to a location $x \in \beta$ is the set

$$\beta^{\uparrow x} = \{y \in \beta \mid y \sqsubset x\}.$$

For $x \in \beta$ we denote by $\mathrm{succ}_\beta(x)$ the successor of x on β if it exists. With this definition we get the equality

$$\beta^{\uparrow \mathrm{succ}_\beta(x)} = \beta^{\uparrow x} \cup \{x\}.$$

The following lemma shows that the desired property of π, which is described above, can always be guaranteed.

Lemma 9. *If $t \to^{\omega}_{\mathcal{R}} T(\mathcal{A})$ without loop, then every infinite path π with $\pi : t \to^{\omega}_{\mathcal{R}} T(\mathcal{A})$ has the following properties*:

(i) $\lim(\pi)$ *has an infinite branch β.*

(ii) *For every infinite branch β of* $\lim(\pi)$ *there is a mapping $\rho_\beta : \beta \to Q$ with the following property. For each $x \in \beta$ there are infinitely many $i \in \mathbb{N}$ such that there is an accepting run of $\mathcal{A}$ on $\pi(i)$ that agrees with ρ_β on the initial segment $\beta^{\uparrow x}$ of β.*

Proof. Let $\pi : t \to^{\omega}_{\mathcal{R}} T(\mathcal{A})$ be an infinite path and let $[x_0/s_0], [x_1/s_1], \ldots$ be the derivation of π.

From Lemma 7 we can conclude that π is stable and hence $\lim(\pi)$ is infinite according to Lemma 8. If we view the locations in $\lim(\pi)$ as vertices and the successor relation on locations as edge relation, then $\lim(\pi)$ is a finitely branching (graph theoretic) tree. Thus, we can choose an infinite branch β of $\lim(\pi)$ by König's lemma, establishing (i).

To define ρ_β we again make use of König's lemma by defining a graph with vertices of the form (τ, x) with location $x \in \beta$ and mapping τ defined on $\beta^{\uparrow x}$. Formally, (τ, x)

is a vertex of this graph if τ is a mapping $\tau : \beta^{\uparrow x} \to Q$, and there are infinitely many trees on π that are accepted by $\mathcal{A}$ with an accepting run that agrees with τ on $\beta^{\uparrow x}$. Note that (τ, ε), where τ is the mapping with empty domain, is a vertex of the graph because $\beta^{\uparrow \varepsilon}$ is empty and therefore every accepting run agrees with τ on $\beta^{\uparrow \varepsilon}$. Furthermore, this graph has infinitely many vertices because for each x the initial segment $\beta^{\uparrow x}$ is finite. Thus, since $\beta^{\uparrow x} \subseteq \lim(\pi)$, there exists $i \in \mathbb{N}$ such that $\beta^{\uparrow x} \subseteq D_{\pi(j)}$ for all $j \geq i$. Hence, among the infinitely many accepting runs on trees on π, there must be infinitely many that agree on $\beta^{\uparrow x}$, defining a mapping τ such that (τ, x) is a vertex of the graph.

We continue by defining the edge relation of the graph. Between two vertices (τ_1, x_1) and (τ_2, x_2) there is an edge if $x_2 = \mathrm{succ}_\beta(x_1)$ and $\tau_1(y) = \tau_2(y)$ for all $y \in \beta^{\uparrow x_1}$.

This graph is an infinite tree (in the graph-theoretic sense) with root (τ, ε). The degree of this graph is bounded by $|Q|$. Therefore, by König's lemma, there is an infinite path through this graph. The mappings on this path agree on growing initial segments of β. In the limit this path defines a mapping ρ_β that satisfies condition (ii). □

Now we are ready to develop the decision procedure for the problem "$T'_i \to^\omega_{\mathcal{R}} T(\mathcal{A}(q))$ without loop." The idea is, for π and β as in the previous lemma, to simulate the substitutions on π along β with a finite amount of information. We define a finite graph $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ with edges labeled from the set $\{0, \ldots, k-1, /, !\}$, where k is the maximal rank of symbols used in the RGTRS $\mathcal{R}$. Passing an edge in $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ corresponds to different actions in the simulation of the substitutions along β. The symbols $0, \ldots, k-1$ mean that we go down the branch β, the symbol / means that we simulate a sequence of substitutions at the current location on β, and the symbol ! means that we simulate a sequence of substitutions at the current location on β in which a tree occurs from the set that should be visited infinitely often. An infinite path with infinitely many !-edges through $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ corresponds to an $\mathcal{R}$-path with infinitely many trees from the set that should be visited infinitely often.

We first define the graph $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ and then give a more detailed description of the idea and the correctness proofs.

Recall that $\mathcal{A}'_i$ denotes an NTA for T'_i, the right-hand side of the ith rewriting rule of $\mathcal{R}$. Define the automaton $\mathcal{B}' = \bigcup_{i=1}^m \mathcal{A}'_i$ and call its state set P', i.e., $P' = \bigcup_{i=1}^m Q'_i$.

Definition 1. The finite graph $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ has the vertex set $P' \times Q \times Q$. The edges of $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ are labeled from the set $\{0, \ldots, k-1, /, !\}$ and are defined as follows:

(1) $(p, q, r) \xrightarrow{i} (\hat{p}, \hat{q}, \hat{r})$ for $i \in \{0, \ldots, k-1\}$ if there is $l \in \{i+1, \ldots, k\}$, a symbol $a \in A_l$, a transition $(p_0, \ldots, p_{l-1}, a, p)$ in $\mathcal{B}'$, and transitions $(q_0, \ldots, q_{l-1}, a, q)$, $(r_0, \ldots, r_{l-1}, a, r)$ in $\mathcal{A}$ with the following properties:
— $\hat{p} = p_i, \hat{q} = q_i, \hat{r} = r_i$.
— For each $h \in \{0, \ldots, l-1\} \setminus \{i\}$, $T(\mathcal{B}'(p_h)) \to^*_{\mathcal{R}} T(\mathcal{A}(r_h)) \to^*_{\mathcal{R}} T(\mathcal{A}(q_h))$.

(2) $(p, q, r) \xrightarrow{/} (\hat{p}, q, r)$ if there is $i \in \{1, \ldots, m\}$ such that $\hat{p} \in F'_i$ and $T(\mathcal{B}'(p)) \to^*_{\mathcal{R}} T_i$.

(3) $(p, q, r) \xrightarrow{!} (\hat{p}, q, q)$ if there is $i \in \{1, \ldots, m\}$ such that $\hat{p} \in F'_i$ and $T(\mathcal{B}'(p)) \to^*_{\mathcal{R}} T(\mathcal{A}(r)) \to^*_{\mathcal{R}} T_i$.

Assume that there is a path π as in Lemma 9 that visits $T(\mathcal{A}(q))$ infinitely often and starts with a tree t from T_i'. Let $p \in F_i'$ be such that $t \in T(\mathcal{B}'(p))$.

The idea of the graph $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ is to simulate the substitutions that are made on π along the branch β. During this simulation we go down the branch β and jump to increasing positions on the path π. So, we are always at a location y in β and at a position n on π such that y is in $D_{\pi(n)}$.

The first component of the $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ vertices holds information about the trees on π, namely what state of $\mathcal{B}'$ may be reached when reading the subtree $\pi(n)^{\downarrow y}$.

The second component of the $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ vertices keeps track of the value $\rho_\beta(y)$. We know that there are infinitely many trees on π that are accepted with runs of $\mathcal{A}$ that agree with ρ_β on growing initial segments of β. Nevertheless, these runs may differ from ρ_β from a certain location onwards. The third component of the $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ vertices contains this information. As soon as we pass an accepted tree, the third component is reset to the value of the second component, and we wait for the next tree that is accepted with a run that agrees with ρ_β on the initial segment of β up to the current location y.

We start at position $n = 0$ on π at the location $y = \varepsilon$ at β with the vertex (p, q, q). An edge labeled with $i \in \{0, \ldots, k-1\}$ means that we go down one step along β to $\mathrm{succ}_\beta(y) = yi$, so the new y is the successor of the old y on β. The edges labeled with / mean that we jump to a new position n on π, namely to the position just after the last substitution at the current location y on β. Finally, the edges labeled with ! mean the same as the /-edges with the difference that between the current position n and the new position n on π there is a tree that is accepted by $\mathcal{A}(q)$ with a run that agrees with ρ_β on the initial segment of β up to the location that was the current location the last time an !-edge was taken. Condition (ii) of Lemma 9 ensures that we can infinitely often use such an !-edge.

Lemma 10. *Let $i \in \{1, \ldots, m\}$ and $q \in Q$. If $T_i' \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$ without loop, then there is $p \in F_i'$ and an infinite path with infinitely many !-edges through $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ starting from (p, q, q).*

Proof. Let π be an infinite path as in Lemma 9 with $t \in T_i'$ and $\mathcal{A} = \mathcal{A}(q)$. Let β be an infinite branch in $\lim(\pi)$ as in Lemma 9(i) and let $\rho_\beta : \beta \to Q$ be as in Lemma 9(ii). We know that the first tree $\pi(0)$ on π is in T_i'. Let $p \in F_i'$ such that there is an accepting run of $\mathcal{A}_i'(p)$ on $\pi(0)$.

To formalize the idea described above we inductively define a sequence of tuples $(p_j, q_j, r_j, y_j, z_j, z_j', n_j)$ such that the sequence (p_j, q_j, r_j) is an infinite path with infinitely many !-edges through $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ starting from (p, q, q). The other auxiliary components keep track of the current location y_j on β, the location z_j that was the current location the last time an !-edge was used, the location z_j' that was the current location the last time a /-edge or !-edge was used, and the current position n_j on π. For this inductive definition we need some notation.

