Recurrent Reachability Analysis in Regular Model Checking

Anthony Widjaja To and Leonid Libkin

LFCS, School of Informatics, University of Edinburgh
anthony.w.to,libkin@ed.ac.uk

Abstract. We consider the problem of recurrent reachability over infinite systems given by regular relations on words and trees, i.e, whether a given regular set of states can be reached infinitely often from a given initial state in the given transition system. Under the condition that the transitive closure of the transition relation is regular, we show that the problem is decidable, and the set of all initial states satisfying the property is regular. Moreover, our algorithm constructs an automaton for this set in polynomial time, assuming that a transducer of the transitive closure can be computed in poly-time. We then demonstrate that transition systems generated by pushdown systems, regular ground tree rewrite systems, and the well-known process algebra PA satisfy our condition and transducers for their transitive closures can be computed in poly-time. Our result also implies that model checking EF-logic extended by recurrent reachability predicate (EGF) over such systems is decidable.

1 Introduction

Infinite-state systems play an important role in verification as they capture many scenarios that cannot be adequately described by standard finite-state models. For example, the behavior of parameterized systems needs to be checked regardless of the number of processes, and this is often most suitably represented by an infinite-state system.

The most common verification problems for such systems can be abstracted as reachability and recurrent reachability [2, 3, 8, 9]. Reachability asks if a given state, or a state in a given set, can be reached from an initial state. Checking these is essential for verifying safety of infinite-state systems, as we want to find counterexamples to specifications saying that bad states cannot be reached. If we have slightly weaker specifications saying that undesirable states can only appear in some initial portion of each execution path, then counterexamples to those are formalized as *recurrent reachability*, i.e. the existence of a witnessing path that infinitely often goes through a given set of states. In the CTL* notation, recurrent reachability for a set L is $\mathbf{EGF}L$. Observe that although for finite systems recurrent reachability is reducible to reachability, this is not the case for infinite systems in general, e.g. lossy channel systems (see [1]).

To make the questions of model checking meaningful for infinite systems, they need to have an effective finite representation. Often, the state space is described

by regular word or tree languages, and transitions are given by regular word or tree transducers: this is the general framework of regular model-checking [2, 3]. Without restrictions, this does not guarantee decidability even for the simplest reachability properties. Hence, one normally restricts the class of transducers so that their iterations would remain regular [7, 17]. Then such infinite-state systems have an effective word-automatic or tree-automatic presentation [5, 6]. Some of the most well-known and most studied classes of such systems include pushdown systems [14, 15, 22, 27], prefix-recognizable graphs [11, 24], ground tree rewrite systems [13, 19], and the process algebra PA [4, 20, 22].

For such systems, reachability has been extensively studied [3, 9, 14–17, 23, 27]. Much less is known about recurrent reachability. Unlike reachability, it is not immediately seen to be decidable even under the assumption that the transitive closure of the transition relation is representable by a regular transducer. One recent result [18, 19] showed that recurrent reachability is decidable for infinite-state transition systems generated by ground tree rewrite systems.

Our main contributions are as follows. We look at arbitrary infinite-state transition systems that have an automatic representation (either word-automatic or tree-automatic) and that further satisfy the condition that the transitive closure of its transition relation is regular. We then show the following:

1. For every regular language L, the set of all states that satisfy a recurrent reachability property $\mathbf{EGF}L$ is also regular. This observation gives rise to two flavors of the model-checking problem: the global problem is to construct an automaton accepting the set of states satisfying $\mathbf{EGF}L$, and the local problem is to verify whether a given word/tree satisfies the property.
2. We give a *generic* poly-time algorithm that solves both model-checking problems for $\mathbf{EGF}L$ given the following as inputs: word/tree regular transducers defining a transition relation and its transitive closure, and a nondeterministic word/tree automaton defining L. For positive answers, our algorithm also constructs some witnessing infinite paths using Büchi word/tree automata as finite representations. In particular, if the transducer defining the transitive closure can itself be computed in poly-time, we obtain a poly-time algorithm for checking recurrent reachability properties. One can also combine our algorithm with the semi-algorithms for computing iterating transducers developed in regular model checking (e.g. [2, 3, 7, 17]).
3. We then look at some particular examples of transition systems in which the transitive closure of the transition relation is regular for which an iterating transducer is poly-time computable. As corollaries, we obtain poly-time algorithms for recurrent reachability over pushdown systems, ground tree rewrite systems, and PA-processes. These also imply that the extension of the **EF**-logic [8, 19, 23] with the **EGF** operator remains decidable for all those examples. For the first two examples, our results follow from known results [15, 18, 19] proven using specialized methods for pushdown systems and ground tree rewrite systems, although their methods do not show how to compute witnessing paths, which are also of interests in verification. Our results for PA-processes are new.

Outline of the paper In Section 2, we recall some basic definitions. In Section 3 we prove our results for transition systems that have word-automatic presentations, and provide applications to pushdown systems. In Section 4 we prove results for tree-automatic presentations, and provide applications to ground tree rewrite systems and PA-processes. We conclude in Section 5 with future work.

2 Preliminaries

Transition systems Let $\mathrm{AP} = \{P_1, \ldots, P_n\}$ be a finite set of *atomic propositions*. A *transition system* over AP is

$$\mathcal{S} \;=\; \langle S,\; \rightarrow,\; \lambda \rangle,$$

where S is a set of *states*, $\rightarrow \subseteq S \times S$ is a *transition relation*, and $\lambda : \mathrm{AP} \rightarrow 2^S$ is a function defining which states satisfy any particular atomic proposition. The set S is not required to be finite.

We write $\rightarrow^+$ (resp. $\rightarrow^*$) to denote the transitive (resp. transitive-reflexive) closure of $\rightarrow$. If $S' \subseteq S$, then $pre^*(S')$ (resp. $post^*(S')$) denotes the set of states s that can reach (resp. be reached from) some state in S'.

Recurrent reachability Given a transition system $\mathcal{S} = \langle S, \rightarrow, \lambda \rangle$ and a set $S' \subseteq S$, we write $s \rightarrow^\omega S'$ iff there exists an infinite sequence $\{s_i\}_{i \in \mathbb{N}}$ such that $s_0 = s$ and $s_i \in S'$ and $s_{i-1} \rightarrow^+ s_i$ for all $i > 0$. By transitivity of $\rightarrow^+$, every infinite subsequence of such a sequence $\{s_i\}_{i \in \mathbb{N}}$ that starts with s_0 is also a witness for $s \rightarrow^\omega S'$. We write $Rec(S')$ to denote the set of states s such that $s \rightarrow^\omega S'$. We will also write $Rec(S', \rightarrow^+)$ to emphasize the transitive binary relation in use.

Words, Trees, and Automata We assume basic familiarity with automata on finite and infinite words and trees (see [12, 25]). Fix a finite alphabet Σ. For each finite word $w = w_1 \ldots w_n \in \Sigma^*$, we write $w[i, j]$, where $1 \leq i \leq j \leq n$, to denote the segment $w_i \ldots w_j$. Given an automaton $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$, a run of $\mathcal{A}$ on w is a function $\rho : \{0, \ldots, n\} \rightarrow Q$ with $\rho(0) = q_0$ that obeys δ. In this case, the length $\|\rho\|$ of ρ is n. The last state $\rho(n)$ appearing in ρ is denoted by $\mathbf{last}(\rho)$; the first state $\rho(0)$ is denoted by $\mathbf{first}(\rho)$. We also define *run segments* to be runs that do not necessarily start from q_0. Given a run segment $\rho' : \{0, \ldots, m\} \rightarrow Q$ such that $\mathbf{first}(\rho') = \mathbf{last}(\rho)$, we may concatenate ρ and ρ' to obtain a new run $\rho \odot \rho' : \{0, \ldots, n + m\} \rightarrow Q$ defined in the obvious way. We also use the notation $\rho[i, j]$ to denote the segment $\rho(i) \ldots \rho(j)$. A run on an ω-word $w \in \Sigma^\omega$ is a function $\rho : \mathbb{N} \rightarrow Q$ with $\rho(0) = q_0$ that obeys δ. We use abbreviations NWA and NBWA for nondeterministic (Büchi) word automata.

