The Complexity of Verifying Ground Tree Rewrite Systems

Stefan Göller
Universität Bremen, Institut für Informatik

Anthony Widjaja Lin
Oxford University Computing Laboratory

***Abstract*—Ground tree rewrite systems (GTRS) are an extension of pushdown systems with the ability to spawn new subthreads that are hierarchically structured. In this paper, we study the following problems over GTRS: (1) model checking EF-logic, (2) weak bisimilarity checking against finite systems, and (3) strong bisimilarity checking against finite systems. While they are all known to be decidable, we show that problems (1) and (2) have nonelementary complexity, whereas problem (3) is shown to be in coNEXP by finding a syntactic fragment of EF whose model checking complexity is complete for $\mathsf{P}^{\mathsf{NEXP}}$. The same problems are studied over a more general but decidable extension of GTRS called regular GTRS (RGTRS), where regular rewriting is allowed. Over RGTRS we show that all three problems have nonelementary complexity. We also apply our techniques to problems over PA-processes, a well-known class of infinite systems in Mayr's PRS (Process Rewrite Systems) hierarchy. For example, strong bisimilarity checking of PA-processes against finite systems is shown to be in coNEXP, yielding a first elementary upper bound for this problem.**

I. Introduction

Pushdown systems (PDS) are natural abstractions of sequential programs with unbounded recursions. The problems of verifying pushdown systems have hitherto been well-studied (cf. [2], [25], [26]). In addition to recursions, concurrency is indisputably another feature that commonly arises in real-world programs. Multithreading is often introduced by design, e.g., to achieve speedup or to make programming more convenient.

Ground tree rewrite systems (GTRS) (cf. [4], [8], [9], [14]), which were also studied under the name ground term rewrite systems in the rewriting community, are an extension of PDS with the ability to spawn new subthreads that are hierarchically structured, which in turn may terminate and return some values to their parents. While the rules of a PDS rewrite a prefix of a given word, the rules of a GTRS rewrite a *subtree* of a given tree. The decidability status for many standard verification (i.e. model checking and equivalence checking) problems over GTRS is well-known. For example, reachability, recurrent reachability, and fair termination are decidable (cf. [4], [7], [14], [22]). Moreover, model checking first-order logic with reachability predicates is decidable [9], which implies the decidability of model checking the common fragment of Computation Tree Logic (CTL) known as EF-logic. In turn, by a reduction to EF-logic (as shown in [11]), we also obtain the decidability of the problems of weak/strong bisimilarity checking against finite systems over GTRS. In fact, most of these decidability results are known to hold for regular GTRS (RGTRS) [14], which are a well-known extension of GTRS with more general rewrite rules that are given by tree automata[1]. On the negative side, it is known that most linear-time and branching-time logics — such as LTL and CTL — have undecidable model checking problems over GTRS (cf. [14], [24]). This is in stark contrast with pushdown systems, over which model checking monadic second-order logic is still decidable [17].

The precise complexities of some verification problems over GTRS and extensions thereof are also known. For example, reachability and recurrent reachability are polynomial time solvable even for RGTRS (cf. [14]). In contrast, model checking first-order logic with reachability predicates over GTRS has nonelementary complexity [21] since the infinite binary tree with a descendant can be easily generated by a fixed GTRS (in fact, by a one-state pushdown system). The precise complexity of EF-logic model checking over GTRS (and extensions thereof) was stated as an open question in [14]. The best known upper bound is currently nonelementary, while the best known lower bound is only PSPACE (which holds already for pushdown systems [2], [26]). Likewise, for the problems of strong/weak bisimilarity checking against finite systems over GTRS (and RGTRS), the best known upper bound is nonelementary, while the best known lower bound is only PSPACE (which holds already for pushdown systems [13], [20]). Interestingly, the same nonelementary gaps are also currently present (cf. [20]) when these three problems are considered over similar infinite-state models like PA and PAD processes, which are well-known classes of infinite systems in Mayr's PRS (Process Rewrite Systems) hierarchy (cf. [16]). Note that these three problems are only PSPACE-complete over pushdown systems (cf. [2], [13], [20], [26]).

Contributions. We investigate the following verification problems over GTRS (and the extension RGTRS): (1) model checking EF-logic, (2) weak bisimilarity checking against finite systems, and (3) strong bisimilarity checking against finite systems. Such problems are arguably the most basic verification problems over infinite-state systems, especially in the concurrent setting (cf. [16]). Our main contribution is to pinpoint the complexity of these problems.

The starting point of our paper is a proof that EF-logic model checking over GTRS has a nonelementary complexity, already when considering EF formulas with two occurrences

[1]This extension is analogous to how *prefix-recognizable systems* [5] extend PDS.

of EF operators that are nested. This shows that the existing automata-based algorithms for the problem (cf. [7], [9], [14]) are in some sense optimal answering Löding's open question [14]. The lower bound proof is achieved by an exponential reduction from the decidable first-order theory over finite words, which is well-known to have a nonelementary complexity [21]. With the same arguments one can also show that Hennesy-Milner logic (i.e. the fragment with no EF operators) suffice to show the nonelementary lower bound over the more general class of RGTRS.

We then proceed to look at the fragment EF_1 of EF-logic consisting of formulas with EF operator nesting depth at most one (i.e. in the parse tree of the formula, every branch has at most one occurence of the operator EF). This fragment is interesting for two reasons. Firstly, as mentioned above, our proof of the nonelementary lower bound for problem (1) over GTRS requires precisely two nested occurrences of EF operators. Secondly, there is a polynomial time reduction from problem (3) to the problem of model checking EF_1 formulas over GTRS if the formulas are represented as DAGs, which are single-exponentially more succinct than the standard tree representation of formulas. Our result is that the problem of model checking EF_1 over GTRS is $\mathsf{P}^{\mathsf{NEXP}}$-complete (i.e. within the second level of the exponential hierarchy). This result cannot be obtained by simply applying the existing automata-based algorithms for EF-logic model checking (cf. [7], [9], [14]). Moreover, a further analysis of our proof shows that problem (3) is solvable in coNEXP. This has substantially closed the nonelementary gap with the best known lower bound for the problem, which is PSPACE. In fact, these proof techniques can be easily applied to derive better upper/lower bounds for verification problems PA-processes: (1) strong bisimilarity checking of PA-processes against finite systems is solvable in coNEXP giving the first elementary upper bound for this problem (cf. [20]), and (2) model checking EF-logic over PA-processes is $\mathsf{P}^{\mathsf{NEXP}}$-hard improving the best known lower bound of PSPACE for the problem (cf. [16]).

We then consider two natural extensions of the problem of checking strong bisimilarity against finite systems over GTRS: (i) checking strong bisimilarity against finite systems over the more general class RGTRS, and (ii) checking weak bisimilarity against finite systems over GTRS. In contrast, it turns out that both of these extensions have a nonelementary complexity. These results are the most technically involved in this paper with the lower bound proof of problem (ii) building upon the lower bound proof of problem (i). Bisimilarity checking has a standard interpretation as a game between two players (cf. [13]) called Attacker (who aims to show non-bisimilarity) and Defender (who aims to show bisimilarity). The difficulty of proving a complexity lower bound for bisimilarity checking problem is due to the asymmetry between the power of Attacker and Defender (Attacker is often more powerful) in such games. Known lower bound techniques for bisimilarity checking, a.k.a. "Defender's forcing", are often implemented by the help of finite control unit, which many infinite-state models have (e.g. PDS and Petri nets). The difficulty of providing Defender's forcing techniques in the absence of finite control unit is witnessed by the plethora of open problems concerning decidability/complexity of equivalence checking over infinite-state models like PA and PAD processes (cf. [20]). The lack of global finite control unit often means that Defender does not have an *immediate* way of punishing Attacker (i.e. forcing him not to do something bad). In the case of GTRS or RGTRS, this means that at any given time Attacker may replace any of (potentially unbounded number of) the subtrees that are present in the current configuration (i.e. a tree). In this paper, we provide the first methods for implementing Defender's forcing technique over infinite-state models that lack finite-control unit resulting in strong (i.e. nonelementary) lower bounds.

Our results for (R)GTRS are summarized in Table I.

Related work. Other related problems over GTRS have been shown to be decidable: confluence (cf. [8]), model checking fragments of LTL [23], and generalized recurrent reachability [24].

Several other extensions of PDS with multithreading capabilities have been considered in [3], [12], [16], [19]. Among these extensions, the class of process rewrite systems [16], which generalize both Petri nets and pushdown systems by providing hierarchical structures to threads, seem to have tight connections with GTRS. It is interesting to note that there is also a nonelementary gap between the best known upper/lower bounds for EF-logic model checking and strong/weak bisimilarity checking against finite systems over two proper subclasses of process rewrite systems known as PA-processes and PAD-processes (cf. [15], [16], [20]). PA-processes have also been shown to be a good abstraction of parallel programs for the purpose of interprocedural data-flow analysis (cf. [10]).

Organization. Section II contains preliminaries. In Section III, we analyze the complexity of model checking EF-logic (and its fragments). We prove a nonelementary lower bound for strong bisimilarity checking against finite systems in Section IV. Results concerning weak bisimilarity checking against finite systems for GTRS are discussed in Section V. Section V depends on Section IV which in turn depends on Section III-A.

For detailed proofs and applications to PA-processes, see our technical report.

II. Preliminaries

For each $i, j \in \mathbb{Z}$, we denote by $[i, j]$ the interval $\{i, i+1, \ldots, j-1, j\}$. Throughout the rest of this paper, we fix a countable set of *action labels* Act. For a set X let 2^X denote the *powerset* of X.