For $x \in \beta$ let $\mathrm{stable}(x)$ denote the minimal position on π such that x is stable on $\pi[\mathrm{stable}(x), \infty)$. There are two possibilities for the last substitution before the position $\mathrm{stable}(x)$. Either a subtree rooted at a proper prefix of x was rewritten, or the subtree at x was rewritten itself. For those locations where the latter holds we know that $(\pi(\mathrm{stable}(x)))^{\downarrow x} \in T_l'$ for some $l \in \{1, \ldots, m\}$. Hence, we can fix a partial run

ρ'_x of $\mathcal{A}'_l$ on $\pi(\text{stable}(x))$ with $\rho'_x(x) \in F'_l$. We will need these runs to update the first component of our tuples, which are states from $\mathcal{B}'$. If ε is stable on π, then ρ'_ε is not defined. So, for the first updates to be correct we let ρ'_ε be an accepting run of $\mathcal{A}'_i(p)$ on $\pi(0)$, independent of whether or not ε is stable on π.

The initial tuple is $(p_0, q_0, r_0, y_0, z_0, z'_0, n_0) = (p, q, q, \varepsilon, \varepsilon, \varepsilon, 0)$. For all $j \in \mathbb{N}$ we define $p_j = \rho'_{z'_j}(y_j)$ and $q_j = \rho_\beta(y_j)$. Note that this is compatible with the definition for $j = 0$. Below we will show that the right-hand side in the definition of p_j is always defined (property (F)). Now, assume that for $j \in \mathbb{N}$ the tuple $(p_j, q_j, r_j, y_j, z_j, z'_j, n_j)$ is already defined. By Lemma 9(ii) we can choose a minimal $l_j \geq n_j$ such that $\pi(l_j)$ is accepted by $\mathcal{A}(q)$ with a run ρ_j that agrees with ρ_β on $\beta^{\uparrow \text{succ}_\beta(z_j)}$. Note that l_j and ρ_j depend on n_j and z_j only. To define the next tuple we have to distinguish three cases (corresponding to the three types (1), (2), (3) of edges in $G_{\text{Rec}}(\mathcal{R}, \mathcal{A})$):

(a) If y_j is stable on $\pi[n_j, \infty)$, then

$$y_{j+1} = \text{succ}_\beta(y_j), \quad z_{j+1} = z_j, \quad z'_{j+1} = z'_j, \quad n_{j+1} = n_j, \quad r_{j+1} = \rho_j(y_{j+1}).$$

If y_j is not stable on $\pi[n_j, \infty)$, then we distinguish two subcases:

(b) If $l_j \geq \text{stable}(y_j)$, then

$$y_{j+1} = y_j, \quad z_{j+1} = z_j, \quad z'_{j+1} = y_j, \quad n_{j+1} = \text{stable}(y_j), \quad r_{j+1} = r_j.$$

(c) If $l_j < \text{stable}(y_j)$, then

$$y_{j+1} = y_j, \quad z_{j+1} = y_j, \quad z'_{j+1} = y_j, \quad n_{j+1} = \text{stable}(y_j), \quad r_{j+1} = q_j.$$

The following properties can be shown by induction on j:

(A) $y_j, z_j, z'_j \in \beta$ and $z_j \sqsubseteq z'_j \sqsubseteq y_j$.
This is obvious from the definitions of y_j, z_j, and z'_j.

(B) $y_j \in D_{\pi(n_j)}$.
For $j = 0$ this is obvious. In case (a) y_j is stable on $\pi[n_j, \infty)$ and therefore, since β is an infinite branch in $\text{lim}(\pi)$, $\text{succ}_\beta(y_j) \in D_{\pi(n_{j+1})}$ because $n_{j+1} = n_j$. Note that for the same reasons $\text{succ}_\beta(y_j) \in D_{\pi(l_j)}$ and hence r_{j+1} is well defined. In cases (b) and (c) this is obvious since $n_{j+1} = \text{stable}(y_j)$ and $y_{j+1} = y_j$.

(C) If $u \sqsubset y_j$, then u is stable on $\pi[n_j, \infty)$.
In cases (b) and (c) $n_{j+1} = \text{stable}(y_j)$ and $y_{j+1} = y_j$. Thus, even y_{j+1} is stable on $\pi[n_{j+1}, \infty)$ and therefore all of its prefixes are. In case (a) y_j is stable on $\pi[n_j, \infty)$ and hence all its prefixes are. The claim follows from the definition of y_{j+1} and n_{j+1}.

(D) $n_j = \text{stable}(z'_j)$ for $j \geq 1$ and if ε is stable on π, then also for $j = 0$.
Induction base. If ε is stable on π, then the claim obviously holds for $j = 0$. In the other case, i.e., if ε is not stable on π, we have to start for $j = 1$. However, if ε is not stable on π, then (b) or (c) is used in the definition for $j = 1$. Hence, $n_1 = \text{stable}(\varepsilon)$ and $z'_1 = y_0 = \varepsilon$.
Induction step. The claim directly follows from the inductive hypothesis and the definitions of n_{j+1} and z'_{j+1}.

(E) $\rho'_{z'_j}$ is defined.

Induction step. In case (a) $z'_{j+1} = z'_j$ and hence $\rho'_{z'_{j+1}} = \rho'_{z'_j}$ is defined by induction. In cases (b) and (c) $z'_{j+1} = y_j$ and y_j is not stable on $\pi[n_j, \infty)$. From (C) we know that all proper prefixes of y_j are stable on $\pi[n_j, \infty)$ and thus the last substitution before y_j becomes stable must rewrite the subtree at y_j.

(F) $\rho'_{z'_j}(y_j)$ is defined.

From (A) we know that z'_j is a prefix of y_j. To prove the claim it remains to show that y_j is in the domain of $\pi(\text{stable}(z'_j))$. For $j = 0$ this is obvious and for $j \geq 1$ we have $\text{stable}(z'_j) = n_j$ by (D) and $y_j \in D_{\pi(n_j)}$ by (B).

(G) $r_j = \rho_j(y_j)$.

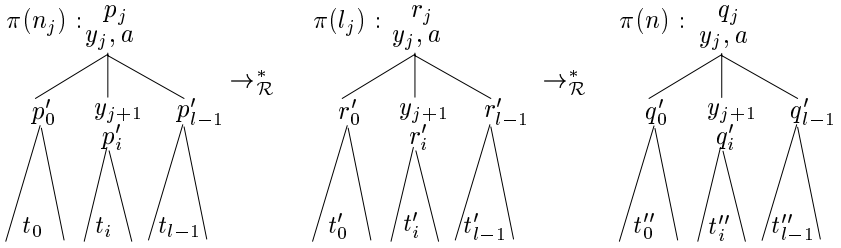
For $j = 0$ this is immediate from the fact that ρ_0 is an accepting run of $\mathcal{A}(q)$ and hence $r_0 = q = \rho_0(\varepsilon) = \rho_0(y_0)$. In cases (a) and (b) this directly follows from the fact that $l_{j+1} = l_j$ and hence $\rho_{j+1} = \rho_j$. In case (c) $z_{j+1} = y_{j+1}$, so ρ_{j+1} agrees with ρ_β on $\beta^{\uparrow \text{succ}_\beta(y_{j+1})}$. Hence, $\rho_{j+1}(y_{j+1}) = \rho_\beta(y_{j+1}) = q_{j+1} = q_j = r_{j+1}$. The equality $q_j = q_{j+1}$ follows from $y_j = y_{j+1}$.

The sequence we have defined corresponds to an infinite path through $G_{\text{Rec}}(\mathcal{R}, \mathcal{A})$ in the following sense. If tuple $j+1$ was defined from tuple j using case (a), (b), or (c), then there is an edge in $G_{\text{Rec}}(\mathcal{R}, \mathcal{A})$ between (p_j, q_j, r_j) and $(p_{j+1}, q_{j+1}, r_{j+1})$ of type (1), (2), or (3), respectively.

If (a) is used, then let i be such that $y_{j+1} = \text{succ}_\beta(y_j) = y_j i$. We show that the conditions for an i-edge of type (1) are satisfied. Let $a \in A_l$ be the symbol (of rank l) at location y_j in the tree $\pi(n_j)$. Note that y_j is stable on $\pi[n_j, \infty)$ and hence location y_j is labeled by a on all trees $\pi(n)$ for $n \geq n_j$. By definition of p_j and by (D), $\rho'_{z'_j}$ is a partial run of $\mathcal{B}'$ on $\pi(n_j)$ with $\rho'_{z'_j}(y_j) = p_j$. Let $(p'_0, \ldots, p'_{l-1}, a, p_j)$ be the transition of $\mathcal{B}'$ used in $\rho'_{z'_j}$ at y_j. Since $z'_{j+1} = z'_j$ and $y_{j+1} = y_j i$, we get $p'_i = p_{j+1}$.

By (G) we know that $\rho_j(y_j) = r_j$. Let $(r'_0, \ldots, r'_{l-1}, a, r_j)$ be the transition used in ρ_j at y_j. Since $r_{j+1} = \rho_j(y_{j+1})$, we get that $r'_i = r_{j+1}$.