Given a finite direction alphabet Υ, a *tree domain* is a non-empty prefix-closed set $D \subseteq \Upsilon^*$. The empty word (denoted by ε) is referred to as the root. Words $u \in D$ so that no ui is in D are called *leaves*. A *tree* T is a pair (D, τ), where D is a tree domain and τ is a node-labeling function mapping D to Σ.

The tree T is said to be *finite* if D is finite; otherwise, it is said to be *infinite*. The tree T is said to be *complete* if, whenever $u \in D$, if $ui \in D$ for some $i \in \Upsilon$, then $uj \in D$ for all $j \in \Upsilon$. If T is infinite, it is said to be *full* if $D = \Upsilon^*$. The set of all finite trees over Υ and Σ is denoted by $\textsc{Tree}_\Upsilon(\Sigma)$. If $\Upsilon = \{1, 2\}$, we write $\textsc{Tree}_2(\Sigma)$ for $\textsc{Tree}_\Upsilon(\Sigma)$.

A (top-down) *tree automaton* $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ over finite Σ-labeled trees has a transition function $\delta : Q \times \Sigma \to 2^{Q^m}$, where $m = |\Upsilon|$. For our constructions, it will be most convenient to define runs on trees with virtual leaf nodes. We define $\mathbf{virt}(T)$ to be the $(\Sigma \cup \bot)$-labeled Υ-tree (D', τ') such that $D' := D \cup \{vi : v \in D, i \in \Upsilon\}$ and if $u \in D$, then $\tau'(u) := \tau(u)$; if $u \notin D$, then $\tau'(u) = \bot$. Notice that $\mathbf{virt}(T)$ is complete. A *run* of $\mathcal{A}$ on T, i.e. a mapping $\rho : D' \to Q$, starts in the initial state q_0 and for each node u labeled a with children $u1, \ldots, um$, we have $(\rho(u1), \ldots, \rho(um)) \in \delta(q, a)$. A run is *accepting* if $\rho(u) \in F$ for each leaf $u \in D'$.

We abbreviate nondeterministic tree automata as NTA, and write NBTA for tree automata over full infinite trees with a Büchi acceptance condition (that will be sufficient for our purposes). For all kinds of automata, $L(\mathcal{A})$ stands for the word or tree languages accepted by $\mathcal{A}$. Also for all types of automata $\mathcal{A}$, we write $\mathcal{A}^q$ for $\mathcal{A}$ in which the initial state is set to q.

Transducers These will be given by *letter-to-letter automata* that accept binary (and, more generally, k-ary) relations over words and trees (cf. [5, 6]). We start with words. Given two words $w = w_1 \ldots w_n$ and $w' = w'_1 \ldots w'_m$ over the alphabet Σ, we define a word $w \otimes w'$ of length $k = \max\{n, m\}$ over alphabet $\Sigma_\bot \times \Sigma_\bot$, where $\Sigma_\bot = \Sigma \cup \{\bot\}$ and $\bot \notin \Sigma$, as follows:

$$w \otimes w' \;=\; \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \ldots \begin{bmatrix} a_k \\ b_k \end{bmatrix}, \text{ where } a_i \;=\; \begin{cases} w_i & i \le n \\ \bot & i > n, \end{cases} \text{ and } b_i \;=\; \begin{cases} w'_i & i \le m \\ \bot & i > m. \end{cases}$$

In other words, the shorter word is padded with $\bot$'s, and the ith letter of $w \otimes w'$ is then the pair of the ith letters of padded w and w'. A letter-to-letter automaton is simply an automaton over $\Sigma_\bot \times \Sigma_\bot$, and a binary relation R over Σ^* is *regular* if the set $\{w \otimes w' : (w, w') \in R\}$ is accepted by a letter-to-letter automaton $\mathcal{R}$. We shall refer to such an automaton as a *transducer* over Σ^*, since it can be alternatively viewed as mapping words $w \in \Sigma^*$ nondeterministically into words w' so that $w \otimes w'$ is accepted by $\mathcal{R}$.

Given two trees $T_1 = (D_1, \tau_1)$ and $T_2 = (D_2, \tau_2)$, we define $T = T_1 \otimes T_2$ as a tree over the labeling alphabet $\Sigma^2_\bot$ similarly to the definition of $w \otimes w'$. That is, the domain of T is $D_1 \cup D_2$, and the labeling $\tau : D_1 \cup D_2 \to \Sigma^2_\bot$ is defined as $\tau(u) = (a_1, a_2)$ so that $a_i = \tau_i(u)$ if $u \in D_i$ and $\bot$ otherwise, for $i = 1, 2$.

As for words, a binary relation R over $\textsc{Tree}_\Upsilon(\Sigma)$ is regular if there is a tree automaton $\mathcal{R}$ over $\textsc{Tree}_\Upsilon(\Sigma^2_\bot)$ accepting the set $\{T_1 \otimes T_2 \mid (T_1, T_2) \in R\}$. We also view it as a transducer that nondeterministically assigns to a tree T_1 any tree T_2 so that $(T_1, T_2) \in R$. If the binary relation R defined by a transducer $\mathcal{R}$ is transitive, we shall refer to $\mathcal{R}$ itself as being transitive.

Automatic transition systems In this paper, we deal with infinite transition systems that can be finitely represented by word or tree automata. We say that a transition system $\mathcal{S} = \langle S, \rightarrow, \lambda \rangle$ over AP is *word-automatic* if, for some finite alphabet Σ, we have $S = \Sigma^*$, the relation $\rightarrow$ is a regular relation on S, and each $\lambda(P_i)$ is a regular subset of S. Likewise, a transition system $\mathcal{S}$ over AP is *tree-automatic* if, for some Υ and Σ, we have $S = \textsc{Tree}_\Upsilon(\Sigma)$, and all of $\rightarrow$ and $\lambda(P_i)$'s are regular tree relations/languages over $\textsc{Tree}_\Upsilon(\Sigma)$.

We measure the size of such a word- or tree-automatic transition system $\mathcal{S}$ as the total size of the transducer for $\rightarrow$, and the automata for S and $\lambda(P_i)$, for $P_i \in \text{AP}$. We shall assume that these are nondeterministic.

As mentioned already, pushdown systems and prefix-recognizable graphs are examples of word-automatic infinite transition systems, while PA-processes and graphs generated by ground tree rewrite systems are examples of tree-automatic transition systems.