Complexity theory: The following complexity classes will be of relevance: P, AP $=$ PSPACE, NEXP, coNEXP, k-EXP, ELEMENTARY $= \bigcup_k k$-EXP. The class $\mathsf{P}^{\mathsf{NEXP}}$ denotes deterministic polynomial time with oracle access to NEXP. It is in the second level of exponential hierarchy, which is in EXPSPACE. Unless stated otherwise *reductions* are always

Model checking	GTRS	RGTRS
EF_0	PSPACE-complete	
EF_1	$\mathsf{P}^{\mathsf{NEXP}}$-complete	
$\mathsf{EF}_{k,\ k\geq 2}$	NONELEMENTARY	

Bisimilarity against finite systems	GTRS	RGTRS
$\sim$	PSPACE $\cdots$ coNEXP	
$\approx$	NONELEMENTARY	

TABLE I
OVERVIEW OF OUR RESULTS

polynomial time many-to-one reductions. We define the tower function $\mathsf{TOWER} : \mathbb{N} \to \mathbb{N}$ as $\mathsf{TOWER}(0) = 1$ and $\mathsf{TOWER}(n+1) = 2^{\mathsf{TOWER}(n)}$ for each $n \in \mathbb{N}$.

First-order logic over words: A (binary) *word* is a finite sequence $a_1 a_2 \dots a_n$, where $a_i \in \{0,1\}$ for each $i \in [1,n]$ that we also identify with the logical structure $\overline{w} = (U, P_0, P_1, <)$, where $U = [1,n]$ is the universe, unary predicates $P_a = \{i \in [1,n] \mid a_i = a\}$ for each $a \in \{0,1\}$ and the binary predicate $<$. By a result of Stockmeyer the *first-order theory over words* is nonelementary [21]. We assume in this paper that first-order formulas are given in prenex normal form.

Transition systems and EF logic: A *transition system* is a tuple $\mathcal{T} = (C, \mathbb{A}, \{\xrightarrow{a} \mid\ a \in \mathbb{A}\})$, where C is a set of *configurations*, $\mathbb{A} \subseteq \mathsf{Act}$ is some finite set of *actions*, and $\xrightarrow{a} \subseteq C \times C$ is a set of transitions for each action $a \in \mathbb{A}$. Let us fix a subset $\Sigma \subseteq \mathbb{A}$ of $\mathcal{T}$'s actions. By $\xrightarrow{\Sigma}$ we abbreviate $\bigcup_{a\in\Sigma} \xrightarrow{a}$. We prefer to use the infix notation $c \xrightarrow{a} d$ instead of $(c,d) \in \xrightarrow{a}$. Similar remarks apply to $\xrightarrow{\Sigma}$. We say that $\mathcal{T}$ is *finite* in case C is finite – we often use the notation $\mathfrak{F}$ to refer to finite transition systems. Formulas of EF-logic are given by the following grammar, where $\Sigma \subseteq \mathbb{A}$: $\varphi ::= \mathtt{true} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\Sigma\rangle\varphi \mid \langle\Sigma^*\rangle\varphi$. We introduce the usual abbreviations $\mathtt{false} = \neg\mathtt{true}$, $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $[\Sigma]\varphi = \neg\langle\Sigma\rangle\neg\varphi$, and $[\Sigma^*]\varphi = \neg\langle\Sigma^*\rangle\neg\varphi$. We note that all of our lower bound proofs also hold for the more restricted version of EF logic where each occurence of the operator $\langle\Sigma^*\rangle$ satisfies $\Sigma = \mathbb{A}$. For each transition system $\mathcal{T} = (C, \mathbb{A}, \{\xrightarrow{a} \mid\ a \in \mathbb{A}\})$ and each EF-formula φ define the set of all configurations $[\![\varphi]\!]_\mathcal{T} \subseteq C$ that satisfy φ by induction on the structure of φ as $[\![\mathtt{true}]\!]_\mathcal{T} = C$, $[\![\neg\varphi]\!]_\mathcal{T} = C \setminus [\![\varphi]\!]_\mathcal{T}$, $[\![\varphi_1 \wedge \varphi_2]\!]_\mathcal{T} = [\![\varphi_1]\!]_\mathcal{T} \cap [\![\varphi_2]\!]_\mathcal{T}$, $[\![\langle\Sigma\rangle\varphi]\!]_\mathcal{T} = \{c \in C \mid \exists d \in [\![\varphi]\!]_\mathcal{T} : c \xrightarrow{\Sigma} d\}$, $[\![\langle\Sigma^*\rangle\varphi]\!]_\mathcal{T} = \{c \in C \mid \exists d \in [\![\varphi]\!]_\mathcal{T} : c \xrightarrow{\Sigma}{}^* d\}$. We write $(\mathcal{T}, c) \models \varphi$ whenever $c \in [\![\varphi]\!]_\mathcal{T}$. The EF *nesting depth* $\mathsf{nd}(\varphi)$ of an EF-formula φ is inductively defined as follows: $\mathsf{nd}(\mathtt{true}) = 0$, $\mathsf{nd}(\neg\varphi) = \mathsf{nd}(\varphi)$, $\mathsf{nd}(\varphi_1 \wedge \varphi_2) = \max\{\mathsf{nd}(\varphi_1), \mathsf{nd}(\varphi_2)\}$, $\mathsf{nd}(\langle\Sigma\rangle\varphi) = \mathsf{nd}(\varphi)$, $\mathsf{nd}(\langle\Sigma^*\rangle\varphi) = \mathsf{nd}(\varphi)+1$. For each $i \geq 0$, we denote by EF_i the syntactic fragment of EF restricted to formulas of EF nesting depth at most i. We remark that EF_0 is Henessy-Milner logic (HM).

Strong bisimulation equivalence: Let $\mathcal{T}_1 = (C_1, \mathbb{A}, \{\xrightarrow{a}_1 \mid\ a \in \mathbb{A}\})$ and $\mathcal{T}_2 = (C_2, \mathbb{A}, \{\xrightarrow{a}_2 \mid\ a \in \mathbb{A}\})$ be two transition systems over a common set of actions $\mathbb{A}$. A relation $R \subseteq C_1 \times C_2$ is a *strong bisimulation* if for each $(c_1, c_2) \in R$ the following two conditions hold for each $a \in \mathbb{A}$: (1) for every $c_1 \xrightarrow{a}_1 c_1'$ there is some $c_2 \xrightarrow{a}_2 c_2'$ with $(c_1', c_2') \in R$ and (2) for every $c_2 \xrightarrow{a}_2 c_2'$ there is some $c_1 \xrightarrow{a}_1 c_1'$ with $(c_1', c_2') \in R$. We say that c_1 is *strongly bisimilar* to c_2 (abbreviated by $c_1 \sim c_2$) whenever there is a strong bisimulation R such that $(c_1, c_2) \in R$.

Weak bisimulation equivalence: Let us fix a *silent action* $\tau \notin \mathbb{A}$ and let $\mathbb{A}_\tau = \mathbb{A} \cup \{\tau\}$. Moreover let $\mathcal{T}_1 = (C_1, \mathbb{A}_\tau, \{\xrightarrow{a}_1 \mid\ a \in \mathbb{A}_\tau\})$ and $\mathcal{T}_2 = (C_2, \mathbb{A}_\tau, \{\xrightarrow{a}_2 \mid\ a \in \mathbb{A}_\tau\})$ be two transition systems over the common alphabet $\mathbb{A}_\tau$. We define the binary relations $\overset{\tau}{\Longrightarrow}_i = (\xrightarrow{\tau}_i)^*$ and $\overset{a}{\Longrightarrow}_i = (\xrightarrow{\tau}_i)^* \circ \xrightarrow{a}_i \circ (\xrightarrow{\tau}_i)^*$ for each $a \in \mathbb{A}$ and for each $i \in \{1,2\}$. A binary relation $R \subseteq C_1 \times C_2$ is a *weak bisimulation* if for each $(c_1, c_2) \in R$ the following two conditions hold for each $a \in \mathbb{A}_\tau$: (1) for every $c_1 \xrightarrow{a}_1 c_1'$ there is some $c_2 \overset{a}{\Longrightarrow}_2 c_2'$ with $(c_1', c_2') \in R$ and (2) for every $c_2 \xrightarrow{a}_2 c_2'$ there is some $c_1 \overset{a}{\Longrightarrow}_1 c_1'$ with $(c_1', c_2') \in R$. We say that c_1 is *weakly bisimilar* to c_2 (abbreviated by $c_1 \approx c_2$) whenever there is a weak bisimulation R such that $(c_1, c_2) \in R$. In the appendix we describe how bisimilarity corresponds to a game between Attacker and Defender.

Ranked trees: Let $\preceq$ denote the prefix order on $\mathbb{N}^*$, i.e. $x \preceq y$ for $x, y \in \mathbb{N}^*$ if there is some $z \in \mathbb{N}^*$ such that $y = xz$, and $x \prec y$ if $x \preceq y$ and $x \neq y$. A *ranked alphabet* is a collection of pairwise disjoint finite alphabets $A = (A_i)_{i\in[k]}$ for some $k \geq 0$. For simplicity we identify A with $\bigcup_{i\in[k]} A_i$. A *(ranked) tree* (over the ranked alphabet A) is a mapping $T : D_T \to A$, where for $D_T \subseteq [1,k]^*$ we have (1) D_T is nonempty, finite and prefix-closed and (2) for each $x \in D_T$ with $T(x) \in A_i$ we have $x1, \dots, xi \in D_T$ and $xj \notin D_T$ for each $j > i$. We say that D_T is the *domain* of T – we call these elements *nodes*. A *leaf* is a node x with $T(x) \in A_0$. We also refer to $\varepsilon \in D_T$ as the *root* of T. By Trees_A we denote the set of all ranked trees over the alphabet A. Let T be a ranked tree and let x be a node of T. We define $xD_T = \{xy \in [1,k]^* \mid y \in D_T\}$ and $x^{-1}D_T = \{y \in [1,k]^* \mid xy \in D_T\}$. By $T^{\downarrow x}$ we denote the *subtree of* T with root x, i.e. the tree with domain $D_{T^{\downarrow x}} = x^{-1}D_T$ defined as $T^{\downarrow x}(y) = T(xy)$. Let $S, T \in \mathsf{Trees}_A$ and let x be a node of T. We define $T[x/S]$ to be the tree that is obtained by replacing $T^{\downarrow x}$ in T by S, more formally $D_{T[x/S]} = (D_T \setminus xD_{T^{\downarrow x}}) \cup xD_S$ with $T[x/S](y) = T(y)$ if $y \in D_T \setminus xD_{T^{\downarrow x}}$ and $S(z)$ if $y = xz$ with $z \in D_S$. Define $|T|$ as the number of nodes in a tree T. We also use the notation t to refer to ranked trees, in particular subtrees.