Furthermore, by Lemma 9(ii), there exists $n > l_j$ such that $\pi(n)$ is accepted by $\mathcal{A}(q)$ with a run that agrees with ρ_β on $\beta^{\uparrow \text{succ}_\beta(y_{j+1})}$. Let $(q'_0, \ldots, q'_{l-1}, a, q_j)$ be the transition that is used in such a run at location y_j. In particular, this means that in this run the state at y_{j+1} must equal q_{j+1}, i.e., $q'_i = q_{j+1}$. For $h \in \{0, \ldots, l-1\}$ we denote the subtrees at $y_j h$ in $\pi(n_j)$, $\pi(l_j)$, and $\pi(n)$ by t_h, t'_h, and t''_h, respectively. The situation looks as follows, where not the whole trees but only the subtrees of $\pi(n_j)$, $\pi(l_j)$, and $\pi(n)$ at y_j are shown:



Since y_j is stable on $\pi[n_j, \infty)$, we can conclude from Lemma 3(ii) that $t_h \to_{\mathcal{R}}^* t'_h \to_{\mathcal{R}}^* t''_h$ for each $h \in \{0, \ldots, l-1\}$. Hence, the transitions $(p'_0, \ldots, p'_{l-1}, a, p_j)$, $(r'_0, \ldots, r'_{l-1}, a, r_j)$, and $(q'_0, \ldots, q'_{l-1}, a, q_j)$ satisfy the conditions from (1) and there is an i-edge from (p_j, q_j, r_j) to $(p_{j+1}, q_{j+1}, r_{j+1})$ in $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$.

If (b) is used, then $z'_{j+1} = y_j = y_{j+1}$ and $n_{j+1} = \mathrm{stable}(y_j)$. Hence $\rho'_{z'_{j+1}}$ is a partial run of $\mathcal{A}'_i$ on $\pi(n_{j+1})$ with $\rho'_{z'_{j+1}}(z'_{j+1}) \in F'_i$ for some $i \in \{1, \ldots, m\}$. By the definition of p_{j+1} this implies that $p_{j+1} \in F'_i$. Furthermore, we have $\pi(n_{j+1} - 1)^{\downarrow y_j} \in T_i$ because the rewriting rule $T_i \hookrightarrow T'_i$ is applied to this subtree at this position on π. By (D), $\rho'_{z'_j}$ is a partial run on $\pi(n_j)$ and $p_j = \rho'_{z'_j}(y_j)$. Since, by (C), all proper prefixes of y_j are stable on $\pi[n_j, \infty)$, we get $\pi(n_j)^{\downarrow y_j} \to_{\mathcal{R}}^* \pi(n_{j+1} - 1)^{\downarrow y_j}$ from $\pi(n_j) \to_{\mathcal{R}}^* \pi(n_{j+1})$ and Lemma 3(ii), and hence $T(\mathcal{B}'(p_j)) \to_{\mathcal{R}}^* T_i$. Therefore, the conditions of (2) for a /-edge between (p_j, q_j, r_j) and $(p_{j+1}, q_{j+1}, r_{j+1})$ in $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ are satisfied. Similarly, the conditions for an !-edge between (p_j, q_j, r_j) and $(p_{j+1}, q_{j+1}, r_{j+1})$ are satisfied if (c) is used because the tree $\pi(l_j)$ is lying between $\pi(n_j)$ and $\pi(n_{j+1})$, and $\pi(l_j)^{\downarrow y_j} \in T(\mathcal{A}(r_j))$ by (G).

It remains to show that (c) is used infinitely often to obtain a path with infinitely many !-edges. As long as (c) is not used, (b) must be used eventually because only finitely many locations from β can be stable on a fixed suffix of π. If (a) or (b) is used, then $l_{j+1} = l_j$. However, every time (b) is used the n_j value increases. By definition of l_j we have $l_j \geq n_j$. Therefore, rule (c) has to be applied eventually. □

Having shown the "completeness" of $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ we now come to the correctness as stated in the following lemma. One should note that we can only conclude $T'_i \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$ from the existence of a path through $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ as required in the lemma. It is not possible to conclude $T'_i \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$ *without loop* since the required path through $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ only witnesses some $\pi : T'_i \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$ that does not contain a loop. For $T'_i \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$ without loop to hold *every* $\pi : T'_i \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$ needs to be without loop.

Lemma 11. *Let $i \in \{1, \ldots, m\}$ and $q \in Q$. If there is an infinite path containing infinitely many !-edges through $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ starting from (p, q, q), for some $p \in F'_i$, then $T'_i \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q))$.*

Proof. We cut the infinite path through $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ into segments ending with an !-edge and show what kind of $\mathcal{R}$-path we can construct from such a finite path segment in $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$. The desired result is then obtained by concatenating these segments, as we will see later.

So let $p, p' \in P', q, r, q' \in Q$, and $v_1, \ldots, v_n$ be vertices of $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ such that

$$(p, q, r) \xrightarrow{\lambda_1} v_1 \xrightarrow{\lambda_2} v_2 \xrightarrow{\lambda_3} \cdots \xrightarrow{\lambda_n} v_n \xrightarrow{!} (p', q', q')$$

in $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ with $\lambda_1, \ldots, \lambda_n \in \{0, \ldots, k-1, /\}$. Let x be the location that is obtained from the sequence $\lambda_1 \cdots \lambda_n$ by omitting all /.

We prove the following claim by induction on n.

Claim. *There is $t \in T(\mathcal{B}'(p))$ such that for all $t' \in T(\mathcal{B}'(p'))$ there is $t'' \in T_A$ with*

$$t \to_{\mathcal{R}}^* T(\mathcal{A}(r)) \to_{\mathcal{R}}^+ t'' \qquad \text{with} \quad (t'')^{\downarrow x} = t' \quad \text{and} \quad t''[x/q'] \to_{\mathcal{A}}^* q. \qquad (\star)$$

If $n = 0$, then $(p, q, r) \xrightarrow{!} (p', q', q')$ with $q' = q$ and $x = \varepsilon$. By definition of the !-edges there are $i \in \{1, \ldots, m\}$ and $t \in T(\mathcal{B}'(p))$ such that $p' \in F_i'$ and $t \to_{\mathcal{R}}^* T(\mathcal{A}(r)) \to_{\mathcal{R}}^* T_i$. Let $t' \in T(\mathcal{B}'(p'))$ and set $t'' = t'$. Then $t'' \in T_i'$ because $t'' = t' \in T(\mathcal{B}'(p'))$ and $p' \in F_i'$. By definition of t'' and because $x = \varepsilon$ and $q' = q$ we obviously have $(t'')^{\downarrow x} = t'$ and $t''[x/q'] \to_{\mathcal{A}}^* q$. Furthermore, $t \to_{\mathcal{R}}^* T(\mathcal{A}(r)) \to_{\mathcal{R}}^* T_i \to_{\mathcal{R}} t''$ because $t'' \in T_i'$.

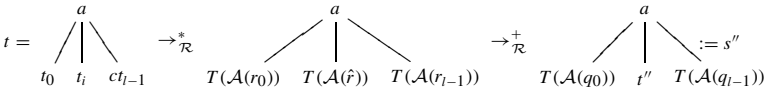
If $n \geq 1$, then

$$(p, q, r) \xrightarrow{\lambda_1} (\hat{p}, \hat{q}, \hat{r}) \xrightarrow{\lambda_2} v_2 \xrightarrow{\lambda_3} \cdots \xrightarrow{\lambda_n} v_n \xrightarrow{!} (p', q', q').$$

Let $\hat{x}$ be the location obtained from $\lambda_2 \cdots \lambda_n$ by omitting /. By the induction hypothesis there is $\hat{t} \in T(\mathcal{B}'(\hat{p}))$ such that for all $t' \in T(\mathcal{B}'(p'))$ there is a t'' such that $(\star)$ is valid for $\hat{t}, \hat{x}, \hat{q}, \hat{r}, t'$, and t''.

If $\lambda_1 = /$, then $x = \hat{x}, q = \hat{q}, r = \hat{r}$. Furthermore, there is $i \in \{1, \ldots, m\}$ such that $\hat{p} \in F_i'$ and there is $t \in T(\mathcal{B}'(p))$ with $t \to_{\mathcal{R}}^* T_i$. However, $\hat{p} \in F_i'$ implies $\hat{t} \in T_i'$ and hence $t \to_{\mathcal{R}}^* T_i \to_{\mathcal{R}} \hat{t}$. Since $(\star)$ holds for $\hat{t}$ it also holds for t because $x = \hat{x}$, $q = \hat{q}$, and $r = \hat{r}$.

Now consider the case $\lambda_1 = i \in \{0, \ldots, k-1\}$. Let $l \in \{i+1, \ldots, k\}$, $a \in A_l$, and p_h, q_h, r_h for $h \in \{0, \ldots, l-1\}$ be as required in item (1) from the definition of $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$. Then there are $t_h \in T(\mathcal{B}'(p_h))$ for $h \in \{0, \ldots, l-1\} \backslash \{i\}$ such that $t_h \to_{\mathcal{R}}^* T(\mathcal{A}(r_h)) \to_{\mathcal{R}}^* T(\mathcal{A}(q_h))$. Furthermore, let $t_i = \hat{t}$. If we define $t = a(t_1, \ldots, t_l)$, then $t \in T(\mathcal{B}'(p))$. Let $t' \in T(\mathcal{B}'(p'))$. The part $t \to_{\mathcal{R}}^* T(\mathcal{A}(r)) \to_{\mathcal{R}}^+ s''$ of the claim follows as indicated in the diagram below. In this diagram sets of trees as, e.g., $T(\mathcal{A}(r_0))$ mean that there is a tree from that set that can be substituted at this position. The rightmost tree in the diagram is defined to be s''.



Note that the tree in the middle is in $T(\mathcal{A}(r))$ because $\hat{r} = r_i$ and the transition $(r_0, \ldots, r_{l-1}, a, r)$ is in $\mathcal{A}$.