3 Recurrent reachability: the word case

We call a word-automatic transition system $\mathcal{S} = \langle S, \rightarrow, \lambda \rangle$ *transitive* if the relation $\rightarrow^+$ is regular. As we shall see shortly, if $\mathcal{S}$ is transitive, then the set $Rec(L)$ is regular too, for an arbitrary regular language L. This gives rise to two variants of the model-checking problem for recurrent reachability: in the global model-checking problem, we are given $\mathcal{S}$ and a language L represented by an NWA $\mathcal{A}$, and we want to construct an NWA accepting $Rec(L(\mathcal{A}))$. In the local version, we also have a word w, and we must check whether $w \in Rec(L(\mathcal{A}))$. That is,

GLOBAL MODEL-CHECKING:	INPUT: 1) A transitive word-automatic $\mathcal{S}$ 2) An NWA $\mathcal{A}$ OUTPUT: A description of $Rec(L(\mathcal{A}))$

LOCAL MODEL-CHECKING:	INPUT: 1) A transitive word-automatic $\mathcal{S}$ 2) An NWA $\mathcal{A}$ 3) a word w OUTPUT: *yes*, if $w \rightarrow^\omega L(\mathcal{A})$ *no*, otherwise

Throughout this section, we assume that the transition relation $\rightarrow$ of transitive $\mathcal{S}$ is given by a transducer $\mathcal{R}$, and that $\mathcal{R}^+$ is the transducer for $\rightarrow^+$ (which exists by the transitivity assumption). We shall also use the transducer for $\rightarrow^*$, denoted by $\mathcal{R}^*$. It can be obtained from $\mathcal{R}^+$ by letting it accept pairs $w \otimes w$.

Theorem 1. *Given a transitive word-automatic transition system $\mathcal{S} = \langle S, \rightarrow, \lambda \rangle$ and an NWA $\mathcal{A}$, the set $Rec(L(\mathcal{A}))$ of states w such that $w \rightarrow^\omega L(\mathcal{A})$ is regular.*

Moreover, if the transducer $\mathcal{R}^+$ for $\rightarrow^+$ is computable in time $t(|\mathcal{R}|)$, then one can compute an NWA recognizing $Rec(L(\mathcal{A}))$ of size $O(|\mathcal{R}^+|^2 \times |\mathcal{A}|)$ in time $t(|\mathcal{R}|) + O(|\mathcal{R}^+|^3 \times |\mathcal{A}|^2)$.

Corollary 2. *Given a transitive word-automatic transition system $\mathcal{S} = \langle S, \to, \lambda\rangle$ and an NWA $\mathcal{A}$, such that the transducer $\mathcal{R}^+$ is poly-time computable, both global and local model-checking for recurrent reachability are solvable in poly-time.*

As another corollary, consider the **EF**, **EX**-fragment of CTL, known as the EF-logic [23, 27]. Its formulae over $\mathrm{AP} = \{P_1, \ldots, P_n\}$ are given by

$$\varphi, \varphi' := \top \mid P_i,\ i \leq n \mid \varphi \vee \varphi' \mid \neg\varphi \mid \mathbf{EX}\varphi \mid \mathbf{EF}\varphi.$$

Each formula, evaluated over a transition system $\mathcal{S} = \langle S, \to, \lambda\rangle$, defines a set $[\![\varphi]\!]_{\mathcal{S}} \subseteq S$ as follows:

(1) $[\![\top]\!]_{\mathcal{S}} = S$; (2) $[\![P_i]\!]_{\mathcal{S}} = \lambda(P_i)$;
(3) $[\![\varphi \vee \varphi']\!]_{\mathcal{S}} = [\![\varphi]\!]_{\mathcal{S}} \cup [\![\varphi']\!]_{\mathcal{S}}$; (4) $[\![\neg\varphi]\!]_{\mathcal{S}} = S \ - \ [\![\varphi]\!]_{\mathcal{S}}$;
(5) $[\![\mathbf{EX}\varphi]\!]_{\mathcal{S}} = \{s \mid \exists s' :\ s \to s' \text{ and } s' \in [\![\varphi]\!]_{\mathcal{S}}\}$;
(6) $[\![\mathbf{EF}\varphi]\!]_{\mathcal{S}} = \{s \mid \exists s' :\ s \to^* s' \text{ and } s' \in [\![\varphi]\!]_{\mathcal{S}}\}$.

If $\to^*$ is given by a regular transducer, then $[\![\varphi]\!]_{\mathcal{S}}$ is clearly effectively regular [6], and so the model-checking problem for EF-logic is decidable. We now extend this to the (EF+EGF)-logic, defined as the extension of EF-logic with the formulae $\mathbf{EGF}\varphi$ with the semantics

$$[\![\mathbf{EGF}\varphi]\!]_{\mathcal{S}} \ = \ \mathit{Rec}([\![\varphi]\!]_{\mathcal{S}}, \to^+) \ = \ \{s \mid s \to^\omega [\![\varphi]\!]_{\mathcal{S}}\}.$$

Theorem 1 extends decidability to (EF+EGF)-logic:

Corollary 3. *If $\mathcal{S} = \langle S, \to, \lambda\rangle$ is a transitive word-automatic transition system such that the transducer $\mathcal{R}^+$ is computable, then for each formula φ of (EF+EGF)-logic, the set $[\![\varphi]\!]_{\mathcal{S}}$ is regular, and an NWA defining $[\![\varphi]\!]_{\mathcal{S}}$ can be effectively constructed.*

We now prove Theorem 1. Throughout the proof, we let $\mathcal{M}$ stand for $\mathcal{R}^+$ and use unambiguous abbreviations such as $\mathit{Rec}(\mathcal{A}, \mathcal{M})$ for $\mathit{Rec}(L(\mathcal{A}), L(\mathcal{M}))$. By definition, we have $w \in \mathit{Rec}(\mathcal{A}, \mathcal{M})$ iff there exists a sequence $\{s_i\}_{i\in\mathbb{N}}$ of words with $s_0 = w$ such that $s_{i-1} \otimes s_i \in L(\mathcal{M})$ and $s_i \in L(\mathcal{A})$ for all $i > 0$. We now divide $\mathit{Rec}(\mathcal{A}, \mathcal{M})$ into two sets $\mathit{Rec}_1(\mathcal{A}, \mathcal{M})$ and $\mathit{Rec}_2(\mathcal{A}, \mathcal{M})$, where $\mathit{Rec}_1(\mathcal{A}, \mathcal{M})$ contains words with a witnessing infinite sequence $\{s_i\}_{i\in\mathbb{N}}$ that satisfies $s_j = s_k$ for some $j < k$, and $\mathit{Rec}_2(\mathcal{A}, \mathcal{M})$ contains words with a witnessing infinite sequence $\{s_i\}_{i\in\mathbb{N}}$ that satisfies $s_j \neq s_k$ for all distinct $j, k \in \mathbb{N}$. We shall write Rec_1 and Rec_2 when the intended automata $\mathcal{A}$ and $\mathcal{M}$ are clear from the context. Now notice that $\mathit{Rec}(L(\mathcal{A})) = \mathit{Rec}_1 \cup \mathit{Rec}_2$. It is easy to construct an NWA that recognizes Rec_1. Observe that, for all word w, we have $w \in \mathit{Rec}_1$ iff there exists a word w' such that $w \to^* w'$, $w' \to^+ w'$, and $w' \in L(\mathcal{A})$. By taking a product and then applying projection (e.g. see [6]), we can compute an NWA $\mathcal{A}_1$ that recognizes Rec_1 in time $O(|\mathcal{R}^+|^2 \times |\mathcal{A}||)$ with $|\mathcal{A}| = O(|\mathcal{R}^+|^2 \times |\mathcal{A}|)$.