Regular tree languages: A *nondeterministic (bottom-up) tree automaton (NTA)* is a tuple $\mathcal{A} = (Q, F, A, \Delta)$, where Q is a

finite set of *states*, $F \subseteq Q$ is a set of *final states*, $A = (A_i)_{i\in[k]}$ is a ranked alphabet, and $\Delta \subseteq \bigcup_{i\in[k]} Q^i \times A_i \times Q$ is the *transition relation*. By $L(\mathcal{A}) = \{T \in \mathsf{Trees}_A \mid \text{there is an accepting run of } \mathcal{A} \text{ on } T\}$. A set of trees $S \subseteq \mathsf{Trees}_A$ is *regular* if $S = L(\mathcal{A})$ for some NTA $\mathcal{A}$. By $\mathsf{Trees}_A^{\leq n} = \{T \in \mathsf{Trees}_A : |T| \leq n\}$ we denote the set of all trees over A with at most n nodes.

(Regular) ground tree rewriting systems: A *regular ground tree rewriting system (RGTRS)* is a tuple $\mathcal{R} = (A, \mathbb{A}, R)$, where A is a ranked alphabet, $\mathbb{A} \subseteq \mathsf{Act}$ is finite set of actions, and R is finite set of rewriting rules of the form $L \overset{a}{\hookrightarrow} L'$, where L and L' are regular tree languages given as NTA. A ground tree rewriting system (GTRS) is an RGTRS such that each regular tree language is singleton (given explictly). The transition system of $\mathcal{R}$ is $\mathcal{T}(\mathcal{R}) = (\mathsf{Trees}_A, \mathbb{A}, \{\overset{a}{\longrightarrow} \mid a \in \mathbb{A}\})$, where for each $a \in \mathbb{A}$, we have $T \overset{a}{\longrightarrow} T'$ if and only if there is some $x \in D_T$ and some rule $L \overset{a}{\hookrightarrow} L' \in R$ such that $T^{\downarrow x} = S$ and $T' = T[x/S']$ for some $S \in L$ and some $S' \in L'$.

Decision problems: In this paper we will be interested in the following decision problems.

EF *model checking* asks, given a (R)GTRS $\mathcal{R} = (A, \mathbb{A}, R)$, a $T \in \mathsf{Trees}_A$ and an EF-formula φ, to decide if $(\mathcal{T}(\mathcal{R}), T) \models \varphi$ holds? The analogous question can be asked for the syntactic fragments EF_i of EF. EF model checking of RGTRS is proven to be decidable in time $\mathsf{TOWER}(O(n))$ in [14].

Strong Bisimilarity Checking against Finite Systems asks, given a (R)GTRS $\mathcal{R} = (A, \mathbb{A}, R)$, $T \in \mathsf{Trees}_A$, a finite $\mathfrak{F} = (C, \mathbb{A}, \{\overset{a}{\longrightarrow} \mid a \in \mathbb{A}\})$ and a configuration $c \in C$, to decide if $T \sim c$ holds. *Weak Bisimilarity Checking against Finite Systems* asks, given a (R)GTRS $\mathcal{R} = (A, \mathbb{A}_\tau, R)$, $T \in \mathsf{Trees}_A$, a finite $\mathfrak{F} = (C, \mathbb{A}_\tau, \{\overset{a}{\longrightarrow} \mid a \in \mathbb{A}_\tau\})$ and a configuration $c \in C$, to decide if $T \approx c$ holds.

By a result from [11] (see also Theorem 1 and Corollary 1 of [13]) weak bisimilarity checking against finite systems is polynomial time reducible to model checking EF logic, where formulas are given in DAG representation. Analogously, strong bisimilarity can be reduced in polynomial time to model checking formulas of the kind $\varphi_1 \wedge [\mathbb{A}^*]\varphi_2$, where φ_1, φ_2 are EF_0 formulas in DAG representation.

III. Model Checking

A. EF_2 *(resp.* EF_0*) is nonelementary over GTRS (resp. RGTRS)*

Our first result is that model checking EF_2 over GTRS has nonelementary complexity, which answers the open question by Löding [14].

Theorem 1. *Model checking* EF_2 *over GTRS is nonelementary.*

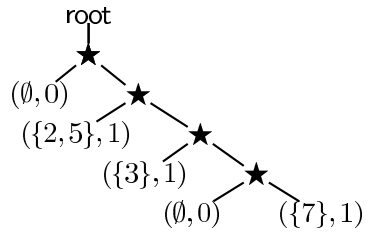
This proof of this theorem can easily be adapted to show that model checking EF_0 over RGTRS has nonelementary complexity. This lower bound proof is achieved by an exponential reduction from the decidable first-order theory over finite words, which is well-known to have a nonelementary complexity [21]. Roughly speaking, we design our GTRS in such a way that in the first phase it reaches from an input tree a huge tree whose yield (a.k.a. frontier) we interpret as a word, which will correspond to a word that witnesses *satisfiability* of an input first-order formula over finite words. This can be realized by the first occurrence of the EF operator in the input formula. In a second phase we mimic the assignment of variables of the first-order sentence by labeling leaves appropriately. In the third and final phase, we check via a deterministic bottom-up tree automaton whether the huge tree (whose leaves are now labeled with variables of the first-order sentence) satisfies the remaining unquantified subformula. This can be realized by the second occurrence of the EF operator.

Let us now proceed with the proof. Fix a first-order sentence over binary words $\psi = \exists x_1 \forall x_2 \ \ldots \ \exists x_{2n-1} \forall x_{2n} \ \varphi(x_1, \ldots, x_{2n})$. Without loss of generality we will assume $\overline{w} \not\models \psi$ for each binary word w with $|w| < 2$. Our goal is to compute in exponential time a GTRS $\mathcal{R} = (A, \mathbb{A}, R)$, some initial tree $\mathsf{start} \in \mathsf{Trees}_A$, and an EF_2-formula θ such that $\exists w \in \{0,1\}^* : \ \overline{w} \models \varphi$ iff $\mathsf{start} \models \theta$.

We define our set of actions as $\mathbb{A} = \{a_i \mid i \in [1, 2n]\} \cup \{\mathsf{down}, \mathsf{up}_0, \mathsf{up}_1, \mathsf{up}_2\}$ and let $P = \left(\left(2^{[1,2n]} \cup \{\bot\}\right) \times \{0,1\}\right)$ denote the set of *proper leaf labels*. The first component label $\bot$ will not be relevant in this but in subsequent sections. We define the ranked alphabet $A = (A_i)_{i\in\{0,1,2\}}$ of $\mathcal{R}$ as follows, where the set Q will be defined later: $A_0 = \{\mathsf{start}\} \cup P \cup Q$, $A_1 = \{\mathsf{root}\}$ and $A_2 = \{\bigstar\}$.

The regular tree language Combs consists of precisely those trees $T \in \mathsf{Trees}_A$ such that (1) $T^{-1}(\mathsf{root}) = \{\varepsilon\}$, i.e. the (one and) only node of T that is labeled with root is the root of T, (2) for each leaf x of T we have $T(x) \in P$, (3) for each inner node $x \neq \varepsilon$ of T we have that $T(x) = \bigstar$ and that x has a left child that is a leaf, and finally (4) there is at least one inner node x (with $T(x) = \bigstar$) that is the child of ε. For each $I \subseteq [1, 2n]$ define the regular tree language Combs_I to consist of precisely those combs $T \in \mathsf{Combs}$ such that (1) for each leaf x of T we have $T(x) \in 2^I \times \{0,1\}$ and (2) for each two distinct leaves x, x' of T with $T(x) = (J, \alpha)$ and $T(x') = (J', \alpha')$ we have $J \cap J' = \emptyset$ and (3) $I = \bigcup\{J \mid x \text{ is a leaf of } T \text{ and } T(x) = (J, \alpha)\}$.

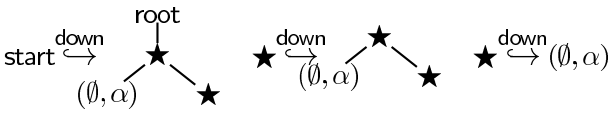
Let us give an example for a tree in $\mathsf{Combs}_{\{2,3,5,7\}}$:



Intuitively, think of the sequence of the *second components* of leaf labels of $T \in \mathsf{Combs}_I$ (i.e. the second-component projection of the labels of the yield of T) to correspond to a binary word, and moreover, for each leaf x of T, think of the *first component* of $T(x)$ to correspond to the index set of variables $\{x_1, \ldots, x_n\}$ of φ that have been bound to

the corresponding position in the word. Hence every comb in Combs_I corresponds to a unique binary word along with a variable valuation with domain I. By Combs_φ denote the trees from $\mathsf{Combs}_{[1,2n]}$ whose word and variable assignment interpretation satisfies φ.

The following three rewriting rules allow to reach all members of $\mathsf{Combs}_\emptyset$ from the singleton tree start, where $\alpha \in \{0,1\}$:



Next, we add the following rules that allows to rewrite the leafs of combs (this rewriting will correspond to assigning variables to the leaves), where $\alpha \in \{0,1\}$ and $I \subseteq [1,2n]$: $\langle I,\alpha\rangle \overset{a_i}{\hookrightarrow} \langle I \cup \{i\},\alpha\rangle$ for each $i \in [1,2n] \setminus I$.

In a next step, we compute in exponential time in $|\psi|$ a nondeterministic tree automaton $\mathcal{A} = (Q,F,A,\Delta)$ that accepts Combs_φ. We add the state set Q to A_0 of our GTRS $\mathcal{R}$. Then we add the following rewriting rules to R (which will realize the bottom-up computation of $\mathcal{A}$): (1) for each rule $(q,q',a,q'') \in \Delta \cap (Q^2 \times A_2 \times Q)$ we add the rewriting rule $a(q,q') \overset{\mathsf{up}_2}{\hookrightarrow} q''$, (2) for each rule $(a,q') \in \Delta \cap (A_0 \times Q)$ we add the rewriting rule $a \xrightarrow{\mathsf{up}_0} q'$, and (3) for each rule $(q,\mathsf{root},q') \in \Delta \cap (Q \times \{\mathsf{root}\} \times Q)$ where $q' \in F$ we add the rewriting rule $\mathsf{root}(q') \xrightarrow{\mathsf{up}_1} \mathsf{root}$.