By induction, we know that $t''[\hat{x}/q'] \to_{\mathcal{A}}^* \hat{q}$. From $\hat{q} = q_i$ and $x = i\hat{x}$ it follows that $s''[x/q'] \to_{\mathcal{A}}^* q$ because the transition $(q_0, \ldots, q_{l-1}, a, q)$ is in $\mathcal{A}$. Furthermore, we have $(s'')^{\downarrow x} = (t'')^{\downarrow \hat{x}} = t'$. This ends the proof of the claim. Now we show how to iterate this result.

On a path with infinitely many !-edges starting in (p_1, q_1, q_1) with $p_1 \in F_i'$ let $(p_2, q_2, q_2), (p_3, q_3, q_3), \ldots$ be the vertices reached after the !-edges. For all $j \geq 1$ there is $t_j \in T(\mathcal{B}'(p_j))$ such that $(\star)$ holds for location x_j (so x_j is obtained from the edge

labels of the corresponding path segment by omitting /) and for all $t' \in T(\mathcal{B}'(p_{j+1}))$. In particular, we can use t_{j+1} in place of t'. So, for each j, let t_j'' be a tree with

$$t_j \to_{\mathcal{R}}^* T(\mathcal{A}(q_j)) \to_{\mathcal{R}}^+ t_j'' \qquad \text{with} \quad (t_j'')^{\downarrow x_j} = t_{j+1} \quad \text{and} \quad t_j''[x_j/q_{j+1}] \to_{\mathcal{A}}^* q_j.$$

From the condition $(t_j'')^{\downarrow x_j} = t_{j+1}$ we get an infinite path of the form

$$t_1 \to_{\mathcal{R}}^* t_1'' \to_{\mathcal{R}}^* t_1''[x_1/t_2''] \to_{\mathcal{R}}^* t_1''[x_1/t_2''[x_2/t_3'']] \to_{\mathcal{R}}^* \cdots,$$

and with the condition $t_j''[x_j/q_{j+1}] \to_{\mathcal{A}}^* q_j$ we can conclude that there are infinitely many trees from $T(\mathcal{A}(q_1))$ on this path. Furthermore, t_1 is in T_i' because $p_1 \in F_i'$. Thus, $T_i' \to_{\mathcal{R}}^{\omega} T(\mathcal{A}(q_1))$. □

Since we want to use $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ in a decision procedure we need the following lemma.

Lemma 12. *The graph $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ can be constructed effectively in time $\mathcal{O}(|\mathcal{R}|^4 \cdot |\mathcal{A}|^4)$.*

Proof. To construct $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ we have to check the conditions for the edges in (1)–(3) of Definition 1. These are mainly instances of the reachability problem. It is clear that instances of the reachability problem with three sets involved are more difficult than the ones with only two sets involved. So we have to estimate the complexity of

(i) $T(\mathcal{B}'(p)) \to_{\mathcal{R}}^* T(\mathcal{A}(r)) \to_{\mathcal{R}}^* T(\mathcal{A}(q))$ in (1) and
(ii) $T(\mathcal{B}'(p)) \to_{\mathcal{R}}^* T(\mathcal{A}(r)) \to_{\mathcal{R}}^* T_i$ in (3).

According to Lemma 2 the set of all (p, q, r) with property (i) can be computed in time $\mathcal{O}(|\mathcal{R}|^3 \cdot |\mathcal{A}| \cdot (|\mathcal{A}| + |\mathcal{R}|))$ because $|\mathcal{B}'| \leq |\mathcal{R}|$ by definition of $|\mathcal{R}|$.

For (ii) let $\mathcal{B}$ denote the automaton $\bigcup_{i=1}^m \mathcal{A}_i$. We can compute the set of all tuples (q_1, q_2, q_3) of states from $\mathcal{B}'$, $\mathcal{A}$, and $\mathcal{B}$ with $T(\mathcal{B}'(q_1)) \to_{\mathcal{R}}^* T(\mathcal{A}(q_2)) \to_{\mathcal{R}}^* T(\mathcal{B}(q_3))$ in time $\mathcal{O}(|\mathcal{R}|^4 \cdot |\mathcal{A}|)$, again by Lemma 2. For q_3 belonging to one of the sets F_i we obtain the desired result.

The time needed to compute all instances (p, i, j) satisfying $T(\mathcal{B}'(p)) \to_{\mathcal{R}}^* T_i$ can be bounded as for (ii) because these are special instances of the problems from (ii).

Then we can check for each pair (p, q, r), $(\hat{p}, \hat{q}, \hat{r})$ of vertices of $G_{\mathrm{Rec}}(\mathcal{R}, \mathcal{A})$ whether the conditions of (1)–(3) are satisfied. After the previous computations these tests can be made in constant time and have to be carried out for $\mathcal{O}(|\mathcal{R}|^2 \cdot |\mathcal{A}|^4)$ pairs. Therefore, the overall time complexity can clearly be bounded by $\mathcal{O}(|\mathcal{R}|^4 \cdot |\mathcal{A}|^4)$. □

Now we are ready to summarize the results in an algorithm to solve the recurrence problem. The algorithm is shown in Figure 3.

Theorem 4. *The algorithm* RECUR($\mathcal{R}$,$\mathcal{A}$) *from Figure* 3 *computes in time $\mathcal{O}(|\mathcal{R}|^4 \cdot |\mathcal{A}|^4)$ an ε-NTA $\mathcal{A}_{\mathcal{R},\omega}$ of size $\mathcal{O}(|\mathcal{R}|^2 \cdot |\mathcal{A}|)$ with $T(\mathcal{A}_{\mathcal{R},\omega}) = \{t \in T_A \mid t \to_{\mathcal{R}}^{\omega} T(\mathcal{A})\}$.*

Algorithm RECUR

INPUT: RGTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ with $R = \{T_1 \overset{\sigma_1}{\hookrightarrow} T_1', \ldots, T_m \overset{\sigma_m}{\hookrightarrow} T_m'\}$,
NTAs $\mathcal{A}_i = (Q_i, A, \Delta_i, F_i)$ with $T(\mathcal{A}_i) = T_i$ for $i = 1, \ldots, m$
NTAs $\mathcal{A}_i'$ with $T(\mathcal{A}_i') = T_i'$ for $i = 1, \ldots, m$
NTA $\mathcal{A} = (Q, A, \Delta, F)$

1. Construct $G_{\text{Rec}}(\mathcal{R}, \mathcal{A})$ (Definition 1).
2. Mark each pair (i, q) with $T_i' \to_{\mathcal{R}}^* T(\mathcal{A}(q)) \to_{\mathcal{R}}^* T_i$.
3. Mark each pair (i, q) with $q \in Q, i \in \{1, \ldots, m\}$ such that there is $p \in F_i'$ and an infinite path with infinitely many !-edges through $G_{\text{Rec}}(\mathcal{R}, \mathcal{A})$ starting in (p, q, q).
4. Let $\mathcal{B}' = \mathcal{A} \cup \bigcup_{i=1}^m \mathcal{A}_i$ with F as set of final states.
5. Add ε-transitions (p, q) to $\mathcal{B}'$ if $p \in F_i$ and (i, q) is marked. Obtain the automaton $\mathcal{B}$.
6. Let $\mathcal{A}_{\mathcal{R},\omega} = (\mathcal{B}_{!\varepsilon})_{\text{pre}_{\mathcal{R}}^*}$.

OUTPUT: ε-NTA $\mathcal{A}_{\mathcal{R},\omega}$

Fig. 3. Algorithm to solve the recurrence problem.

Proof. For the time complexity the operations on $G_{\text{Rec}}(\mathcal{R}, \mathcal{A})$ are the dominating factor because $(\mathcal{B}_{!\varepsilon})_{\text{pre}_{\mathcal{R}}^*}$ can be constructed in time $\mathcal{O}(|\mathcal{R}|^2 \cdot (|\mathcal{B}| + |\mathcal{R}|))$ according to Proposition 3 and Theorem 3. By Lemma 12, $G_{\text{Rec}}(\mathcal{R}, \mathcal{A})$ itself can be constructed in time $\mathcal{O}(|\mathcal{R}|^4 \cdot |\mathcal{A}|^4)$ and the set of all (p, q, q) such that there is an infinite path with infinitely many !-edges starting in (p, q, q) can be determined by an algorithm computing the strongly connected components of $G_{\text{Rec}}(\mathcal{R}, \mathcal{A})$. This can be done in linear time in the size of $G_{\text{Rec}}(\mathcal{R}, \mathcal{A})$, which can clearly be bounded by $\mathcal{O}(|\mathcal{R}|^4 \cdot |\mathcal{A}|^4)$.

The automaton $\mathcal{B}$ is of size $|\mathcal{R}| \cdot |\mathcal{A}|$ and hence, by Proposition 3, the size of $\mathcal{B}_{!\varepsilon}$ is of the same order. Therefore, by Theorem 2, $\mathcal{A}_{\mathcal{R},\omega}$ is of size $\mathcal{O}(|\mathcal{R}| \cdot (|\mathcal{R}| \cdot |\mathcal{A}| + |\mathcal{R}|)) = \mathcal{O}(|\mathcal{R}|^2 \cdot |\mathcal{A}|)$.