Thus, it remains to construct the automaton $\mathcal{A}_2$ for Rec_2. We shall first compute a Büchi automaton $\mathcal{B}$ that recognizes an ω-word which represents the witnessing infinite sequence for membership in Rec_2. Once $\mathcal{B}$ is constructed, it is easy to obtain $\mathcal{A}_2$ as we shall see later. The most obvious representation of the

infinite sequence $\{s_i\}_{i\in\mathbb{N}}$ is $s_0 \otimes s_1 \otimes \ldots$. The problem with this representation is that it requires an infinite alphabet, and possibly infinitely many copies of the automata $\mathcal{A}$ and $\mathcal{M}$ to check whether $s_i \in L(\mathcal{A})$ and $s_{i-1} \rightarrow^+ s_i$ for all $i > 0$. Therefore, the first step towards solving the problem is to analyze the infinite witnessing paths and to show that it is sufficient to consider only infinite sequences of a special form. For the rest of this section, we let $\mathcal{A} = (Q_1, \delta_1, q_0^1, F_1)$ and $\mathcal{M} = \mathcal{R}^+ = (Q_2, \delta_2, q_0^2, F_2)$.

Lemma 4. *For every word $w \in \Sigma^*$, it is the case that $w \in Rec_2(\mathcal{A})$ iff there exist two infinite sequences $\{\alpha_i\}_{i\in\mathbb{N}}$ and $\{\beta_i\}_{i\in\mathbb{N}}$ of words over Σ such that*

1. *$\alpha_0 = w$ and $|\alpha_i| > 0$ for all $i > 0$,*
2. *$|\alpha_i| = |\beta_i|$ for all $i \in \mathbb{N}$,*
3. *there exists an infinite run r of $\mathcal{A}$ on $\beta_0\beta_1\ldots$ such that, for all $i \in \mathbb{N}$, the automaton $\mathcal{A}^q$ accepts α_{i+1}, where $q = r(|\beta_0 \ldots \beta_i|)$,*
4. *there exists an infinite run r' of $\mathcal{M}$ on $(\beta_0 \otimes \beta_0)(\beta_1 \otimes \beta_1)\ldots$ such that, for all $i \in \mathbb{N}$, $\mathcal{M}^q$ accepts $\alpha_i \otimes \beta_i\alpha_{i+1}$ where $q = r'(|\beta_0 \ldots \beta_{i-1}|)$.*

One direction of the lemma is easy: if 1)–4) hold, then from the infinite sequences $\{\alpha_i\}_{i\geq 0}$ and $\{\beta_i\}_{i\geq 0}$ we can form a new sequence $\{s_i\}_{i\geq 0}$ with $s_i := \beta_0 \ldots \beta_{i-1}\alpha_i$. Condition (3) ensures that $s_i \in L(\mathcal{A})$ for all $i > 0$, and condition (4) implies that $s_i \rightarrow^+ s_{i+1}$ for all $i \geq 0$. This implies that $w \in Rec_2(\mathcal{A})$ and thus proving sufficiency in Lemma 4.

The idea of the proof of Theorem 1 is that the sequences $\{\alpha_i\}_{i\geq 0}$ and $\{\beta_i\}_{i\geq 0}$ compactly represent a sequence $\{s_i\}_{i\geq 0}$ witnessing $w \in Rec_2(\mathcal{A})$. We shall later construct a Büchi automaton that recognizes precisely all ω-words of the form

$$(\alpha_0 \otimes \beta_0)\begin{bmatrix}\#\\\#\end{bmatrix}(\alpha_1 \otimes \beta_1)\begin{bmatrix}\#\\\#\end{bmatrix}(\alpha_2 \otimes \beta_2)\begin{bmatrix}\#\\\#\end{bmatrix}\ldots \qquad (*)$$

such that the sequences $\{\alpha_i\}_{i\geq 0}$ and $\{\beta_i\}_{i\geq 0}$ satisfy r.h.s. of Lemma 4. From such an automaton $\mathcal{B}$ it is easy to obtain an automaton recognizing $\alpha_0 = w \in Rec_2$.

Now we shall prove the other direction in Lemma 4: that the sequences $\{\alpha_i\}_{i\geq 0}$ and $\{\beta_i\}_{i\geq 0}$ exist under the assumption $w \in Rec_2(\mathcal{A})$. We will first need to extend the definition of $Rec_2(\mathcal{N}, \mathcal{T})$ to allow not necessarily transitive transducers $\mathcal{T}$: $w \in Rec_2(\mathcal{N}, \mathcal{T})$ iff there exists a sequence $\{s_i\}_{i\geq 0}$ of words such that $s_0 = w$, $s_i \neq s_{i'}$ for all distinct $i, i' \in \mathbb{N}$, $s_i \in L(\mathcal{N})$ for all $i > 0$, and $s_j \otimes s_k \in L(\mathcal{T})$ for all $k > j \geq 0$.

Lemma 5. *Suppose $\mathcal{N}$ and $\mathcal{T}$ are, respectively, an automaton and a transducer over Σ. For every word $w \in \Sigma^*$, if $w \in Rec_2(\mathcal{N}, \mathcal{T})$, then there exists a word $w'w''$ such that*

1. *$|w'| = |w|$ and $|w''| > 0$,*
2. *$w \otimes w'w'' \in L(\mathcal{T})$,*
3. *there exist an accepting run r of $\mathcal{N}$ on $w'w''$, and a run r' of $\mathcal{T}$ on $w' \otimes w'$ such that $w'' \in Rec_2(\mathcal{N}^{q_1}, \mathcal{T}^{q_1'})$, where $q_1 = r(|w|)$ and $q_1' = r'(|w|)$.*

Proof. Suppose that $w \in Rec_2(\mathcal{N}, \mathcal{T})$. Then, there exists an infinite sequence $\sigma = \{s_i\}_{i\in\mathbb{N}}$ such that $s_0 = w$, $s_i \neq s_{i'}$ for all distinct $i, i' \in \mathbb{N}$, and it is the case that, for all $i > 0$, the word s_i is in $L(\mathcal{N})$ with accepting run η_i, and for all distinct pair of indices $i' > i \geq 0$, we have $s_i \otimes s_{i'} \in L(\mathcal{T})$. As there are only finitely many different words of length $|w|$ but infinitely many different words in σ, we may assume that $|s_i| > |w|$ for all $i \geq 1$; for, otherwise, we may simply omit these words from σ. Now every word s_i, where $i > 0$, can be written as $s_i = u_i v_i$ for some words u_i, v_i such that $|u_i| = |w|$ and $|v_i| > 0$. As there are only finitely many different words of length $|w|$ and finitely many different runs of $\mathcal{N}$ of length $|w|$, by pigeonhole principle there must exist $k > 0$ such that $u_j = u_k$ and $\eta_j[0, |w|] = \eta_k[0, |w|]$ for infinitely many $j > 0$. Let $w' := u_k$ and $\eta := \eta_k[0, |w|]$. Therefore, we may discard all words s_i in σ with $i \geq 1$ such that $u_i \neq w'$ or η is not a prefix of η_i. By renaming indices, call the resulting sequence $\sigma = \{s_i\}_{i\in\mathbb{N}}$ and, for all $i \geq 1$, denote by η_i the accepting run of $\mathcal{N}$ on s_i that has η as a prefix. Notice that σ is still a witness for $w \in Rec_2(\mathcal{N}, \mathcal{T})$. So, let $\theta_{j,k}$, where $0 \leq j < k$, be the accepting run of $\mathcal{T}$ on $s_j \otimes s_k$. Let $\mathcal{C}$ be the *finite* set of all runs of $\mathcal{T}$ on $w' \otimes w'$. Notice that it is not necessarily the case that $|\mathcal{C}| = 1$ as $\mathcal{T}$ is nondeterministic. Consider the edge-labeled undirected graph $G = (V, \{E_u\}_{u\in\mathcal{C}})$ such that $V = \mathbb{Z}^+$ and

$$E_u = \{\{j, k\} :\ 0 < j < k \text{ and } u \text{ is a prefix of } \theta_{j,k}\ \}.$$