Finally we define θ as

$$\langle \mathsf{down}^* \rangle \langle a_1 \rangle [a_2] \cdots \langle a_{2n-1} \rangle [a_{2n}] \langle \{\mathsf{up}_0, \mathsf{up}_2\}^* \rangle \langle \mathsf{up}_1 \rangle \mathtt{true}.$$

One can easily check that $\exists w \in \{0,1\}^* : \overline{w} \models \psi$ iff $(\mathcal{T}(\mathcal{R}), \mathsf{start}) \models \theta$ which concludes the proof.

B. Model checking EF_1 *over GTRS is complete for* $\mathsf{P}^{\mathsf{NEXP}}$

Our nonelementary lower bound proof above uses nested occurences of two EF operators. Our main result now is that prohibitting nested occurences of EF operators yields an elementary model checking complexity.

Theorem 2. *Over GTRS model checking formulas* $\langle \Sigma^* \rangle \varphi$ *with* $\Sigma \subseteq \mathbb{A}$ *and* $\varphi \in \mathsf{EF}_0$ *is in* NEXP.

Before sketching a proof of this theorem, we mention the following corollary, which can be easily derived by (1) establishing a polynomial space procedure using NEXP oracles (invoked whenever subformulas of the form $\langle \Sigma^* \rangle \psi$ are seen), and (2) using the fact that $\mathsf{PSPACE}^{\mathsf{NEXP}} = \mathsf{P}^{\mathsf{NEXP}}$ [1].

Corollary 3. *Model checking* EF_1 *over GTRSs is in* $\mathsf{P}^{\mathsf{NEXP}}$.

We now sketch the proof of Theorem 2. Let us now suppose that $\langle \Sigma^* \rangle \varphi$ is the given formula, $\mathcal{R} = (A, \mathbb{A}, R)$ is the given GTRS, and $T_0 \in \mathsf{Trees}_A$ is the input tree. We wish to check whether $(\mathcal{T}(R), T_0) \models \langle \Sigma^* \rangle \varphi$. Let us compute in polynomial time (cf. [14]) an NTA $\mathcal{A}$ that recognizes the set $\mathsf{post}_R^{\Sigma^*}(T_0)$ of configurations of $\mathcal{R}$ reachable from T_0 by applications of rules with labels from Σ. It now suffices to show how to compute NTAs that recognize $[\![\varphi]\!]_{\mathcal{T}(\mathcal{R})}$. We do not use the standard automata construction (e.g. [14]) for the set of trees satisfying a given EF_0 formula with respect to a given GTRS since it suffers from a nonelementary blow-up. Given an EF_0 formula, let $\mathsf{mrank}(\varphi)$ be the *modality rank* of φ, i.e., the maximum nesting depth of $\langle \cdot \rangle$ operators in φ.

We now show that $[\![\varphi]\!]_{\mathcal{T}(\mathcal{R})}$ can be expressed as a union of regular tree languages, each of which can be expressed by a tree automaton $\mathcal{A}_i$ of singly-exponential size. Furthermore, we can check whether some $L(\mathcal{A}_i)$ intersects with $L(\mathcal{A})$ in nondeterministic time exponential in $|\varphi|$ and $|\mathcal{R}|$.

Lemma 4. *We have* $[\![\varphi]\!]_{\mathcal{T}(\mathcal{R})} = \bigcup_{i \in I} L(\mathcal{A}_i)$*, for a family* $\{\mathcal{A}_i\}_{i \in I}$*, where* $|\mathcal{A}_i| = \exp(|\varphi|, |\mathcal{R}|)$*. One can nondeterministically check whether* $L(\mathcal{A})$ *intersects with some* $L(\mathcal{A}_i)$ *in time* $\exp(|\varphi|, |\mathcal{R}|)$.

As we shall see in our proof below, the parameter $|\varphi|$ in the above lemma can be replaced by $\mathsf{mrank}(\varphi)$. As a corollary, this yields the same NEXP upper bound for the model checking problem when φ is given as a DAG.

We now give a proof of Lemma 4. Let $r = \mathsf{mrank}(\varphi)$. We start by defining a standard equivalence relation on Trees_A based on the modality rank of EF_0 formulas: given two trees $T, T' \in \mathsf{Trees}_A$ and $i \in \mathbb{N}$, write $T \simeq_i T'$ if for every EF_0 formula ψ with $\mathsf{mrank}(\psi) \leq i$: $(\mathcal{T}(\mathcal{R}), T) \models \psi$ iff $(\mathcal{T}(\mathcal{R}), T') \models \psi$. In other words, $T \simeq_i T'$ iff T and T' agree on every formula ψ with modality rank at most i. It is obvious that $\simeq_i$ is an equivalence relation and that $T \simeq_{i+1} T'$ implies $T \simeq_i T'$. Furthermore, it is well-known that the equivalence relation $\simeq_i$ is of finite index, i.e., the number of equivalence classes of $\simeq_i$ is finite. For each equivalence class $\mathcal{C}$ of $\simeq_r$, it is clear that either $(\mathcal{T}(\mathcal{R}), T) \models \varphi$ for all $T \in \mathcal{C}$, or $\mathcal{T}(\mathcal{R}), T \not\models \varphi$ for all $T \in \mathcal{C}$. For the former case, we say that the equivalence class $\mathcal{C}$ is *positive*; otherwise, it is *negative*. Therefore, one idea is to define the family $\{\mathcal{A}_i\}_{i \in I}$ of NTAs by associating an NTA for each positive equivalence class $\mathcal{C}$ of $\simeq_r$. Two problems with this approach, however, are: (1) this does not give a good way of computing an NTA for each positive equivalence class, and (2) this does not reveal an upper bound on the index of $\simeq_r$.

We now define a finer relation $\equiv_i$ (for each $i \in \mathbb{N}$) that will give extra information which will help us solve these two problems. To this end, let K be the maximum number of nodes in the tree appearing in any rewrite rule in $\mathcal{R}$. Also, let $N_i = i \cdot K$. Given any two trees $T, T' \in \mathsf{Trees}_A$, we define $T \equiv_i T'$ iff for each tree $t \in \mathsf{Trees}_A$ either of the following is true: (i) the number of times t appears as a subtree of T equals the number of times t appears as a subtree of T', and (ii) the number of times t appears as a subtree exceeds N_i both for T and T'. In other words, $T \equiv_i T'$ iff each subtree with at most N_i nodes appears in T and T' the same number of times (up to some threshold). As before, it is easy to check that $\equiv_i$ is an equivalence relation and that $T \equiv_{i+1} T'$ implies $T \equiv_i T'$. To complete the proof of Lemma 4, we proceed as follows: (1) show that $\equiv_i$ is finer than $\simeq_i$, (2) checking whether a function $f : \mathsf{Trees}_A^{\leq N_i} \to [0, N_i]$ actually describes an equivalence class of $\equiv_i$ can be done rather efficiently, (3)

testing whether an equivalence class of $\equiv_i$ is positive (with respect to φ) can be done rather efficiently, and (4) for each positive equivalence class $\mathcal{C}$ of $\equiv_i$, an NTA $\mathcal{A}_\mathcal{C}$ recognizing $\mathcal{C}$ can be computed rather efficiently. As we will see, these will imply Lemma 4. For step (1) the following lemma can be shown.

Lemma 5. *For each tree $T, T' \in \mathsf{Trees}_A$, it is the case that $T \equiv_i T'$ implies $T \simeq_i T'$.*

Intuitively, this lemma holds since satisfaction of EF_0 formulas of modality rank i is only affected by the number of occurences of trees of depth N_i (up to some threshold).

Let us now proceed to step (2). Recall that each equivalence class $\mathcal{C}$ of $\equiv_r$ can be described by a function from $f_\mathcal{C} : \mathsf{Trees}_A^{\leq N_r} \to [0, N_r]$. The converse, however, is false, e.g., it is impossible to have a class $\mathcal{C}$ with $f_\mathcal{C}(T) > 1$ for a tree T with two nodes but $f_\mathcal{C}(T') = 0$ for all trees T' with one node. Also note that the special case where $f(T) = 0$ for all $T \in \mathsf{Trees}_A^{\leq N_r}$ is impossible for an equivalence class since trees have nonempty domain by definition. Therefore, we need to be able to check whether a given function $f : \mathsf{Trees}_A^{\leq N_r} \to [0, N_r]$ *actually* describes an equivalence class in $\equiv_r$. To this end, recall first that any function f that describes an equivalence class of $\equiv_r$ counts each subtree of trees T in $\mathsf{Trees}_A^{\leq N_r}$ with $f_\mathcal{C}(T) > 0$, i.e., if t is a subtree of T, then t contributes to the value of $f(t)$. We will first define a new function $g : \mathsf{Trees}_A^{\leq N_r} \to [0, N_r]$ that avoids this "double counting". This can be done by the following algorithm: "Set $g(T) := 0$ for all $T \in \mathsf{Trees}_A^{\leq N_r}$ and repeat the following for each $T \in \mathsf{Trees}_A^{\leq N_r}$ with $f(T) > 0$ (ordered by the number of nodes, starting from the largest): (1) Let $f(T) := f(T) - 1$, (2) $g(T) := g(T) + 1$, (3) Go through all nodes u of T (except when u is the root of T) and substract $f(T^{\downarrow u})$ by 1 (if becomes negative, then terminate abruptly)". Observe that if this algorithm terminates abruptly, then f does not actually describe an equivalence class of $\mathcal{C}$. Furthermore, the algorithm runs in time exponential in $r \leq |\varphi|$ and $|\mathcal{R}|$ simply because $|\mathsf{Trees}_A^{\leq N_r}| = \exp(r, |\mathcal{R}|)$ can be shown. . Now, suppose that the function g has been successfully computed from the given function f. This implies that g describes a *forest* F with each tree $T \in \mathsf{Trees}_A^{\leq N_r}$ occurring $g(T)$ many times. The original function f then describes an equivalence class iff such a forest can be further "connected into a big tree". This last check can be done using the following lemma.