For the correctness we have to show by Lemma 5 that $\mathcal{A}_{\mathcal{R},\omega}$ recognizes the set $\text{pre}_{\mathcal{R}}^*(\text{Rec}_1(\mathcal{R}, \mathcal{A}) \cup \text{Rec}_2(\mathcal{R}, \mathcal{A}))$. Since $\mathcal{A}_{\mathcal{R},\omega} = (\mathcal{B}_{!\varepsilon})_{\text{pre}_{\mathcal{R}}^*}$ it is sufficient to show that

$$\text{Rec}_1(\mathcal{R}, \mathcal{A}) \cup \text{Rec}_2(\mathcal{R}, \mathcal{A}) \subseteq T(\mathcal{B}_{!\varepsilon}) \subseteq \text{pre}_{\mathcal{R}}^*(\text{Rec}_1(\mathcal{R}, \mathcal{A}) \cup \text{Rec}_2(\mathcal{R}, \mathcal{A})).$$

If $t \in T(\mathcal{B}_{!\varepsilon})$, then there are states p, q of $\mathcal{B}$ and $x \in D_t$ such that $t \to_{\mathcal{B}'}^* t[x/p] \to_{\mathcal{B}} t[x/q] \to_{\mathcal{B}'}^* F$, according to Proposition 3. This implies that there is $i \in \{1, \ldots, m\}$ such that $p \in F_i$ and (i, q) is marked (line 5 of the algorithm). Since $\mathcal{B}'$ is a disjoint union of $\mathcal{A}$ and $\mathcal{A}_1, \ldots, \mathcal{A}_m$ we can conclude $t^{\downarrow x} \in T_i$ from $t \to_{\mathcal{B}'}^* t[x/p]$ and $p \in F_i$. The pair (i, q) is marked and thus q is a state of $\mathcal{A}$. Hence we can deduce from $t[x/q] \to_{\mathcal{B}'}^* F$ that $t[x/q] \to_{\mathcal{A}}^* F$. Now it is easy to see that t is in $\text{Rec}_1(\mathcal{R}, \mathcal{A})$ if (i, q) was marked in line 2 of the algorithm. If (i, q) was marked in line 3, then $T_i' \to_{\mathcal{R}}^\omega T(\mathcal{A}(q))$ by Lemma 11. If $T_i' \to_{\mathcal{R}}^\omega T(\mathcal{A}(q))$ without loop, then t is in $\text{Rec}_2(\mathcal{R}, \mathcal{A})$. If $T_i' \to_{\mathcal{R}}^\omega T(\mathcal{A}(q))$ with loop, we can conclude that $t \to_{\mathcal{R}}^\omega T(\mathcal{A})$ with loop and hence $t \in \text{pre}_{\mathcal{R}}^*(\text{Rec}_1(\mathcal{R}, \mathcal{A}))$ by Lemma 4. This proves the inclusion $T(\mathcal{B}_{!\varepsilon}) \subseteq \text{pre}_{\mathcal{R}}^*(\text{Rec}_1(\mathcal{R}, \mathcal{A}) \cup \text{Rec}_2(\mathcal{R}, \mathcal{A}))$.

If $t \in \text{Rec}_1(\mathcal{R}, \mathcal{A}) \cup \text{Rec}_2(\mathcal{R}, \mathcal{A})$ one can easily show that t is accepted by $\mathcal{B}_{!\varepsilon}$ using an ε-transition (p, q) for a pair (i, q) marked in line 2 if $t \in \text{Rec}_1(\mathcal{R}, \mathcal{A})$. For $t \in \text{Rec}_2(\mathcal{R}, \mathcal{A})$, using Lemma 10, one obtains that t is accepted by $\mathcal{B}_{!\varepsilon}$ using an ε-transition (p, q) for a pair (i, q) marked in line 3. □

5. Undecidable Properties

In this section we prove that model-checking for GTR graphs with the temporal operators AF, EU, and AGF is undecidable (see Table 1). All the proofs use reductions from undecidable properties of deterministic Turing machines. We construct, given a Turing machine M, a GTRS $\mathcal{R}(M)$ that can simulate computations of M. Of course, since reachability is decidable for GTR graphs, it is not possible to exactly simulate a Turing machine with a GTRS. Therefore, in $G_{\mathcal{R}(M)}$ there will be paths that correspond to correct computations of M and there will also be paths that correspond to computations of M with some errors. However, the way $\mathcal{R}(M)$ is constructed allows us to detect these errors. Every time an error occurs in the computation, the path in $G_{\mathcal{R}(M)}$ contains a tree from a regular set T^M_{err}. If T^M_{stop} contains all trees coding a halting configuration of M, then $G_{\mathcal{R}(M)}$ is a model of the formula $AF(T^M_{\text{err}} \vee T^M_{\text{stop}})$ iff every path in $G_{\mathcal{R}(M)}$ that starts in the initial tree eventually reaches a tree modeling an error or a halting configuration. If the initial tree codes the initial configuration of M on the empty tape, then this means that M stops on the empty tape and therefore we have encoded the halting problem.

The construction of $\mathcal{R}(M)$ is given in the next subsection and in the following subsections it is shown how to use the idea sketched above to show the undecidability results.

The general idea underlying the simulation of Turing machines is similar to the idea used in [13]. In [13] it is shown via a reduction from the halting problem for counter machines that the problem of universal reachability for basic parallel processes is undecidable.

5.1. *Simulation of Turing Machines*

We only give a brief description of the Turing machine model we use. For an introduction to Turing machines see, e.g., [15]. A deterministic Turing machine (DTM) is a tuple $M = (Q, \Gamma, B, q_{\text{in}}, q_s, \delta)$, where Q is a finite set of states, Γ is the tape alphabet (disjoint from Q), $B \subset \Gamma$ is the input alphabet, q_{in} is the initial state, q_s is the halting state, and $\delta : (Q \backslash \{q_s\}) \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function. The transition function is completely specified and q_s is the unique halting state. Furthermore, we assume that the tape is infinite to the left and to the right. The tape alphabet Γ contains a blank symbol $\sqcup$ that is not an element of the input alphabet.

A configuration κ of M is a word $\kappa = a_1 \cdots a_k q b_l \cdots b_1$ with $q \in Q$, $k, l \geq 0$, and $a_i, b_j \in \Gamma$ for all $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, l\}$. We define the successor configuration of κ as follows, where we assume $a_0 = b_0 = \sqcup$ to also cover the cases for $l = 0$ or $k = 0$:

— $a_1 \cdots a_{k-1} p a_k c b_{l-1} \cdots b_1$ if $\delta(q, b_l) = (p, c, L)$, and
— $a_1 \cdots a_k c p b_{l-1} \cdots b_1$ if $\delta(q, b_l) = (p, c, R)$.

If κ' is the successor configuration of κ, then this is denoted by $\kappa \vdash_M \kappa'$. As usual $\vdash^*_M$ denotes the transitive and reflexive closure of $\vdash_M$.

To simulate Turing machines by GTR we define for each configuration κ a corresponding tree $t(\kappa)$ coding this configuration. For $\kappa = a_1 \cdots a_k q b_l \cdots b_1$ let $t(\kappa)$ be the tree depicted in Figure 4.

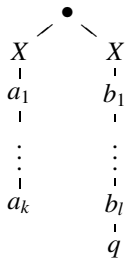


Fig. 4. Coding of a configuration $a_1 \cdots a_k q b_l \cdots b_1$ by a tree.

With this coding of configurations it is not possible to exactly simulate the transitions of M by a GTRS because in a transition a symbol from Γ has to be moved either from the left branch to the right branch of the tree or the other way round. To realize this with GTR the whole tree has to be rewritten. Since the length of Turing machine configurations is not bounded this would require an infinite set of rewriting rules. Therefore, the symbol that has to be moved from one branch to the other has to be guessed. To be able to detect wrong guesses, a protocol for the simulation of Turing machine transitions is introduced.

The upper part of Figure 5 shows the correct simulation of the transition $\delta(q, a) = (p, b, L)$ when M is in the configuration cqa. The initial tree (on the left-hand side of the figure) codes the configuration cqa and the final tree (on the right-hand side of the figure) codes the configuration pcb.

The single steps in the simulation are the following. The symbols a and q are replaced by b and p. Since M moves the head to the left, the symbol c, which is at the

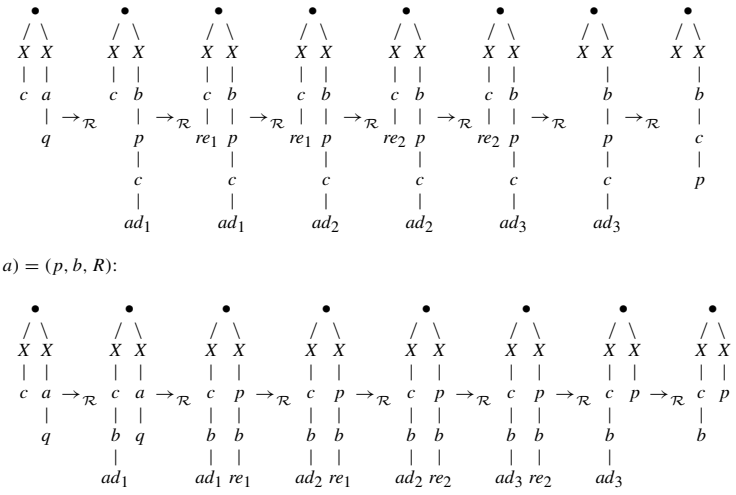


Fig. 5. Example for the simulation of Turing machine transitions by a GTRS.

end of the left branch, has to be guessed. The symbol ad_1 indicates that c, the symbol directly above ad_1, was added to the right branch of the tree. Now, the left branch has to confirm with re_1, where re stands for "remove" because the c has to be removed from the left branch. Then the two branches alternately increase their ad_i and re_i symbols until they reach ad_3 and re_2. Finally, the c can be removed from the left branch and then it is inserted at the appropriate place in the right branch.