Notice that $\{E_u\}_{u\in\mathcal{C}}$ is a partition of $\{\{j, k\} : j \neq k, j, k > 0\}$, and so G is a complete graph. By (infinite) Ramsey theorem, G has a monochromatic complete infinite subgraph $H = (V', E_u)$ for some $u \in \mathcal{C}$. Set $r' := u$. Notice that if V' contains the elements $i_1 < i_2 < \ldots$, then θ_{i_j,i_k} with $k > j \geq 1$ has u as a prefix. Therefore, we can discard all words s_i ($i > 0$) in σ such that $i \notin V'$ and by renaming indices call the resulting sequence $\sigma = \{s_i\}_{i\in\mathbb{N}}$. We also adjust the sequence $\{\eta_i\}_{i>0}$ of accepting runs by omitting the appropriate runs and adjusting indices. We now set w'' to be the unique word v such that $s_1 = w'v$. It is easy to see that (1) and (2) are satisfied. Setting $r = \eta_1$, it is easy to check that $w'' \in Rec_2(\mathcal{N}^{q_1}, \mathcal{T}^{q_1'})$ with witnessing sequence $\{t_i\}_{i>0}$, where t_i is the unique word such that $s_i = w't_i$ for all $i > 0$. □

Now it is not difficult to inductively construct the desired sequences $\{\alpha_i\}_{i\geq 0}$ and $\{\beta_i\}_{i\geq 0}$ by using lemma 5 at every induction step. The gist of the proof is that from the word $w'w''$ given by lemma 5 at induction step k, we will set $\beta_k = w'$, $\alpha_{k+1} = w''$, and extend the partial runs r and r' in lemma 4. Notice that we now have $w'' \in Rec_2(\mathcal{N}^{q_1}, \mathcal{T}^{q_1'})$, which sets up the next induction step. See full version for a detailed argument. This completes the proof of lemma 4.

Now we construct a Büchi automaton $\mathcal{B}$ accepting ω-words of the form $(*)$, where α_i's and β_i's are given by Lemma 4. We first give an informal description of how to implement $\mathcal{B}$. The automaton $\mathcal{B}$ will attempt to guess the runs r and r', while at the same time checking that the runs satisfy conditions 3–4 in Lemma 4. To achieve this, $\mathcal{B}$ will run a copy of $\mathcal{A}$ and $\mathcal{M}$, while simultaneously also running a few other copies of $\mathcal{A}$ and $\mathcal{M}$ to check that the runs r and r' guessed so far satisfy conditions 3) and 4) along the way. The automaton $\mathcal{B}$

consists of three components depicted as Boxes 1, 2, and 3 in Figure 1. The first box is used for reading the prefix of the input before the first occurrence of $\begin{bmatrix}\#\\\#\end{bmatrix}$, while the other boxes are used for reading the remaining suffix. Boxes 2–3 are essentially identical, i.e., they have the same sets of states and essentially the same transition functions. When $\mathcal{B}$ arrives in Box 2, it will read a single letter in Σ^2 and goes to Box 3 so as to make sure that $|\alpha_i| > 0$ for each $i > 0$. When $\mathcal{B}$ is in Box 3, it will go to Box 2 upon reading the letter $\begin{bmatrix}\#\\\#\end{bmatrix}$. We will set all states in Box 2 as the final states so as to make sure that infinitely many $\begin{bmatrix}\#\\\#\end{bmatrix}$ is seen, i.e., the sequences $\{\alpha_i\}_i$ and $\{\beta_i\}_i$ are both infinite, and each words α_i and β_i are finite.

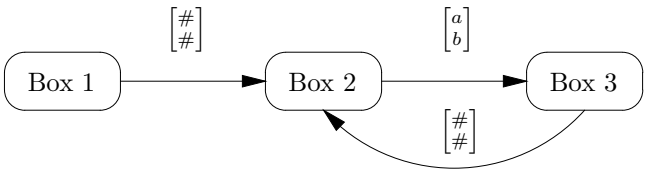

Fig. 1. A bird's eye view of the Büchi automaton $\mathcal{B}$

More formally, the automaton $\mathcal{B} = (\Sigma^2 \cup \{\begin{bmatrix}\#\\\#\end{bmatrix}\}, Q, \delta, q_0, F)$ is defined as follows. We set $Q := (Q_1 \times Q_2 \times Q_2) \uplus (Q_1 \times Q_2 \times Q_1 \times Q_2 \times Q_2 \times \{1, 2\})$, where $Q_1 \times Q_2 \times Q_2$ are the states in box #1, and $Q_1 \times Q_2 \times Q_1 \times Q_2 \times Q_2 \times \{i\}$ are states in box $\#(i+1)$. The initial state is $q_0 := (q_0^1, q_0^2, q_0^2)$. The first and the last components in each state are meant for guessing the infinite runs r and r'. The second component of each state in box #1 is used for guessing a prefix of the accepting run of $\mathcal{M}$ on $\alpha_0 \otimes \beta_0\alpha_1$. The automaton $\mathcal{B}$ will finish this guessing when it reaches box #3 upon the completion of parsing $\alpha_1 \otimes \beta_1$. When $\mathcal{B}$ is presently in box #2 or #3 and reading $\alpha_i \otimes \beta_i$, where $i > 0$, the third and fourth components of the states are used for checking that $\beta_0 \dots \beta_{i-1}\alpha_i \in L(\mathcal{A})$ and $\beta_0 \dots \beta_{i-2}\alpha_{i-1} \otimes \beta_0 \dots \beta_{i-1}\alpha_i \in L(\mathcal{M})$, respectively. At the same time, the second component will be checking that $\beta_0 \dots \beta_{i-1}\alpha_i \otimes \beta_0 \dots \beta_i\alpha_{i+1} \in L(\mathcal{M})$, which will be completed in the next iteration. We now formally define the transition function. We set

$$\delta((q, q', q''), \begin{bmatrix}a\\b\end{bmatrix}) := \begin{cases} \delta_1(q, b) \times \delta_2(q', \begin{bmatrix}a\\b\end{bmatrix}) \times \delta_2(q'', \begin{bmatrix}b\\b\end{bmatrix}) & \text{, if } a, b \neq \# \\ (q, q'', q, q', q'', 1) & \text{, if } a = b = \# \\ \emptyset & \text{, otherwise.} \end{cases}$$

and, when $\mathcal{B}$ is in a state in $Q_1 \times Q_2 \times Q_1 \times Q_2 \times Q_2 \times \{i\}$, where $i = 1, 2$, we define

$$\delta((q_1, q_2, q_1', q_2', q_2'', i), \begin{bmatrix}a\\b\end{bmatrix}) := \delta_1(q_1, b) \times \delta_2(q_2, \begin{bmatrix}a\\b\end{bmatrix}) \times \delta_1(q_1', a) \times \delta_2(q_2', \begin{bmatrix}\perp\\a\end{bmatrix}) \times \delta_2(q_2'', \begin{bmatrix}b\\b\end{bmatrix}) \times \{2\}$$

if $a, b \neq \#$. If $q_1' \in F_1$, and $q_2' \in F_2$, then we set

$$\delta((q_1, q_2, q_1', q_2', q_2'', 2), \begin{bmatrix}\#\\\#\end{bmatrix}) = (q_1, q_2'', q_1, q_2, q_2'', 1).$$

Finally, the set of final states are $F := Q_1 \times Q_2 \times Q_1 \times Q_2 \times Q_2 \times \{1\}$. It is easy to see that $\mathcal{B}$, as claimed, recognizes exactly ω-words of the word of the form $(*)$ such that the sequences $\{\alpha_i\}_{i\in\mathbb{N}}$ and $\{\beta_i\}_{i\in\mathbb{N}}$ satisfy the conditions in Lemma 4.