Lemma 6. *The function $f : \mathsf{Trees}_A^{\leq N_r} \to [0, N_r]$ describes an equivalence class in $\equiv_r$ iff the function $g : \mathsf{Trees}_A^{\leq N_r} \to [0, N_r]$ (and the forest F corresponding to it) can be successfully computed from f by the above algorithm and that one of the following conditions are satisfied: (1) $\sum_{T \in \mathsf{Trees}_A^{\leq N_r}} g(T) = 1$. (2) $\sum_{T \in \mathsf{Trees}_A^{\leq N_r}} g(T) > 1$, and for some letter a with some rank $h \in \mathbb{N}$ and some trees $T_1, \ldots, T_h$ occuring in the forest F, the tree $a(T_1, \ldots, T_h)$ has more than N_r nodes.*

Observe that this lemma completes step (2) since this test can be performed in time exponential in r and $|\mathcal{R}|$ and polynomial in $|f|$. We now proceed to step (3). This step is rather easy since checking whether an equivalence class $\mathcal{C}$ of $\equiv_r$ described by a function $f_\mathcal{C} : \mathsf{Trees}_A^{\leq N_r} \to [0, N_r]$ is positive can be done in time exponential in r and $|\mathcal{R}|$. Intuitively, the idea is to pick a representative T of $\mathcal{C}$ of exponential size and compute a finite transition system consisting of the neighborhood of T up to depth r. It turns out that the finite system also has size exponential in $r \leq |\varphi|$ and $|\mathcal{R}|$. Therefore, we may use the standard linear-time algorithm for model checking Henessy-Milner (i.e. EF_0) formulas over finite systems.

We now proceed to step (4), which is the final step. For this, we need to show how to compute an NTA recognizing an equivalence class $\mathcal{C}$ of $\equiv_r$ described by a function $f_\mathcal{C} : \mathsf{Trees}_A^{\leq N_r} \to [0, N_r]$.

Lemma 7. *Given a function $f : \mathsf{Trees}_A^{\leq N_r} \to [0, N_r]$ that witnesses an equivalence class $\mathcal{C}$ of $\equiv_r$, we can compute an NTA recognizing precisely $\mathcal{C}$ in time $|f|^{\mathsf{poly}(r, |\mathcal{R}|)} \cdot \exp(r, |\mathcal{R}|)$.*

Roughly speaking, this lemma can be proven as follows. First, compute the function $g : \mathsf{Trees}_A^{\leq N_r} \to [0, N_r]$ using the above algorithm, which avoids double counting of subtrees. Let U denote all trees $t \in \mathsf{Trees}_A^{\leq N_r}$ such that $g(t) = N_r$. Let $t_1, \ldots, t_m$ be an enumeration of all trees $t \in \mathsf{Trees}_A^{\leq N_r}$ with $g(t) > 0$ *without* counting multiplicities. One can now design an NTA that counts that precisely $g(t_i)$ many nodes v occur such that the subtree rooted at v equals t_i, and that an arbitrary number of nodes v can occur such that the subtree rooted at v is a tree in U. It is easy to see that such an NTA of size exponential in r and $|\mathcal{R}|$ can be computed.

To summarize, the proof of Lemma 4 can now be done as follows. The NTAs $\mathcal{A}_i$ in the statement of Lemma 4 will correspond to positive equivalence classes $\mathcal{C}$ described by some functions $f_\mathcal{C} : \mathsf{Trees}_A^{\leq N_r} \to [0, N_r]$. Using the last step above, the NTA $\mathcal{A}_i$ can be computed in time exponential in r and $|\mathcal{R}|$ if f is given as an input. Checking whether $L(\mathcal{A}) \cap L(\mathcal{A}_i) \neq \emptyset$ for some i requires us to nondeterministically guess one such function f, check whether it describes an equivalence class, compute the NTA $\mathcal{A}_i$ corresponding to it, and check for language intersection with $\mathcal{A}$ in the standard way.

Let us discuss the ideas of a matching lower bound for EF_1.

Lemma 8. *Over GTRS model checking formulas $\langle \mathbb{A}^* \rangle \varphi$, where $\varphi \in \mathsf{EF}_0$, is* NEXP*-hard.*

Proof sketch. The reduction is from the $2^n \times 2^n$ tiling problem [18]. The idea is to reach via $\xrightarrow{\mathbb{A}}{}^*$ some binary tree with superleafs, where a *superleaf* is a tree of depth one whose root has arity $2n$. Each child of a superleaf will either have a nullary symbol b_0 or b_1, where the root of a superleaf contains a tile type. Each superleaf corresponds to a grid element $(i, j) \in [0, 2^n - 1] \times [0, 2^n - 1]$ where the nullary symbols of the first n (resp. last n) children encode i (resp. j) in binary. The formula φ is now a conjunction of EF_0 formulas expressing the following: **(1)** A superleaf for $(0, 0)$ exists, **(2)** whenever there are two superleafs corresponding to the same (i, j) then their tile types are the same, **(3)** if there

is a superleaf for (i,j) with $i < 2^n - 1$ (resp. $j < 2^n - 1$), then there is a superleaf for $(i+1,j)$ (resp. $(i,j+1)$), and finally **(4)** the horizontal and vertical tile conditions hold for every superleaf. □

By encoding *circuit value* into nodes of trees (gates and its evaluations will be represented in nodes in the tree) and invoking a subroutine to the trees that realized the domino problem one can prove a matching lower bound for EF_1.

Theorem 9. *Over GTRS model checking* EF_1 *is hard for* $\mathsf{P}^{\mathsf{NEXP}}$.

Remark. $\mathsf{P}^{\mathsf{NEXP}}$*-completeness for* EF_1 *model checking over PA-processes (cf. [16]) can be shown using the same techniques; see the technical report for a sketch.*

IV. Strong bisimilarity against finite systems

Since strong bisimilarity checking can be reduced to EF model checking, the following theorem is known.

Theorem 10 ([14], [11])**.** *Strong bisimilarity checking of RGTRS against finite systems is decidable in time* $\mathsf{TOWER}(O(n))$.

Over GTRS, however, we obtain an elementary upper bound. It can be derived via a reduction to model checking formulas of the kind $\varphi_1 \wedge [\mathbb{A}^*]\varphi_2$ where φ_1 and φ_2 are EF_0 DAG-formulas and then applying our upper bound result from Theorem 2. [By the above remark, this technique can also be used to prove a coNEXP upper bound for the same problem over PA-processes (cf. [20]).]

Theorem 11. *Strong bisimilarity checking of GTRS against finite systems is in* coNEXP.

As a main result of this section we prove that strong bisimilarity between a regular ground tree rewrite system and a finite system has nonelementary complexity.

Theorem 12. *Strong bisimilarity checking of RGTRS against finite systems is nonelementary.*

Although the proof also goes via an exponential reduction from the first-order theory over finite words, due to the lack of finite control unit in (R)GTRS it is not merely an adaptation of the proof of the nonelementary lower bound for EF_2 model checking over GTRS from the previous section. Roughly speaking, we implement Defender's forcing technique by providing rewriting rules of the form $L \hookrightarrow T$, where L is a regular tree language and T is an explicit ranked tree. Such rules will allow Defender to punish Attacker in case he did not play in a way that corresponds to evaluating the first-order sentence on the huge tree. However, the biggest obstacle we have to overcome is the possibility of Attacker assigning an *arbitrary permutation* of the variables in the input first-order sentence to leaves of the tree.

Let us proceed to the proof of Theorem 12. We reuse some of the notation that was introduced in Section III-A. Again, let us fix a first-order sentence interpreted over binary words $\psi = \exists x_1 \forall x_2 \ \ldots \ \exists x_{2n-1} \forall x_{2n} \varphi(x_1, \ldots, \varphi_{2n})$ and let us assume again $\overline{w} \not\models \psi$ for each binary word w with $|w| < 2$. Our goal is to compute in exponential time a RGTRS $\mathcal{R} = (A, \mathbb{A}, R)$, some initial tree $\mathsf{start} \in \mathsf{Trees}_A$, some finite transition system $\mathfrak{F} = (C, \mathbb{A}, \{\xrightarrow{a}_{\mathfrak{F}} \mid a \in \mathbb{A}\})$, and a configuration $s_\emptyset \in C$ such that there is some $w \in \{0,1\}^*$ with $\overline{w} \models \psi$ iff $\mathsf{start} \not\sim s_\emptyset$. We call a subset $I \subseteq [1,2n]$ *game-conform* if $I = [1,k]$ for some $k \in [0,2n]$ and *non-game-conform* otherwise. Analogously, we call a comb $T \in \mathsf{Combs}_I$ *game-conform* (resp. *non-game-conform*) if I is game-conform (resp. non-game-conform). Each game-conform comb $T \in \mathsf{Combs}_{[1,k]}$ naturally induces a valuation ν_T of variables with indices from $[1,k]$ to positions of the yield string defined by T. Let $\varphi[\nu_T]$ denote the formula that is obtained from φ by replacing the information given by ν_T. This can be extended to define $\psi[\nu_T]$. Hence, e.g. $\psi[\nu_T]$ is of the form $\exists x_{k+1} \cdots \forall x_{2n} \varphi[\nu_T]$ in case k is even.

In case $I \subset [1,2n]$ and $i \in [1,2n] \setminus I$ we say a tree $T' \in \mathsf{Combs}_{I \cup \{i\}}$ is an *i-extension* of T if T' can be obtained from T by choosing exactly one leaf x and replacing its label $T(x) = (J, \alpha)$ by $(J \cup \{i\}, \alpha)$. Recall that by Combs_φ we denote the trees from $\mathsf{Combs}_{[1,2n]}$ whose word and variable assignment interpretation satisfies φ. Likeweise let $\mathsf{Combs}_{\overline{\varphi}}$ denote the trees from $\mathsf{Combs}_{[1,2n]}$ whose word and variable assignment interpretation does not satisfy φ.