The idea behind this is the following: If the guess is wrong, e.g., if d was guessed instead of c, then after the second step a tree containing the subtrees $c(re_1)$ and $d(ad_1)$ with $c \neq d$ is reached. This enables us to detect on a path through the graph generated by the rewriting system if an error occurred in the simulation of M. The process of going through all the symbols ad_2, re_2, and ad_3 has technical reasons. It ensures that other errors apart from the wrong guessing, e.g., repeated deletion of symbols without simulation of a transition, can be detected.

Formally, we define the GTRS $\mathcal{R}(M) = (A^M, \Sigma, R^M, t_{\text{in}}^M)$ for a given DTM $M = (Q, B, \Gamma, q_{\text{in}}, q_s, \delta)$ as follows:

— $A^M = A_0^M \cup A_1^M \cup A_2^M$ with $A_2^M = \{\bullet\}$, $A_1^M = Q \cup \Gamma \cup \{X\}$, and $A_0^M = A_1^M \cup \text{Aux}$, where $\text{Aux} = \{ad_1, ad_2, ad_3, re_1, re_2, err\}$. The use of err will be clarified in the proof of Lemma 13.

— As we do not need the transition labels Σ, we let $\Sigma = \{\sigma\}$ and omit all transition labels in the following.

— The initial tree is

$$t_{\text{in}}^M = t(q_{\text{in}}) = \begin{array}{ccc} & \bullet & \\ / & & \backslash \\ X & & X \\ & & | \\ & & q_{\text{in}} \end{array}.$$

— The set R^M contains the following rewriting rules:

(1) For $\delta(q, a) = (p, b, L)$, $c \in \Gamma$:

$$\begin{array}{c} a \\ | \\ q \end{array} \hookrightarrow \begin{array}{c} b \\ | \\ p \\ | \\ c \\ | \\ ad_1 \end{array} \quad \text{and if} \quad a = \sqcup, \text{ then also} \quad \begin{array}{c} X \\ | \\ q \end{array} \hookrightarrow \begin{array}{c} X \\ | \\ b \\ | \\ p \\ | \\ c \\ | \\ ad_1 \end{array}.$$

(2) For $\delta(q, a) = (p, b, R)$:

$$\begin{array}{c} a \\ | \\ q \end{array} \hookrightarrow \begin{array}{c} p \\ | \\ b \\ | \\ re_1 \end{array} \quad \text{and if} \quad a = \sqcup, \quad \text{then also} \quad \begin{array}{c} X \\ | \\ q \end{array} \hookrightarrow \begin{array}{c} X \\ | \\ p \\ | \\ b \\ | \\ re_1 \end{array}.$$

(3) For all $a \in \Gamma \cup \{X\}$, $b \in \Gamma$:

$$a \hookrightarrow \begin{array}{c} a \\ | \\ b \\ | \\ ad_1 \end{array}, \qquad b \hookrightarrow \begin{array}{c} b \\ | \\ re_1 \end{array}, \qquad X \hookrightarrow \begin{array}{c} X \\ | \\ \sqcup \\ | \\ re_1 \end{array}.$$

(4) $ad_1 \hookrightarrow ad_2$, $ad_2 \hookrightarrow ad_3$, $re_1 \hookrightarrow re_2$.

(5) For all $a \in \Gamma \cup \{X\}$, $b \in \Gamma$, $q \in Q$:

$$\begin{array}{c} q \\ | \\ b \\ | \\ ad_3 \end{array} \hookrightarrow \begin{array}{c} b \\ | \\ q \end{array}, \qquad \begin{array}{c} a \\ | \\ b \\ | \\ ad_3 \end{array} \hookrightarrow \begin{array}{c} a \\ | \\ b \end{array}.$$

(6) For all $a \in \Gamma \cup \{X\}$, $b \in \Gamma$, $q \in Q$:

$$\begin{array}{c} a \\ | \\ b \\ | \\ re_2 \end{array} \hookrightarrow a, \qquad \begin{array}{c} q \\ | \\ b \\ | \\ re_2 \end{array} \hookrightarrow q.$$

(7) For all $a \in \Gamma$, $p, q \in Q$:

$$q \hookrightarrow \begin{array}{c} p \\ | \\ err \end{array}, q \hookrightarrow \begin{array}{c} a \\ | \\ p \\ | \\ err \end{array}, \quad \textit{and} \quad \begin{array}{c} q \\ | \\ err \end{array} \hookrightarrow q.$$

The rules from (7) are used to ensure the property from Lemma 13(i), which is needed in Section 5.4.

The goal was to construct $\mathcal{R}(M)$ in such a way that we can identify all the paths in $G_{\mathcal{R}(M)}$ that do not correspond to a correct computation of M by a regular set T^M_{err}. This set is defined as follows. A tree t is in T^M_{err} iff

1. t contains the *err* symbol,
2. t contains more than one ad_i or more than one re_i symbol,
3. t contains a re_i symbol and no ad_i or ad_{i+1} symbol,
4. t contains an ad_2 symbol and no re_i symbol, or
5. t contains subtrees of the form $\begin{array}{c} a \\ | \\ re_i \end{array}$ and $\begin{array}{c} b \\ | \\ ad_j \end{array}$ with $a \neq b$.

This definition only asks for the presence or absence of certain combinations of symbols. These properties can be tested by a tree automaton and therefore T^M_{err} is regular.

The following lemma gives a precise statement in what sense the DTM M can be simulated by $\mathcal{R}(M)$.

Lemma 13.

(i) *For each configuration κ of M: $t^M_{\text{in}} \to^*_{\mathcal{R}(M)} t(\kappa)$.*

(ii) *Let κ and κ' be configurations of M.*

(a) *If $\kappa \vdash^*_M \kappa'$, then there is a path from $t(\kappa)$ to $t(\kappa')$ in $G_{\mathcal{R}(M)}$ not visiting T^M_{err}.*

(b) *If there is a path from $t(\kappa)$ to $t(\kappa')$ in $G_{\mathcal{R}(M)}$ not visiting T^M_{err}, then $\kappa \vdash^*_M \kappa'$.*

(iii) *There is exactly one maximal path in $G_{\mathcal{R}(M)}$ that starts in t^M_{in} and does not visit T^M_{err}.*

Proof. (i) Let $\kappa = a_1 \cdots a_k q b_l \cdots b_1$ be a configuration of M. The left branch of $t(\kappa)$ can be generated by using rewriting rules from (3)–(5):

$$a_i \to_{\mathcal{R}} \begin{array}{c} a_i \\ | \\ a_{i+1} \\ | \\ ad_1 \end{array} \to_{\mathcal{R}} \begin{array}{c} a_i \\ | \\ a_{i+1} \\ | \\ ad_2 \end{array} \to_{\mathcal{R}} \begin{array}{c} a_i \\ | \\ a_{i+1} \\ | \\ ad_3 \end{array} \to_{\mathcal{R}} \begin{array}{c} a_i \\ | \\ a_{i+1} \end{array}.$$

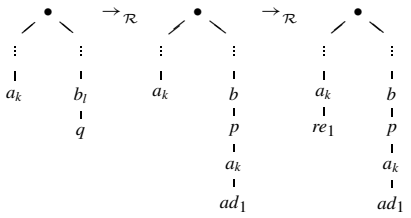
The right branch can be generated by the rewriting rules from (7).

(ii)(a) If $\kappa \vdash_M^* \kappa'$, then the computation steps of M leading from κ to κ' can be simulated by $\mathcal{R}(M)$ as sketched in Figure 5, resulting in the tree $t(\kappa')$.

(b) Let κ, κ' be configurations of M and let π be a path from $t(\kappa)$ to $t(\kappa')$ that does not visit T_{err}^M. We prove the claim by induction on the length of π. Obviously, if π has length 0, then $\kappa = \kappa'$ and therefore $\kappa \vdash_M^* \kappa'$.

So assume that π has length greater than 0 and let $\kappa = a_1 \cdots a_k q b_l \cdots b_1$. We analyze the sequence of rewriting rules that are used on π and show that this sequence has to follow the scheme from Figure 5. Since the definition of $\mathcal{R}(M)$ depends on M we have to analyze two cases. If $\delta(q, b_l) = (p, b, L)$, then $\mathcal{R}(M)$ contains a rule of type (1) that can be applied to the right branch of $t(\kappa)$. We consider this case in more detail below. If $\delta(q, b_l) = (p, b, R)$, then $\mathcal{R}(M)$ contains a rule of type (2) that can be applied to the right branch and one can proceed analogously.

The first rewriting on π has to insert an ad_1 symbol into the tree. If a re_1 or err symbol would be inserted, then the resulting tree would be in T_{err}^M. Assume that the ad_1 symbol is appended to the left branch (using rule (3)). In the next step this ad_1 could be changed into an ad_2 leading to a tree containing an ad_2 symbol and no re symbol, or a rule of type (1) could be applied to the right branch leading to a tree containing two ad_1 symbols. In both cases the resulting tree would be in T_{err}^M. Therefore, in the first step a rewriting rule of type (1) must be used. The only possibility for the second rewriting on π is to add a re_1 symbol to the left branch, using rule (3). If the c that was added to the right branch in the first step does not equal a_k, then the tree after the second step contains the subtrees $\overset{a_k}{\underset{re_1}{|}}$ and $\overset{c}{\underset{ad_1}{|}}$ with $a_k \neq c$ and therefore would be in T_{err}^M. We have shown that the first two steps on π must be of the following form:



Now it is not difficult to see that the next steps on π must lead to $t(\kappa'')$, where κ'' is the successor configuration of κ. By the induction hypothesis we get $\kappa'' \vdash_M^* \kappa'$ and therefore $\kappa \vdash_M^* \kappa'$.