Now, from $\mathcal{B}$ we can easily compute the automaton $\mathcal{A}_2 = (Q', \Sigma, \delta', q_0', F')$ that recognizes Rec_2. Roughly speaking, the automaton $\mathcal{A}_2$ will accept the set of finite words α_0 such that there exist two sequences $\{\alpha_i\}_{i>0}$ and $\{\beta_i\}_{i\geq 0}$ such that the ω-word $(*)$ is accepted by $\mathcal{B}$. Therefore, we will set the new set of states Q' to be $Q_1 \times Q_2 \times Q_2$, i.e., the first component of $\mathcal{B}$ in Fig. 1. We apply projection operation on the transition function δ of $\mathcal{B}$ to obtain δ'. More formally, if $a \in \Sigma$, we set

$$\delta'((q_1, q_2, q_2'), a) = \bigvee_{b\in\Sigma} \delta((q_1, q_2, q_2'), \begin{bmatrix}a\\b\end{bmatrix}).$$

Finally, the new set F' of final states will those states in Q' from which $\mathcal{B}$ can accept some ω-words of the form $\begin{bmatrix}\#\\\#\end{bmatrix} w$ for some ω-word w. For this, we can apply the standard algorithm for testing nonemptiness for Büchi automata, which takes linear time. Theorem 1 is now immediate. □

Application: Pushdown systems We shall use the definition of [10, 11, 22], which subsumes a more common definition of [14, 15, 27] based on configurations of pushdown automata and transitions between them. A *pushdown system* over the alphabet Σ is given by a finite set Δ of rules of the form $u \to v$ where $u, v \in \Sigma^*$. Let $\mathrm{Dom}(\Delta)$ denote the set of words u for which there is a rule $u \to v$ in Δ. Then Δ generates a relation $\to_\Delta$ over Σ^* as follows: $(w, w') \in \mathcal{R}_\Delta$ iff there exist $x, u, v \in \Sigma^*$ such that $w = xu$, $w' = xv$, and $u \to v$ is a rule in Δ. We thus compute recurrent reachability over pushdown systems $\langle \Sigma^*, \to_\Delta, \lambda\rangle$.

The binary relation $\to_\Delta$ is regular, and can be given by a transducer $\mathcal{R}_\Delta$ whose size is linear in $\|\Delta\|$ (where $\|\Delta\|$ is the sum of the lengths of each word in Δ). Caucal [10] proved that, for each pushdown system Δ, the relation $\to_\Delta^*$ is a poly-time-computable rational transduction[1]. Later in [11] he noted that the given transducer is also regular. For completeness, we sketch how his construction gives a regular transducer $\mathcal{R}_\Delta^*$ for $\to_\Delta^*$ in poly-time. Recall the following well-known proposition, which is proven using the standard "saturation" construction (e.g. see [8, 10, 14]).

Proposition 6. *Given a pushdown system Δ and a nondeterministic automaton $\mathcal{A}$, one can compute two automata $\mathcal{A}_{pre^*}$ and $\mathcal{A}_{post^*}$ for $pre^*(L(\mathcal{A}))$ and $post^*(L(\mathcal{A}))$ in poly-time.*

In fact, the algorithm given in [14] computes the automata in cubic time, and the sizes of $\mathcal{A}_{pre^*}$ and $\mathcal{A}_{post^*}$ are at most quadratic in $|\mathcal{A}|$. To construct $\mathcal{R}_\Delta^*$ using this proposition, we shall need the following easy lemma.

[1] Rational transducers are strictly more powerful than regular transducers.

Lemma 7 ([10]). *Given a pushdown system Δ and two words $u, v \in \Sigma^*$, then $u \rightarrow_\Delta^* v$ iff there exist words $x, y, z \in \Sigma^*$ and word $w \in Dom(\Delta) \cup \{\varepsilon\}$ such that $u = xy$, $v = xz$, $y \rightarrow_\Delta^* w$, and $w \rightarrow_\Delta^* z$.*

Now constructing $\mathcal{R}_\Delta^*$ is easy. First, we use Proposition 6 to compute the automata $\mathcal{A}_{pre^*}^w$ and $\mathcal{A}_{post^*}^w$ that recognize $pre^*(w)$ and $post^*(w)$ for every $w \in \mathrm{Dom}(\Delta) \cup \{\varepsilon\}$. Then, on input $u \otimes v$, the transducer guesses a word $w \in \mathrm{Dom}(\Delta) \cup \{\varepsilon\}$ and a position at which the prefix x in Lemma 7 ends, and then simultaneously runs the automata $\mathcal{A}_{pre^*}^w$ and $\mathcal{A}_{post^*}^w$ to verify that the top part y and the bottom part z of the remaining input word (preceding the $\perp$ symbol) satisfy $y \in L(\mathcal{A}_{pre^*}^w)$ and $z \in L(\mathcal{A}_{post^*}^w)$. We thus obtain a transducer $\mathcal{R}^*$ of size $O(\|\Delta\|^2)$. By taking a product, we compute a transducer $\mathcal{R}^+$ of size $O(\|\Delta\|^3)$ in poly-time. Therefore, Theorem 1 implies the following.

Theorem 8. *Both global and local model-checking for recurrent reachability over pushdown systems are solvable in poly-time.*

That is, for a pushdown system Δ and a nondeterministic automaton $\mathcal{A}$ over an alphabet Σ, one can compute in polynomial time an NWA recognizing $Rec(L(\mathcal{A}), \rightarrow_\Delta^+)$.

4 Recurrent reachability: the tree case

Recall that in a tree-automatic transition system $\mathcal{S} = (S, \rightarrow, \lambda)$, the relation $\rightarrow$ and the sets $\lambda(P_i)$'s are given as tree automata. As in the word case, such a transition system is said to be *transitive* if the relation $\rightarrow^+$ is regular. We now extend our results from Section 3 to transitive tree-automatic transition systems.

Theorem 9. *Given an NTA $\mathcal{A}$ and a transitive tree-automatic transition system $\mathcal{S} = \langle S, \rightarrow, \lambda \rangle$, the set $Rec(L(\mathcal{A}))$ of states $T \in S$ such that $T \rightarrow^\omega L(\mathcal{A})$ is regular. Moreover, if the transducer $\mathcal{R}^+$ for $\rightarrow^+$ is computable in time $t(|\mathcal{R}|)$, then one can compute an NTA recognizing $Rec(L(\mathcal{A}))$ of size $O(|\mathcal{R}^+|^2 \times |\mathcal{A}|)$ in time $t(|\mathcal{R}|) + O(|\mathcal{R}^+|^6 \times |\mathcal{A}|^4)$.*

As in the word case, this implies two corollaries:

Corollary 10. *If $\mathcal{S}$ is transitive and tree-automatic and $\mathcal{R}^+$ is poly-time computable, then both global and local model-checking for recurrent reachability are solvable in poly-time.*

Corollary 11. *If $\mathcal{S} = \langle S, \rightarrow, \lambda \rangle$ is a transitive tree-automatic transition system such that the transducer $\mathcal{R}^+$ is computable, then for each formula φ of (EF+EGF)-logic, the set $[\![\varphi]\!]_\mathcal{S}$ is regular, and an NTA defining $[\![\varphi]\!]_\mathcal{S}$ can be effectively constructed.*

The proof follows the same basic steps as the proof of Theorem 1: we first show that it is sufficient to consider only infinite witnessing sequences that have a representation as an infinite tree over a finite labeling alphabet; we then construct a tree automaton (over infinite trees) with a Büchi acceptance condition that

recognizes such sequences; and from such an automaton we construct a NTA for $Rec(L(\mathcal{A}))$ by applying projection and checking nonemptiness for Büchi tree automata. As checking nonemptiness for Büchi tree automata is quadratic [26] instead of linear as in the word case, the degree of the polynomials in Theorem 9 doubles. While all steps are similar to those in the word case, there are many technical differences; in particular in the coding of an infinite sequence by a single infinite tree. See full version for details of the proof.