For each game-conform I we have *two* configurations s_I and $\overline{s_I}$ in $\mathfrak{F}$. For each non-game-conform I we have *one* corresponding configuration u_I in $\mathfrak{F}$. In addition our finite system $\mathfrak{F}$ has the configurations succ and fail. We define as action labels $\mathbb{A} = \{a_i \mid i \in [1,2n]\} \cup \{\varphi\}$.

The idea of the bisimulation game and difficulties

The high level idea of the strong bisimulation game goes as follows, and uses Defender's forcing techniques as e.g. in [13]: (**initial round**) Attacker chooses a comb T from $\mathsf{Combs}_{[1,1]}$ for which he claims that $T \models \psi[\nu_T]$ holds. Defender can only respond $s_\emptyset \xrightarrow{a_1}_{\mathfrak{F}} s_{[1,1]}$. Hence the new pebble configuration is $(T, s_{[1,1]})$.

Next, we repeat the following round game, where the current pebble configuration is $(T, s_{[1,k]})$ where $T \in \mathsf{Combs}_{[1,k]}$ for each round $k = 1, \ldots, 2n-1$: (**universal round**) If k is odd, then Attacker is supposed to move in $\mathfrak{F}$, namely $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} s_{[1,k+1]}$ although the move $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} \overline{s_{[1,k+1]}}$ is possible. Defender is now forced to move in $\mathcal{T}(\mathcal{R})$, namely $T \xrightarrow{a_{k+1}} T'$ for some $k+1$-extension T' of T. This response corresponds to the universal quantification $\forall x_{k+1}$ in ψ. (**existential round**) If k is even, then Attacker is supposed to move in $\mathcal{T}(\mathcal{R})$, namely $T \xrightarrow{a_{k+1}} T'$ for some $k+1$-extension T' of T. This move corresponds to the existential quantification $\exists x_{k+1}$ in ψ. Defender's only possible response in $\mathfrak{F}$ is $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} s_{[1,k+1]}$. (**final round**) Finally, when we are in the pebble configuration $(T, s_{[1,2n]})$, where $T \in \mathsf{Combs}_{[1,2n]}$, the action φ can be performed that allows Attacker to win (via a rule in $\mathcal{R}$ that contains Combs_φ on the left-hand side) iff $T \models \varphi[\nu_T]$.

In order to implement such a game, several difficulties arise. Let us discuss these difficulties for the universal round (the existential round can be treated dually) and give solutions to them. The question is: In the universal round, how can

we force Attacker to make in $\mathfrak{F}$ the move $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} s_{[1,k+1]}$? **1. Difficulty:** What if Attacker moves $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} \overline{s_{[1,k+1]}}$ (which will exist in $\mathfrak{F}$)? **Solution:** We add the rule $\mathsf{Combs}_{[1,k]} \overset{a_{k+1}}{\hookrightarrow} \overline{s_{[1,k+1]}}$ to $\mathcal{R}$ such that Defender has the possibility to establish syntactic equivalence by responding $T \xrightarrow{a_{k+1}} \overline{s_{[1,k+1]}}$ in $\mathcal{T}(\mathcal{R})$ and hence wins. **2. Difficulty:** What if Attacker moves $T \xrightarrow{a_{k+1}} T'$ in $\mathcal{T}(\mathcal{R})$ for some $k+1$-extension of T rather than playing in $\mathfrak{F}$? **Solution:** Defender can react in $\mathfrak{F}$, depending on whether $T' \models \psi[\nu_{T'}]$ or $T' \not\models \psi[\nu_{T'}]$. In case $T' \models \psi[\nu_{T'}]$ she can move to $\overline{s_{[1,k+1]}}$ and can win. In case $T' \not\models \psi[\nu_{T'}]$ she can move to $s_{[1,k+1]}$ and can win. **3. Difficulty:** What if Attacker plays $T \xrightarrow{a_i} T'$ where $i \in [1,k]$? I.e. Attacker plays an action that has already been played. **Solution:** We allow a simple transition to a configuration in $\mathfrak{F}$ from which Defender can surely win since surely $T' \in \mathsf{Combs}_\perp$ and hence $T' \notin \mathsf{Combs}_\varphi$. **4. Difficulty:** What if Attacker plays in $T \xrightarrow{a_i} T'$ in $\mathcal{T}(\mathcal{R})$ where $i > k+1$? I.e. Attacker deviates from playing a sequence of actions $a_1 \cdots a_k$ that correspond to assigning variables to positions of the yield string of the tree. We also say that the current pebble configuration is non-game-conform. **Solution:** We allow in $\mathfrak{F}$ a special transition for Defender $s_{[1,k]} \xrightarrow{a_i} u_{[1,k]\cup\{i\}}$ that allows her to win.

The solutions to Difficulty 1 and 2 are standard and are similar to a technique elaborated in [13]. The solution to Difficulty 3 is straightforward. The real difficulty *in the absence of a finite control* (pushdown systems have a finite control) in the game is Difficulty 4. We have to provide configurations in $\mathfrak{F}$ that allow to remember the set of variables in the current tree T that have been assigned. The difficulty that now arises is that Attacker can continue labeling leafs in T and pretend some moves later that the current tree T is game-conform all of a sudden (and hence threaten to play the above-mentioned punishing moves for instance). We have to carefully design transitions in $\mathfrak{F}$ that sooner or later punish Attacker since he was the one who deviated from playing game-conform.

The finite system: We now define the outgoing transitions of $s_{[1,k]}$ and of $\overline{s_{[1,k]}}$, *for each possible* $k \in [0, 2n-1]$: (1) $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} s_{[1,k+1]}$, (2) $\overline{s_{[1,k]}} \xrightarrow{a_{k+1}}_{\mathfrak{F}} \overline{s_{[1,k+1]}}$, (3) $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} \overline{s_{[1,k+1]}}$ if k is odd, (4) $\overline{s_{[1,k]}} \xrightarrow{a_{k+1}}_{\mathfrak{F}} s_{[1,k+1]}$ if k is even, (5) $s_{[1,k]} \xrightarrow{a_i}_{\mathfrak{F}} s_{[1,2n]}$ for each $i \in [1,k]$, (6) $\overline{s_{[1,k]}} \xrightarrow{a_i}_{\mathfrak{F}} s_{[1,2n]}$ for each $i \in [1,k]$, (7) $s_{[1,k]} \xrightarrow{a_i}_{\mathfrak{F}} u_{[1,k+1]\cup\{i\}}$ for each $i \in [k+2, 2n]$, (8) $\overline{s_{[1,k]}} \xrightarrow{a_i}_{\mathfrak{F}} u_{[1,k+1]\cup\{i\}}$ for each $i \in [k+2, 2n]$, (9) $s_{[1,2n]} \xrightarrow{\varphi}_{\mathfrak{F}} \mathsf{succ}$, (10) $\overline{s_{[1,2n]}} \xrightarrow{a_i}_{\mathfrak{F}} s_{[1,2n]}$ for each $i \in [1,2n]$, (11) $\overline{s_{[1,2n]}} \xrightarrow{\varphi}_{\mathfrak{F}} \mathsf{fail}$ and the transition (12) $\mathsf{fail} \xrightarrow{a_i}_{\mathfrak{F}} \mathsf{fail}$ for each $i \in [1,2n]$.

Let us now define the outgoing transitions of u_I for each non-game-conform $I \subseteq [1,2n]$: (1) $u_I \xrightarrow{a_i}_{\mathfrak{F}} s_{[1,2n]}$ for each $i \in I$, (2) $u_I \xrightarrow{a_i}_{\mathfrak{F}} u_{I\cup\{i\}}$ for each $i \notin I$ for which $I \cup \{i\}$ is non-game-conform, (3) $u_I \xrightarrow{a_i}_{\mathfrak{F}} s_{I\cup\{i\}}$ for each $i \notin I$ for which $I \cup \{i\}$ is game-conform, and (4) $u_I \xrightarrow{a_i}_{\mathfrak{F}} \overline{s_{I\cup\{i\}}}$ for each $i \notin I$ for which $I \cup \{i\}$ is game-conform.

A snapshot of $\mathfrak{F}$ is depicted in Figure 1.

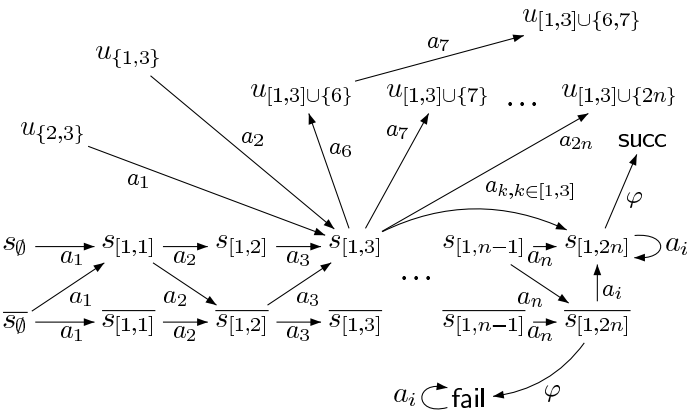


Fig. 1. A snapshot of $\mathfrak{F}$ and the outgoing transitions of $s_{[1,3]}$ in the strong bisimulation game (i ranges over $[1,2n]$).