Note that this also proves (iii) because the sequence of rewriting rules described above is the only possibility to avoid T_{err}^M. Hence, starting in t_{in}^M, there is only one maximal path avoiding T_{err}^M. □

After these technical preparations we discuss the remaining reachability problems. All the undecidability proofs use the construction of $\mathcal{R}(M)$ and the set T_{err}^M. Another set of trees used besides T_{err}^M in the following subsections is the set of trees encoding halting configurations of M:

$T_{\text{stop}}^M = \{t(\kappa) \mid \kappa \text{ is a halting configuration of } M\}$.

Since we assumed that the transition function of M is completely specified, a tree codes a halting configuration iff it contains the unique halting state q_s. Therefore, T^M_{stop} is regular.

5.2. *Universal Reachability*

The problem of universal reachability for GTR graphs is the following:

Given: A GTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ and a regular set of trees $T \subseteq T_A$.
Question: Does every maximal path in $G_{\mathcal{R}}$ starting in t_{in} visit T?

Theorem 5. *The problem of universal reachability for GTR graphs is undecidable.*

Proof. Let M be a DTM. Assume that M eventually reaches a halting configuration κ when started on the empty tape, i.e., in the configuration q_{in}. Let π be the unique maximal path starting in t^M_{in} that does not visit T^M_{err}, according to Lemma 13(iii). By Lemma 13(ii)(a) there is a path π' not visiting T^M_{err} from t^M_{in} to $t(\kappa)$. By the choice of π, π' must be a prefix of π and thus π contains a tree from T^M_{stop}, namely $t(\kappa)$. Hence, all paths through $G_{\mathcal{R}(M)}$ starting in t^M_{in} eventually reach $T^M_{\text{err}} \cup T^M_{\text{stop}}$.

Now assume that M never reaches a halting configuration when started on the empty tape. By Lemma 13(iii) there is a maximal path π through $G_{\mathcal{R}(M)}$ starting in t^M_{in} that never reaches T^M_{err}. If this path contains a tree from T^M_{stop}, then by Lemma 13(ii)(b) the corresponding halting configuration of M is reachable from the initial configuration, contradicting the assumption that M does not stop. Thus, not all paths through $G_{\mathcal{R}(M)}$ starting in t^M_{in} eventually reach $T^M_{\text{err}} \cup T^M_{\text{stop}}$.

Therefore, we can conclude that M stops on the empty tape iff every path through $G_{\mathcal{R}(M)}$ starting in t^M_{in} eventually reaches the regular set $T^M_{\text{err}} \cup T^M_{\text{stop}}$. Since the halting problem for deterministic Turing machines is undecidable, the theorem is proven. □

5.3. *Constrained Reachability*

The problem of constrained reachability for GTR graphs is the following:

Given: A GTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ and regular sets $T_1, T_2 \subseteq T_A$.
Question: Does there exist a path π in $G_{\mathcal{R}}$ starting in t_{in} such that there exists $i \in \mathbb{N}$ with $\pi(i) \in T_2$ and $\pi(j) \in T_1$ for all $j \leq i$?

This problem is called constrained reachability because we ask if T_2 can be reached under the constraint that the path remains in T_1 until it reaches T_2. The simple reachability question is the special case with $T_1 = T_A$.

Theorem 6. *The problem of constrained reachability for GTR graphs is undecidable.*

Proof. Let M be a DTM. Assume that M eventually reaches a halting configuration when started on the empty tape. By Lemma 13(ii)(a) this means that there is path through $G_{\mathcal{R}(M)}$ starting in t^M_{in} that reaches T^M_{stop} while staying in the complement of T^M_{err}, which is regular because T^M_{err} is regular (Proposition 2).

If M does not reach a halting configuration when started on the empty tape, then every path through $G_{\mathcal{R}(M)}$ starting in t_{in}^M must visit T_{err}^M before it can reach T_{stop}^M, by Lemma 13(ii)(b).

Thus, M stops on the empty tape iff $\mathcal{R}(M)$ satisfies the constrained reachability problem for $T_{A^M} \setminus T_{\text{err}}^M$ and T_{stop}^M. □

5.4. *Universal Recurrence*

The problem of universal recurrence for GTR graphs is the following:

Given: A GTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ and a regular set $T \subseteq T_A$.
Question: Does every infinite path through $G_{\mathcal{R}(M)}$ that starts in t_{in} infinitely often visit T?

To show the undecidability of universal and constrained reachability we used reductions from the halting problem for DTMs since these two problems allowed us to specify exactly the path that simulates the correct behavior of the Turing machine.

The problem of universal recurrence does not allow this exact specification because we are interested in paths that only finitely often visit a certain regular set. If we take this set to be the set T_{err}^M, then a finite number of errors in the simulation of the DTM are allowed. This finite number of errors can be used to generate an arbitrary configuration of the DTM before starting the correct simulation. Therefore, we use a reduction from the following problem, which we call *diverging configuration*:

Given: A Turing machine M.
Question: Does there exist a configuration κ of M such that M does not stop when started in κ?

Note that in the above problem there is no restriction to reachable configurations of M.

Lemma 14. *The problem "diverging configuration" for DTMs is undecidable.*

Proof (*Sketch*). The proof uses a reduction from the halting problem of DTMs. Given a DTM M one constructs a DTM M' such that M' eventually stops on every configuration if M eventually stops when started on the empty tape. For this purpose M' simulates an increasing number of steps of M, i.e., M' starts simulating one step of M, resets its simulation of M, simulates two steps of M, resets its simulation, and so on. For this purpose M' maintains one counter for the number of steps that have to be simulated after the next reset and a second counter that keeps track of the number of steps that still have to be carried out until the next reset. If the second counter reaches zero, then the simulation is reset and the second counter is reinitialized with the value of the first counter, which itself is increased by one.

Additionally, after each simulation step, M' checks whether its own configuration is well-formed, i.e., if it contains the two counters and the area for simulating M. If the configuration is malformed, then M' stops.

In this way M' will either stop because of a malformed configuration or, after the next reset of the simulation, start to simulate faithfully the behavior of M on the empty

tape. Hence, M' has a diverging configuration if and only if M does not stop on the empty tape.

The details of the sketched construction can be found in [18]. □

Theorem 7. *The problem of universal recurrence for GTR graphs is undecidable.*

Proof. Let M be a DTM. If there is a path π in $G_{\mathcal{R}(M)}$ that only finitely often visits $T^M_{\text{err}} \cup T^M_{\text{stop}}$, then, as can easily be seen from $\mathcal{R}(M)$, there is an $i \in \mathbb{N}$ such that $\pi(i) = t(\kappa)$ for some configuration κ of M and $\pi[i, \infty)$ is an infinite path that does not visit $T^M_{\text{err}} \cup T^M_{\text{stop}}$ at all. With Lemma 13(ii)(b) we can conclude that M does not stop when started in configuration κ.

If there exists a configuration κ of M such that M does not stop when started in κ, then, by Lemma 13(ii)(a), there is an infinite path through $G_{\mathcal{R}(M)}$ starting in $t(\kappa)$ that does not visit $T^M_{\text{err}} \cup T^M_{\text{stop}}$. By Lemma 13(i) we know that there is a path from t_{in} to $t(\kappa)$. The concatenation of these two paths yields an infinite path through $G_{\mathcal{R}(M)}$ starting in t_{in} that visits $T^M_{\text{err}} \cup T^M_{\text{stop}}$ only finitely often.

Therefore, M has no diverging configuration iff $\mathcal{R}(M)$ satisfies the problem of universal recurrence with $T^M_{\text{err}} \cup T^M_{\text{stop}}$. □

6. A Temporal Logic for Model-Checking

We have considered all the reachability problems from Section 3 independently, except for universal one step reachability, which is covered by one step reachability, and universal constrained reachability, which contains universal reachability as a special case. The goal of this section is to build up a logic with operators for the decidable reachability problems. For our logic we use CTL-like syntax. The operators for one step reachability, reachability, and recurrence are denoted by EX, EF, and EGF. The E in these operators stands for "there is a path" such that a certain property on this path holds. Sometimes it is reasonable to restrict to paths built up from edges labeled from a proper subset of Σ. Thus, we allow the parametrization of this existential quantifier E with a set $\Lambda \subseteq \Sigma$. For this purpose we define, given an RGTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ and $\Lambda \subseteq \Sigma$, the RGTRS $\mathcal{R}|_\Lambda = (A, \Sigma, R|_\Lambda, t_{\text{in}})$, where $R|_\Lambda$ is obtained from R by removing all rules $T \overset{\sigma}{\hookrightarrow} T'$ with $\sigma \in \Sigma \backslash \Lambda$.

For a fixed ranked alphabet A and an alphabet Σ for edge labels the formulas of our logic are defined by the following grammar (in CTL-like syntax):

$$\phi ::= T(\mathcal{A}) \mid \neg\phi \mid \phi \vee \phi \mid E_\Lambda X\phi \mid E_\Lambda F\phi \mid E_\Lambda GF\phi$$

for ε-NTA $\mathcal{A}$ and $\Lambda \subseteq \Sigma$.

The semantics $||\phi||_\mathcal{R}$ of such a formula ϕ with respect to an RGTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$ is the set of all trees where ϕ is satisfied with respect to the relation $\rightarrow_\mathcal{R}$.