Application: Ground tree rewrite systems Ground tree rewrite systems have been intensely studied in the rewriting, automata, and verification communities [12, 13, 18, 19]. We now show that a result by Löding [19] on poly-time model-checking for recurrent reachability and decidability of model checking (EF+EGF)-logic over such systems, which was proved with a specialized method for RGTRSs, can be obtained as a corollary of Theorem 9.

A *ground tree rewrite system* (GTRS) over Σ-labeled Υ-trees is a finite set Δ of transformation rules of the form $t \to t'$ where $t, t' \in \textsc{Tree}_\Upsilon(\Sigma)$. If we permit rules of the form $L \to L'$, where L and L' are tree languages given by some NTAs, then we call Δ a *regular ground tree rewrite system* (RGTRS). Obviously, RGTRSs generalize GTRSs. We define $\|\Delta\|$ as the sum of the sizes of automata in Δ. The RGTRS Δ also generates a binary relation $\to_\Delta$ over $\textsc{Tree}_\Upsilon(\Sigma)$: For a tree T and a node u in it, let T_u be the subtree of T rooted at u. Given two trees T and T', we let $T \to_\Delta T'$ iff there exists a node u in T and a rule $L \to L'$ in Δ such that $T_u \in L$ and $T' = T[t'/u]$ for some $t' \in L'$, where $T[t'/u]$ is the tree obtained from T by replacing the node u by the tree t'.

Given Δ, it is easy to compute a tree transducer $\mathcal{R}_\Delta$ for $\to_\Delta$ in time $O(\|\Delta\|)$; it guesses a node u in the input tree $T \otimes T'$ and a rule in Δ to apply at u in T to obtain T'. The following has been proven in [13] and [12, chapter 3].

Proposition 12. *Given a RGTRS Δ, the transitive closure relation $\to_\Delta^+$ is regular, and a transducer defining it can be computed in time polynomial in $|\mathcal{R}_\Delta|$.*

In fact, the proof for the above proposition constructs "ground tree transducers", which are a subclass of the notion of transducers we are considering in this paper (e.g. see [12, chapter 3]).

Combining this proposition with corollaries 10 and 11, we obtain:

Corollary 13. *(Löding [19]) Both global and local model checking for recurrent reachability over RGTRSs are solvable in poly-time. Model checking (EF+EGF)-logic over RGTRSs with regular atomic predicates is decidable.*

Application: PA-processes PA [4, 22] is a well-known process algebra allowing sequential and parallel compositions, but no communication. It generalizes basic parallel processes (BPP), and context-free processes (BPA), but is incomparable to pushdown processes and Petri nets (e.g. see [22]). PA has found applications in the interprocedural dataflow analysis of parallel programs [16].

We review the basic definitions, following the presentation of [20]: we initially distinguish terms that are equivalent up to simplification laws. The definition of PA usually includes transition labels, which we omit to simplify our presentation (however, the results easily hold when we incorporate transition labels). Fix a finite set $Var = \{X, Y, Z, \ldots\}$ of process variables. *Process terms* over Var, denoted by $\mathcal{F}_{Var}$, are generated by the grammar:

$$t, t' \;:=\; 0 \;\mid\; X, \;\; X \in Var \;\mid\; t.t' \;\mid\; t \| t'$$

where 0 denotes a "nil" process, and $t.t'$ and $t\|t'$ are sequential and parallel compositions, resp. Process terms can be viewed as Σ-labeled binary trees, where $\Sigma = Var \cup \{0, \|, \cdot\}$. In particular, inner nodes are always labeled by '.' or '$\|$', while leaves are labeled by elements in $Var \cup \{0\}$. A PA *declaration* over $\mathcal{F}_{Var}$ is a finite set Δ of rewrite rules of the form $X \to t$, where $X \in Var$, and $t \in \mathcal{F}_{Var}$. We set $\mathrm{Dom}(\Delta) = \{X : (X \to t) \in \Delta, \text{ for some } t \in \mathcal{F}_{Var}\}$, and $Var_\emptyset = Var - \mathrm{Dom}(\Delta)$. The set Δ generates a transition relation $\to_\Delta$ on process terms defined by:

$$\frac{t_1 \to t_1'}{t_1\|t_2 \to t_1'\|t_2} \qquad \frac{t_1 \to t_1'}{t_1.t_2 \to t_1'.t_2} \qquad \frac{}{X \to t}\ (X \to t) \in \Delta$$

$$\frac{t_2 \to t_2'}{t_1\|t_2 \to t_1\|t_2'} \qquad \frac{t_2 \to t_2'}{t_1.t_2 \to t_1.t_2'}\ t_1 \in \mathrm{IsNil}$$

Here IsNil is the set of "terminated" process terms, i.e., those in which all variables are in $Var_\emptyset$. It is easy to see that there is a regular transducer $\mathcal{R}_\Delta$ over process terms for $\to_\Delta$, whose size is linear in the size $\|\Delta\|$ of Δ. It is defined in in the same way as for GTRSs, except that when it guesses a leaf node at which a rule is applied, it must further ensure that v has no '.'-labeled ancestor u such that v is a descendant $u1$ and that T_{u0} is not a terminated process term.

Theorem 14 ([16, 20, 21]). *Given a PA declaration Δ and a NTA $\mathcal{A}$ describing a set of process terms over Var, the sets $pre^*(L(\mathcal{A}))$ and $post^*(L(\mathcal{A}))$ are regular, for which NTAs can be computed in time $O(\|\Delta\| \times |\mathcal{A}|)$, and one can construct a regular transducer $\mathcal{R}^+$ for $\to^+$ in poly-time*[2].

We consider only languages and atomic propositions that are interpreted as regular subsets of $\mathcal{F}_{Var}$. This poses no problem as $\mathcal{F}_{Var}$ is easily seen a regular subset of $\textsc{Tree}_2(\Sigma)$ and no tree $t \in \mathcal{F}_{Var}$ is related by $\to$ to a tree $t' \in \textsc{Tree}_2(\Sigma) - \mathcal{F}_{Var}$. From Theorem 14 and Corollaries 10 and 11, we obtain:

Theorem 15. *Both global and local model checking for recurrent reachability over* **PA** *are solvable in poly-time. Model checking (EF+EGF)-logic over PA is decidable.*

In the study of PA processes, it is common to use a structural equivalence on process terms. We now extend our results to PA modulo structural equivalence.

[2] Lugiez and Schnoebelen first proved this in [20] for a more general notion of transducers, but later in [21] realized that regular transducers suffice.