The infinite system: Recall that C denotes the set of configurations of $\mathfrak{F}$ and that P denotes the set of proper leaf labels as defined in Section III-A. We define the ranked alphabet $A = (A_i)_{i\in\{0,1,2\}}$ of $\mathcal{R}$ as follows: $A_0 = \{\mathsf{start}\} \cup P \cup C$, $A_1 = \{\mathsf{root}\}$ and $A_2 = \{\bigstar\}$. We note that the only relevant trees (i.e. configurations) in $\mathcal{T}(\mathcal{R})$ in our reduction whose leafs are labeled with C are *singleton trees*. To R we add the rewriting rules $c \overset{b}{\hookrightarrow} c'$ for each transition $c \xrightarrow{b}_{\mathfrak{F}} c'$ in $\mathfrak{F}$. Furthermore, we add the following leaf rewriting rules, where $\alpha \in \{0,1\}$: (1) $\langle I, \alpha\rangle \overset{a_i}{\hookrightarrow} \langle I \cup \{i\}, \alpha\rangle$ for each $i \in [1,2n] \setminus I$, (2) $\langle I, \alpha\rangle \overset{a_i}{\hookrightarrow} \langle \perp, \alpha\rangle$ for each $i \in I$, and (3) $\langle \perp, \alpha\rangle \overset{a_i}{\hookrightarrow} \langle \perp, \alpha\rangle$ for each $i \in I$. We define the regular tree language $\mathsf{Combs}_\perp = \mathsf{Combs} \setminus \bigcup_{I\subseteq[1,2n]} \mathsf{Combs}_I$. In other words, $\mathsf{Combs}_\perp$ consists of those combs $T \in \mathsf{Combs}$ that satisfy (1) there is some leaf x of T with $T(x) \in \{\perp\} \times \{0,1\}$ *or* (2) there are two distinct leaves x, x' of T with $T(x) = (J, \alpha)$ and $T(x') = (J', \alpha')$ such that $J \cap J' \neq \emptyset$.

Let us add for each possible $I \subseteq [1,2n]$ and $k \in [0, 2n-1]$, the following rules to R: (1) $\mathsf{Combs}_{[1,k]} \overset{a_{k+1}}{\hookrightarrow} \overline{s_{[1,k+1]}}$ if k is odd, (2) $\mathsf{Combs}_{[1,k]} \overset{a_{k+1}}{\hookrightarrow} s_{[1,k+1]}$ if k is even, (3) $\mathsf{Combs}_{[1,k]\setminus\{i\}} \overset{a_i}{\hookrightarrow} s_{[1,k]}$ for each $i \in [1, k-2]$, (4) $\mathsf{Combs}_{[1,k]\setminus\{i\}} \overset{a_i}{\hookrightarrow} \overline{s_{[1,k]}}$ for each $i \in [1, k-2]$, (5) $\mathsf{Combs}_{I\setminus\{i\}} \overset{a_i}{\hookrightarrow} u_I$ for each $i \in I$ if I is non-game-conform, (6) $\mathsf{Combs}_I \overset{a_i}{\hookrightarrow} s_{[1,2n]}$ for each $i \in I$, (7) $\mathsf{Combs}_\varphi \overset{\varphi}{\hookrightarrow} \mathsf{fail}$, and (8) $\mathsf{Combs}_{\overline{\varphi}} \cup \mathsf{Combs}_\perp \overset{\varphi}{\hookrightarrow} \mathsf{succ}$.

One can easily verify that for each $T \in \mathsf{Combs}_{\overline{\varphi}} \cup \mathsf{Combs}_\perp$ we have $T \sim s_{[1,2n]}$. This following lemma establishes correctness of the construction.

Lemma 13. *Let $I \subseteq [1,2n]$. If I is game-conform, then (1) $s_I \not\sim \overline{s_I}$, (2) $\forall T \in \mathsf{Combs}_I$: $T \not\sim s_I$ iff $T \models \psi[\nu_T]$, and (3) $\forall T \in \mathsf{Combs}_I$: $T \sim \overline{s_I}$ iff $T \models \psi[\nu_T]$. (4) If I is non-game-conform, then for each $T \in \mathsf{Combs}_I$ we have $T \sim u_I$.*

Proof: We prove the lemma by downward induction on $|I| = 2n, 2n-1, \ldots$.

Induction base. Let $|I|$ be maximal, i.e. $I = [1,2n]$ and so I is game-conform. Thus, we only have to prove Points (1),(2), and (3). **(1)** We have to prove that $s_{[1,2n]} \not\sim \overline{s_{[1,2n]}}$. Attacker moves from $s_{[1,2n]} \xrightarrow{\varphi}_{\mathfrak{F}} \mathsf{succ}$ and hence reaches a dead-end. Defender can only respond with $\overline{s_{[1,2n]}} \xrightarrow{\varphi}_{\mathfrak{F}} \mathsf{fail}$ and does not reach a dead-end. Thus $s_{[1,2n]} \not\sim \overline{s_{[1,2n]}}$. **(2)** Let $T \in \mathsf{Combs}_{[1,2n]}$. On the one hand, assume $T \models \psi[\nu_T]$ or equivalently $T \models \varphi[\nu_T]$. Hence, $T \in \mathsf{Combs}_{\varphi}$ by definition, so Attacker can move $T \xrightarrow{\varphi} \mathsf{fail}$ in $\mathcal{T}(\mathcal{R})$ and thus reaches a configuration that is not a dead-end, whereas Defender can only respond with $s_{[1,2n]} \xrightarrow{\varphi}_{\mathfrak{F}} \mathsf{succ}$ in F which is a dead-end. Hence $T \not\sim s_{[1,2n]}$. On the other hand, assume $T \not\models \varphi[\nu_T]$. Hence $T \in \mathsf{Combs}_{\overline{\varphi}}$ and therefore Hence $T \sim s_{[1,2n]}$. Point **(3)** can be proven similar to (2).

Induction step. Let $I \subset [1,2n]$. Let us assume that I is game-conform, i.e. $I = [1,k]$ for some $k \in [2n-1]$.
(1) If k is odd, then Attacker plays $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} s_{[1,k+1]}$ and Defender can only respond with $\overline{s_{[1,k]}} \xrightarrow{a_{k+1}}_{\mathfrak{F}} \overline{s_{[1,k+1]}}$. By induction hypothesis we have $s_{[1,k+1]} \not\sim \overline{s_{[1,k+1]}}$, so $s_{[1,k]} \not\sim \overline{s_{[1,k]}}$. If k is even, then Attacker plays $\overline{s_{[1,k]}} \xrightarrow{a_{k+1}}_{\mathfrak{F}} \overline{s_{[1,k+1]}}$ and Defender can only respond with $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} s_{[1,k+1]}$. By induction hypothesis we have $s_{[1,k+1]} \not\sim \overline{s_{[1,k+1]}}$, so $s_{[1,k]} \not\sim \overline{s_{[1,k]}}$.
(2) Let $T \in \mathsf{Combs}_{[1,k]}$. We only treat the case where k is odd. Then recall that $\psi[\nu_T] = \forall x_{k+1} \exists x_{k+2} \cdots \forall x_{2n} \varphi[\nu_T]$. On the one hand, assume $T \models \psi[\nu_T]$. Hence, in other words, for each $k+1$-extension T' of T we have $T' \models \psi[\nu_{T'}]$. Attacker can now play $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} s_{[1,k+1]}$. Defender cannot play $T \xrightarrow{a_{k+1}} \overline{s_{[k+1]}}$ since $s_{[1,k+1]} \not\sim \overline{s_{[1,k+1]}}$ by Point (1) of induction hypothesis. Defender only has the possibility to respond $T \xrightarrow{a} T'$ in $\mathcal{T}(\mathcal{R})$ for some $k+1$-extension T' of T. Since $T' \models \psi[\nu_{T'}]$ we have $T' \sim s_{[1,k+1]}$ by Point (2) of induction hypothesis, hence $T \not\sim s_{[1,k]}$.
On the other hand, assume $T \not\models \psi[\nu_T]$. Thus, there is some extension $T' \in \mathsf{Combs}_{[k+1]}$ of T such that $T' \not\models \psi[\nu_{T'}]$. We have to show that $T \sim s_{[1,k]}$. • If Attacker plays $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} s_{[1,k+1]}$, then Defender responds $T \xrightarrow{a_{k+1}} T'$ and wins by Point (2) of induction hypothesis. Conversely, if Attacker plays $T \xrightarrow{a_{k+1}} T''$ for some extension $T'' \in \mathsf{Combs}_{[1,k+1]}$ of T, we distinguish two cases. In case $T'' \not\models \psi[\nu_{T''}]$, then Defender plays $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} s_{[1,k+1]}$ and wins by Point (2) of induction hypothesis. In case $T'' \models \psi[\nu_{T''}]$, then Defender responds $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} \overline{s_{[1,k+1]}}$ and wins by Point (3) of induction hypothesis. • If Attacker plays $s_{[1,k]} \xrightarrow{a_{k+1}}_{\mathfrak{F}} \overline{s_{[1,k+1]}}$, then Defender plays $T \xrightarrow{a_{k+1}} \overline{s_{[1,k+1]}}$ in $T(\mathcal{R})$ and vice versa. Defender wins by establishing syntactic equivalence. • If Attacker plays $s_{[1,k]} \xrightarrow{a_i}_{\mathfrak{F}} s_{[1,2n]}$ for some $i \in [1,k]$, then Defender responds $T \xrightarrow{a_i} T''$ for some $T'' \in \mathsf{Combs}_{\bot}$ by labeling a leaf of T and vice versa. We know that each tree from $\mathsf{Combs}_{\bot}$ is bisimilar to $s_{[1,2n]}$. • If Attacker plays $s_{[1,k]} \xrightarrow{a_i}_{\mathfrak{F}} u_{[1,k]\cup\{i\}}$ for some $i \in [k+2, 2n]$, then Defender responds $T \xrightarrow{a_i} u_{[1,k]\cup\{i\}}$ and vice versa. Hence Defender wins by establishing syntactic equivalence. • If Attacker plays $T \xrightarrow{a_i} T''$ for some i-extension $T' \in \mathsf{Combs}_{[1,k]\cup\{i\}}$ of T where $i \in [k+2, 2n]$, then Defender responds $s_{[1,k]} \xrightarrow{a_i}_{\mathfrak{F}} u_{[1,k]\cup\{i\}}$ and vice versa. By Point (4) of induction hypothesis, we have $T'' \sim u_{[1,k]}$, hence Defender wins.
We omit the proof of Point **(3)**.
Let $I \subseteq [1,k]$ be non-game-conform and let $T \in \mathsf{Combs}_I$. We have to prove that $T \sim u_I$.
• If Attacker moves $T \xrightarrow{a_i} T'$ for some $i \in I$ and some $T' \in \mathsf{Combs}_{\bot}$, then Defender responds with $u_I \xrightarrow{a_i}_{\mathfrak{F}} s_{[1,2n]}$ and vice versa, which results in a pair of bisimilar configurations. • If Attacker moves $T \xrightarrow{a_i} T'$ where $T' \in \mathsf{Combs}_{I\cup\{i\}}$ is an i-extension of T, where $i \in [1,2n] \setminus I$ and $I \cup \{i\}$ is non-game-conform, then Defender responds with $u_I \xrightarrow{a_i}_{\mathfrak{F}} u_{I\cup\{i\}}$ and vice versa. Defender wins by Point 2 of induction hypothesis. • If Attacker plays $u_I \xrightarrow{a_i}_{\mathfrak{F}} s_{[1,k]}$ (resp. $u_I \xrightarrow{a_i}_{\mathfrak{F}} \overline{s_{[1,k]}}$), i.e. in particular $I \cup \{i\}$ is game-conform, then Defender responds with $T \xrightarrow{a_i} s_{[1,k]}$ (resp. $T \xrightarrow{a_i} \overline{s_{[1,k]}}$) and vice versa. Hence Defender wins by establishing syntactic equivalence. • If Attacker plays $T \xrightarrow{a_i} T'$ for some i-extension $T' \in \mathsf{Combs}_{[1,k]}$ of T, i.e. $I \cup \{i\} = [1,k]$ is game-conform, then Defender responds $u_I \xrightarrow{a_i}_{\mathfrak{F}} s_{[1,k]}$ in case $T' \models \psi[\nu_{T'}]$ and with $u_I \xrightarrow{a_i}_{\mathfrak{F}} \overline{s_{[1,k]}}$ in case $T' \not\models \psi[\nu_{T'}]$. In the former case Defender wins since $T' \sim s_{[1,k]}$ by Point (2) of induction hypothesis. In the latter case Defender wins since $T' \sim \overline{s_{[1,k]}}$ by Point (3) of induction hypothesis. ■