— $||T(\mathcal{A})||_\mathcal{R} = T(\mathcal{A})$,
— $||\neg\phi||_\mathcal{R} = T_A \backslash ||\phi||_\mathcal{R}$,
— $||\phi_1 \vee \phi_2||_\mathcal{R} = ||\phi_1||_\mathcal{R} \cup ||\phi_2||_\mathcal{R}$,

— $||E_\Lambda X\phi||_\mathcal{R} = \{t \in T_A \mid t \to_{\mathcal{R}|_\Lambda} ||\phi||_\mathcal{R}\}$ $\quad \left[= \text{pre}_{\mathcal{R}_\Lambda}(||\phi||_\mathcal{R})\right]$,
— $||E_\Lambda F\phi||_\mathcal{R} = \{t \in T_A \mid t \to^*_{\mathcal{R}|_\Lambda} ||\phi||_\mathcal{R}\}$ $\quad \left[= \text{pre}^*_{\mathcal{R}_\Lambda}(||\phi||_\mathcal{R})\right]$,
— $||E_\Lambda GF\phi||_\mathcal{R} = \{t \in T_A \mid t \to^\omega_{\mathcal{R}|_\Lambda} ||\phi||_\mathcal{R}\}$.

For an RGTRS $\mathcal{R} = (A, \Sigma, R, t_{\text{in}})$, $t \in T_A$, and a formula ϕ from the above logic we define

$$\mathcal{R}, t \models \phi \quad \text{iff} \quad t \in ||\phi||_\mathcal{R} \qquad \text{and} \qquad \mathcal{R} \models \phi \quad \text{iff} \quad t_{\text{in}} \in ||\phi||_\mathcal{R}.$$

A consequence of the decidability results is the following theorem.

Theorem 8. *For an RGTRS $\mathcal{R}$ and a formula ϕ from the above logic the set $||\phi||_\mathcal{R}$ is a regular set of trees and an automaton $\mathcal{A}_\phi$ accepting $||\phi||_\mathcal{R}$ can be constructed effectively. In particular, it is decidable whether $\mathcal{R} \models \phi$.*

Proof. For the atomic formulas of the form $\phi = T(\mathcal{A})$ the set $||\phi||_\mathcal{R}$ is regular by definition. For the Boolean operators we use Propositions 1 and 2. So we get $\mathcal{A}_{\phi_1 \vee \phi_2} = \mathcal{A}_{\phi_1} \cup \mathcal{A}_{\phi_2}$ and $\mathcal{A}_{\neg\phi} = \overline{\mathcal{A}_\phi}$. For the temporal operators we can define the automata as $\mathcal{A}_{E_\Lambda X\phi} = (\mathcal{A}_\phi)_{\text{pre}_{\mathcal{R}|_\Lambda}}$, $\mathcal{A}_{E_\Lambda F\phi} = (\mathcal{A}_\phi)_{\text{pre}^*_{\mathcal{R}|_\Lambda}}$, and $\mathcal{A}_{E_\Lambda GF\phi} = (\mathcal{A}_\phi)_{\mathcal{R}|_\Lambda,\omega}$. For the construction of $(\mathcal{A}_\phi)_{\mathcal{R}|_\Lambda,\omega}$ we first have to eliminate the ε-transitions from $\mathcal{A}_\phi$ because the algorithm RECUR needs an NTA without ε-transitions as input.

Once we obtained the automaton $\mathcal{A}_\phi$ we can check if t_{in} is in the language $T(\mathcal{A}_\phi)$ to decide whether $\mathcal{R} \models \phi$. □

Although the blow up in the size of the input automaton is polynomial in the constructions for the temporal operators, the iterated application of these constructions results in a blow up that is exponential in the number of nested temporal operators. The construction for the complement automaton is even worse with an exponential blow up, resulting in an automaton with size nonelementary in the number of nested negations. So we have the following complexity.

Theorem 9. *The size of the automaton $\mathcal{A}_\phi$ from Theorem* 8 *is nonelementary in the number of nested negations and exponential in the number of nested temporal operators.*

7. Conclusion

We have analyzed the algorithmic properties of RGTR graphs under the aspect of reachability problems. Our results show that it is still possible to define a logic with a decidable model-checking problem for RGTR graphs that can express nested reachability and fairness (recurrence) properties. On the other hand, several problems that can be solved on the more restricted class of pushdown graphs become undecidable.

Although the undecidability results suggest that the presented fragment of temporal logic is a maximal fragment with a decidable model-checking problem for RGTR graphs, the picture is not yet complete. It is open what kinds of nested temporal formulas can be used under path quantifiers. Examples for such formulas are $\phi_1 = E(GF(T(\mathcal{A}_1)) \wedge$

$GF(T(\mathcal{A}_2)))$ or $\phi_2 = EGF(T(\mathcal{A}_1) \wedge X(T(\mathcal{A}_2)))$. The presented method for solving the recurrence problem cannot be adapted directly to handle conjunctions as in ϕ_1 because for properties of this kind it is necessary to simulate *two* infinite branches of $\lim(\pi)$ rather than one. Perhaps the use of Büchi tree automata for a simulation of substitutions in several directions in the limit tree of infinite paths through the graph provides a possibility of overcoming this difficulty.

Acknowledgments

I thank the editor Joost Engelfriet for his long list of valuable comments, the referees for their suggestions and comments, and Jacques Duparc for the idea of the undecidability proof for the problem "diverging configuration."

References

[1] Achim Blumensath and Erich Grädel. Automatic structures. In *Proceedings of the* 15*th IEEE Symposium on Logic in Computer Science*, *LICS* 2000, pages 51–62. IEEE Computer Society Press, Los Alamitos, CA, 2000.

[2] Walter S. Brainerd. Tree generating regular systems. *Information and Control*, 14:217–231, 1969.

[3] Didier Caucal. On infinite transition graphs having a decidable monadic theory. In *Proceedings of the* 23*rd International Colloquium on Automata*, *Languages and Programming*, *ICALP* '96, pages 194–205. Volume 1099 of Lecture Notes in Computer Science. Springer, Berlin, 1996.

[4] Thomas Colcombet. On families of graphs having a decidable first order theory with reachability. In *Proceedings of the* 29*th International Colloquium on Automata*, *Languages and Programming*, *ICALP* 2002, pages 98–109. Volume 2380 of Lecture Notes in Computer Science. Springer, Berlin, 2002.

[5] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. 1997. `http://www.grappa.univ-lille3.fr/tata`.

[6] Jean-Luc Coquidé, Max Dauchet, Rémi Gilleron, and Sándor Vágvölgyi. Bottom-up tree pushdown automata: classification and connection with rewrite systems. *Theoretical Computer Science*, 127(1):69–98, 1994.

[7] Jean-Luc Coquidé and Rémi Gilleron. Proofs and reachability problem for ground rewrite systems. In *Aspects and Prospects of Theoretical Computer Science*, pages 120–129. Volume 464 of Lecture Notes in Computer Science. Springer, Berlin, 1990.

[8] Max Dauchet, Thierry Heuillard, Pierre Lescanne, and Sophie Tison. Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems. *Information and Computation*, 88(2):187–201, 1990.

[9] Max Dauchet and Sophie Tison. The theory of ground rewrite systems is decidable. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, *LICS* '90, pages 242–248. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[10] E. Allen Emerson. Temporal and modal logic. In J. v. Leeuwen, editor, *Handbook of Theoretical Computer Science*. Volume B, pages 995–1072. Elsevier, Amsterdam, 1990.

[11] Joost Engelfriet. Derivation trees of ground term rewriting systems. *Information and Computation*, 152(1):1–15, 1999.

[12] Javier Esparza, David Hansel, Peter Rossmanith, and Stefan Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of the* 12*th International Conference on Computer Aided Verification*, *CAV* 2000, pages 232–247. Volume 1855 of Lecture Notes in Computer Science. Springer, Berlin, 2000.

[13] Javier Esparza and Astrid Kiehn. On the model checking problem for branching time logics and Basic Parallel Processes. In *Proceedings of the 7th International Conference on Computer Aided Verification*, *CAV '*95, pages 353–366. Volume 939 of Lecture Notes in Computer Science. Springer, Berlin, 1995.

[14] Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[15] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 1979.

[16] Orna Kupferman and Moshe Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *Proceedings of the 12th International Conference on Computer Aided Verification*, *CAV* 2000, pages 36–52. Volume 1855 of Lecture Notes in Computer Science. Springer, Berlin, 2000.

[17] Christof Löding. Model-checking infinite systems generated by ground tree rewriting. In *Proceedings of Foundations of Software Science and Computation Structures*, *FoSSaCS* 2002, pages 280–294. Volume 2303 of Lecture Notes in Computer Science. Springer, Berlin, 2002.

[18] Christof Löding. Infinite Graphs Generated by Tree Rewriting. Ph.D. thesis, RWTH Aachen, 2003.

[19] Richard Mayr. Process rewrite systems. *Information and Computation*, 156(1–2):264–286, 2000.

[20] Christophe Morvan. On rational graphs. In *Proceedings of the Third International Conference on Foundations of Software Science and Computation Structures*, *FoSSaCS '*99, pages 252–266. Volume 1784 of Lecture Notes in Computer Science. Springer, Berlin, 1999.

[21] David E. Muller and Paul E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.

[22] Igor Walukiewicz. Pushdown processes: games and model checking. In *Proceedings of the 8th International Conference on Computer Aided Verification*, *CAV '*96, pages 62–74. Volume 1102 of Lecture Notes in Computer Science. Springer, Berlin, 1996.

Received April 14, 2003 *and in revised form February* 13, 2004, *and in final form June* 2, 2004.
Online publication January 28, 2005.