Let $\equiv$ be the smallest equivalence relation on $\mathcal{F}_{Var}$ that satisfies the following:

$$t.0 \equiv t \qquad 0.t \equiv t \qquad t\|0 \equiv t \qquad t\|t' \equiv t'\|t$$
$$(t\|t')\|t'' \equiv t\|(t'\|t'') \qquad (t.t').t'' \equiv t.(t'.t'')$$

We let $[t]_\equiv$ stand for the equivalence class of t and $[L]_\equiv$ for $\bigcup_{t\in L}[t]_\equiv$. We write $L/\equiv$ for $\{[t]_\equiv |\ t \in L\}$. It was shown in [20] that, for each $t \in \mathcal{F}_{Var}$, $[t]_\equiv$ is a regular tree language, although the set $[L]_\equiv$ need not be regular even for regular L. Given a PA declaration Δ, the equivalence $\equiv$ generates a transition relation $[t]_\equiv \Rightarrow [u]_\equiv$ over $\mathcal{F}_{Var}/\equiv$ which holds iff there exist $t' \in [t]_\equiv$ and $u' \in [u]_\equiv$ such that $t' \to u'$. We need the following result:

Lemma 16 ([20]). *The relation $\equiv$ is bisimulation: for all $t, t', u \in \mathcal{F}_{Var}$, if $t \equiv t'$ and $t \to u$, then there exists $u' \in \mathcal{F}_{Var}$ such that $t' \to u'$ and $u \equiv u'$.*

Now it is not hard to show that, for every NTA $\mathcal{A}$, the set $Rec(L(\mathcal{A}))$ is closed under $\equiv$, if $L(\mathcal{A})$ is closed under $\equiv$. This also implies that $Rec(L(\mathcal{A})) = [Rec(L(\mathcal{A}))]_\equiv = \{t : t \in Rec(L(\mathcal{A})/\equiv, \Rightarrow^+)\}$. In the following, we consider only languages that are closed under $\equiv$.

Theorem 17. *Given an NTA $\mathcal{A}$ such that $L(\mathcal{A})$ is closed under $\equiv$ and a process term $t \in \mathcal{F}_{Var}$, it is possible to decide whether $[t]_\equiv \Rightarrow^\omega L(\mathcal{A})/\equiv$ in PTIME.*

Since $Rec(L(\mathcal{A})) = [Rec(L(\mathcal{A}))]_\equiv$, we need only compute an NTA for $Rec(L(\mathcal{A}))$ and test whether $t \in Rec(L(\mathcal{A}))$. These can be done in PTIME by theorem 15.

We now move to model checking (EF+EGF)-logic over PA modulo $\equiv$. Suppose $\mathcal{S} = \langle S, \to, \lambda\rangle$ is a transition system generated by some PA-declaration and that each $\lambda(P)$ is closed under $\equiv$. In fact, the standard atomic propositions for PA-processes include sets of process terms of the form $[t]_\equiv$ and *action-based predicates*, i.e., sets of all terms t in which some transitions in Δ can be applied (and these are obviously closed under $\equiv$ and of size $O(\|\Delta\|)$). Now Lemma 16 implies that $[\![\varphi]\!]_\mathcal{S}$ is closed under $\equiv$ for (EF+EFG)-formulae φ, and we obtain:

Theorem 18. *The problem of model checking for (EF+EGF)-logic over PA modulo $\equiv$ is decidable whenever all atomic propositions are closed under $\equiv$.*

5 Future work

We mention some possible future work. We would like to further study algorithmic improvements of our general technique, e.g., in its current form it gives a polynomial of degree higher than the specialized technique of [19] for RGTRSs. We would also like to investigate stronger but unrestrictive conditions that ensure decidability of stronger logics (e.g. CTL*) within our framework; it is easy to show that our current condition is insufficient. Finally, we would like to study when our technique could generate elementary complexity algorithms for (EF+EGF)-logic, or just EF-logic alone. This problem is still open even for PA-processes and GTRSs [19, 23].

Acknowledgments We thank Richard Mayr and anonymous referees for their helpful comments. The authors were supported by EPSRC grant E005039, the second author also by EC grant MEXC-CT-2005-024502.

References

1. P. A. Abdulla, and B. Jonsson. Undecidable verification problems for programs with unreliable channels In *Inf. Comput.*, 130(1):71–90, 1996.
2. P. A. Abdulla, B. Jonsson, P. Mahata, and J. d'Orso. Regular tree model checking. In *CAV '02*, pages 555–568.
3. P. A. Abdulla, B. Jonsson, M. Nilsson, and M. Saksena. A survey of regular model checking. In *CONCUR'04*, pages 35–48.
4. J. Baeten and W. Weijland. *Process Algebra*. CUP, 1990.
5. M. Benedikt, L. Libkin, and F. Neven. Logical definability and query languages over ranked and unranked trees. *ACM Trans. Comput. Logic*, 8(2):11, 2007.
6. A. Blumensath and E. Grädel. Automatic structures. In *LICS '00*, pages 51–60.
7. B. Boigelot, A. Legay, and P. Wolper. Iterating transducers in the large. In *CAV'03*, pages 223–235.
8. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR '97*, pages 135–150.
9. A. Bouajjani, A. Legay, and P. Wolper. Handling liveness properties in (ω-)regular model checking. In *INFINITY*, pages 101–115, 2004.
10. D. Caucal. On the regular structure of prefix rewriting. In *CAAP'90*, pages 61–86.
11. D. Caucal. On the regular structure of prefix rewriting. *Theor. Comput. Sci.*, 106(1):61–86, 1992.
12. H. Comon et al. *Tree Automata: Techniques and Applications*. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007.
13. M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *LICS'90*, pages 242–248.
14. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *CAV '00*, pages 232–247.
15. J. Esparza, A. Kucera, and S. Schwoon. Model checking LTL with regular valuations for pushdown systems. *Inf. Comput.*, 186(2):355–376, 2003.
16. J. Esparza and A. Podelski. Efficient algorithms for pre* and post* on interprocedural parallel flow graphs. In *POPL '00*, pages 1–11. USA, 2000. ACM.
17. B. Jonsson and M. Nilsson. Transitive closures of regular relations for verifying infinite-state systems. In *TACAS'00*, pages 220–234.
18. C. Löding. Model-checking infinite systems generated by ground tree rewriting. In *FoSSaCS'02*, pages 280–294.
19. C. Löding. Reachability problems on regular ground tree rewriting graphs. *Theory Comput. Syst.*, 39(2):347–383, 2006.
20. D. Lugiez and P. Schnoebelen. The regular viewpoint on PA-processes. *Theor. Comput. Sci.*, 274(1-2):89–115, 2002.
21. D. Lugiez and P. Schnoebelen. Decidable first-order transition logics for PA-processes. *Inf. Comput.*, 203(1):75–113, 2005.
22. R. Mayr. Process rewrite systems. *Inf. Comput.*, 156(1-2):264–286, 2000.
23. R. Mayr. Decidability of model checking with the temporal logic EF. *Theor. Comp. Sci.*, 256(1-2):31–62, 2001.
24. W. Thomas. Constructing infinite graphs with a decidable MSO-theory. In *MFCS*, pages 113–124, 2003.
25. M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Proc. Banff Higher-Order Workshop*, pages 238–266.
26. M. Y. Vardi, P. Wolper. Automata-theoretic techniques for modal logics of programs. *JCSS* 32(2): 183–221 (1986).
27. I. Walukiewicz. Model checking CTL properties of pushdown systems. In *FSTTCS'00*, pages 127–138.