In order to realize the initial round (as mentioned above) we add the rules $\mathsf{start} \xhookrightarrow{a_1} s_{[1,1]}$ and $\mathsf{start} \xhookrightarrow{a_1} \mathsf{Combs}_{[1,1]}$ to R. Theorem 12 now easily follows.

V. Weak bisimilarity against finite systems

Since weak bisimilarity checking can be reduced to EF model checking, the following theorem follows from known results.

Theorem 14 ([14], [11])**.** *Weak bisimilarity checking of RGTRS against finite systems is decidable in time* $\mathsf{TOWER}(O(n))$.

Our main result in this section is a nonelementary lower bound for weak bisimilarity checking of GTRS against finite systems.

Theorem 15. *Weak bisimilarity of GTRS against finite systems is nonelementary.*

This is technically the most involved result. Although its proof is based on the previous nonelementary lower bound for strong bisimiliarity of RGTRS against finite systems, it is *not* a straightforward adaption of the proof for strong bisimulation of Section IV. In a nutshell, we mimic one such rewriting rule $L \hookrightarrow T$ of a RGTRS, where L is a regular tree language and T is an explicit ranked tree by simulating a tree automaton for L via a sequence of τ-transitions. However, again due to the absence of a finite control unit, we are now faced with new obstacles. Firstly, we have to provide bottom-up computations for numerous tree automata and we have to provide means of making these bottom-up computations

"visible" to the finite system. Secondly, we should not allow two bottom-up computations at the same time although we have to be able to realize the first obstacle. Thirdly, it is still possible that during the bottom-up computation the scanned tree is modified (in particular for the tree language Combs_φ). The finite system has to be able to react to this.

The goal of this section is to sketch our proof that weak bisimilarity checking of GTRS against finite systems has nonelementary complexity. Due to lack of space, we mainly describe the difficulties that one has to deal with in the proof in more details. The reader is recommended to consult our technical report to get a feeling for the proof.

Recall that in Section IV we provided in the RGTRS rules of the kind $L \hookrightarrow c$, where L is a regular tree language and $c \in C$ is a configuration of $\mathfrak{F}$. These rules allowed Defender to punish Attacker in case e.g. Attacker played in $\mathfrak{F}$ rather than in $\mathcal{T}(\mathcal{R})$ or Attacker deviates from playing game-conform. However, since in GTRS the rewriting rules only allow to rewrite trees rather than trees from a tree language, we have to find a way of simulating the computations of tree automata. An ad-hoc approach we follow is to simulate such a rule $L \hookrightarrow c$ by a sequence of τ-transitions in $\mathcal{T}(\mathcal{R})$ that correspond to a *bottom-up computation* of the respective tree automaton.

Let us fix a game-conform comb T. The **major difficulties** that now arise are the following: (**1**) Due to the *absence* of a finite control unit in our GTRS $\mathcal{R}$ we must provide in the rewriting rules of $\mathcal{R}$ the possibility to simulate from T *all possible* tree automata. Hence Defender must have the possibility to react in $\mathfrak{F}$ to each of such computations. (**2**) Assume that we construct in our finite system $\mathfrak{F}$ a response for Defender to react to an initialization of a bottom-up computation of a tree language. We have to be guarantee in $\mathfrak{F}$ that each two such responses are not weakly-bisimilar in order to distinguish different (intializations of) tree languages. (**3**) Since we allow the bottom-up computation for Combs_φ, Attacker can in theory initialize from T a bottom-up computation for Combs_φ *although* $T \notin \mathsf{Combs}_{[1,2n]}$, i.e. not all of the variables have yet been assigned. In doing so, Attacker might add to T the remaining variables that have not yet been assigned and threaten to label the leafs of comb *during the bottom-up computation*. We have to provide a mechanism for Defender to win always as soon as Attacker initializes Combs_φ too early.

ACKNOWLEDGEMENTS:

We thank Thomas Colcombet, Lane Hemasspaandra, Markus Lohrey and Carsten Lutz for discussions. Anthony W. Lin is supported by EPSRC (EP/H026878/1).

REFERENCES

[1] E. Allender, M. Koucky, D. Ronneburger and S. Roy. The Pervasive Reach of Resource-Bounded Kolmogorov Complexity in Computational Complexity Theory. To appear at *JCSS'10*.

[2] A. Bouajjani, J. Esparza and O. Maler. Reachability Analysis of Pushdown Automata: Application to Model-Checking. In *CONCUR'97*, p. 135–150.

[3] A. Bouajjani, M. Müller-Olm and T. Touili. Regular Symbolic Analysis of Dynamic Networks of Pushdown Systems. In *CONCUR'05*, p. 473–487.

[4] W. S. Brainerd. Tree Generating Regular Systems. *Information and Control*, 14(2):217–231, 1969.

[5] D. Caucal. On infinite transition graphs having a decidable monadic theory. *TCS* 290(1):79–115, 2003.

[6] H. Comon *et al.* *Tree Automata Techniques and Applications*. Available at: http://www.grappa.univ-lille3.fr/tata.

[7] J. -L. Coquidé, M. Dauchet, R. Gilleron and S. Vágvölgyi. Bottom-Up Tree Pushdown Automata: Classification and Connection with Rewrite Systems. *Theor. Comput. Sci.*, 127(1):69–98, 1994.

[8] M. Dauchet, T. Heuillard, P. Lescanne and S. Tison. Decidability of the Confluence of Finite Ground Term Rewrite Systems and of Other Related Term Rewrite Systems. *Inf. Comput.*, 88(2):187–201, 1990.

[9] M. Dauchet and S. Tison. The Theory of Ground Rewrite Systems is Decidable. In *LICS'90*, p. 242–248.

[10] J. Esparza and A. Podelski. Efficient algorithms for pre* and post* on interprocedural parallel flow graphs. In *POPL'00*, p. 1–11.

[11] P. Jancar, A. Kucera and R. Mayr. Deciding bisimulation-like equivalences with finite-state processes. *Theor. Comput. Sci.*, 258(1–2):409–433, 2001.

[12] V. Kahlon and A. Gupta. On the analysis of interacting pushdown systems. In *POPL'07*, p. 303–314.

[13] A. Kucera and R. Mayr. On the Complexity of Checking Semantic Equivalences between Pushdown Processes and Finite-state Processes. Inf. Comput., 208(7):772–796, 2010.

[14] C. Löding. *Infinite Graphs Generated by Tree Rewriting*. PhD Thesis, RWTH Aachen, 2003.

[15] D. Lugiez and Ph. Schnoebelen. The regular viewpoint on PA-processes. *Theor. Comput. Sci.*, 274(1–2):89–115,2002.

[16] R. Mayr. Decidability and Complexity of Model Checking Problems for Infinite-State Systems. PhD thesis, TU-Munich, 1998.

[17] D. E. Muller and P. E. Schupp. The Theory of Ends, Pushdown Automata, and Second-Order Logic. *Theor. Comput. Sci.*, 37:51–75, 1985.

[18] C. Papadimitriou. Computational Complexity. *Addison Wesley, 1994*.

[19] S. Qadeer and J. Rehof. Context-Bounded Model Checking of Concurrent Software. In *TACAS'05*, p. 93–107.

[20] J. Srba. Roadmap of Infinite Results. *Bulletin of the EATCS*, 78:163–175, 2002. Updated version: http://www.brics.dk/~srba/roadmap/

[21] L. J. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD Thesis, MIT, 1974.

[22] S. Tison. Fair Termination is Decidable for Ground Systems. In *RTA'89*, p. 462–476.

[23] A. W. To and L. Libkin. Algorithmic Metatheorems for Decidable LTL Model Checking over Infinite Systems. In FoSSaCS'10, p. 221–236.

[24] A. W. To. *Model Checking Infinite-State Systems: Generic and Specific Approaches*. PhD Thesis, University of Edinburgh, 2010.

[25] I. Walukiewicz. Pushdown Processes: Games and Model Checking. In *CAV'96*, pages 62–74.

[26] I. Walukiewicz. Model Checking CTL Properties of Pushdown Systems. In FSTTCS'00, p.127–138.