Weakly-Synchronized Ground Tree Rewriting

(with applications to verifying multithreaded programs)

Anthony Widjaja Lin

Oxford University Department of Computer Science

Abstract. Ground tree rewrite systems (GTRS) are a well-known tree-extension of prefix-rewrite systems on words (a.k.a. pushdown systems), where subtrees (instead of word prefixes) are rewritten. GTRS can model programs with unbounded recursion depth and thread-spawning, wherein the threads have a tree-shaped dependency graph. We consider the extension of GTRS with a finite (global) control unit for synchronizing among the active threads, a.k.a. state-extended GTRS (sGTRS). Since sGTRS is Turing-complete, we restrict the finite control unit to dags possibly with self-loops, a.k.a. weakly-synchronized GTRS (wGTRS). wGTRS can be regarded as a generalization of context-bounded analysis of multipushdown systems with dynamic thread spawning. We show that reachability, repeated reachability, and the complement of model checking deterministic LTL over weakly-synchronized GTRS (wGTRS) are NP-complete by a polynomial reduction to checking existential Presburger formulas, for which highly optimized solvers are available.

1 Introduction

Pushdown systems (PDS) are a natural abstraction of sequential programs with unbounded recursions. Their verification problems have been extensively studied (e.g. see the survey [5]), many of which are not only decidable, but also relatively tractable.

Apart from having function calls, real-world programs are often multithreaded. Given the rapidly increasing popularity of multi-core computers, multithreaded applications are only becoming increasingly more popular. Most popular programming languages (e.g. Java, Python, C++, C#) now have built-in constructs to support multithreading, e.g., `Fork/Join`, `parbegin-parend`, `Parallel.For`. Such constructs allow an unbounded number of threads at any given time (due to thread-spawning). This motivates the study of verification problems on extensions of pushdown systems with multithreading.

In this paper, we start with a well-known extension of PDS called ground tree rewrite systems (GTRS), e.g., see [14]. Since PDS can be thought of as prefix-rewrite systems (prefixes are rewritten based on a given set of rewrite rules on words), GTRS can be construed as a tree-extension of PDS, wherein subtrees (instead of prefixes) are rewritten based on a given set of rewrite rules on ranked trees. Owing to the tree structure of GTRS, one can easily mimic the effect of `parbegin-parend` and `Parallel.For` language constructs, whereby a parent thread spawns several child threads and waits for the return values of computation of these child threads. This statement actually only holds *so long as there is no "shared" (global) variables*, which permit synchronization between the active (instead of waiting) threads.

A natural way to extend GTRS so as to allow synchronization via shared variables between the active threads is to extend GTRS with a finite number of (global) control states. Such an extension is called state-extended GTRS (sGTRS). Unlike GTRS, for which reachability and repeated reachability are solvable in polynomial time, sGTRS can easily simulate multistack pushdown automata for which most verification problems quickly become undecidable [19].

One way to extend GTRS with control states while staying within the realm of decidability is to disallow cycles (other than self-loops) in the transition graph of the control states of sGTRS. Such transition graphs of (with initial/accepting states) are often called 1-weak automata [16]. The class of sGTRS which satisfies this restriction is called weakly-synchronized (or weakly-extended) GTRS, which we abbreviate as wGTRS [21]. It is known [21] that reachability, repeated reachability, and the complement of model checking deterministic fragment of LTL (LTL_{det}) over wGTRS are all solvable in exponential time, but are NP-hard. LTL model checking over GTRS is known to be undecidable (e.g. see [4, 11]). For GTRS, reachability and repeated reachability are in P (e.g. [14]).

What is the modeling power of wGTRS? Useful timing and event constraints can be embedded in 1-weak automata (e.g. see [9]). In addition, we shall see later that wGTRS: (1) generalizes multipushdown systems with bounded context switches [18] by allowing dynamic thread creation using `parbegin-parend` or `Parallel.For` constructs, and (2) provides a natural underapproximation of sGTRS, where global synchronizations take place for at most a given bound n of times.

Contributions. The main contribution of this paper is a technique for showing optimal complexity of model checking weakly-synchronized GTRS by a poly-time reduction to satisfiability of existential Presburger formulas, which is NP-complete and for which there are highly-optimized solvers. We firstly consider the global reachability problem: given a wGTRS $\mathcal{P}$ over ranked alphabet Σ and two tuples $(s_0, \mathcal{S})$, $(t_0, \mathcal{T})$ of control states of $\mathcal{P}$ and tree automata over Σ, decide if there exists a path from some configuration (s_0, T_1) of $\mathcal{P}$, where $T_1 \in \mathcal{L}(\mathcal{S})$ to some configuration (t_0, T_2) of $\mathcal{P}$, where $T_2 \in \mathcal{L}(\mathcal{T})$. We show in Section 4 that this problem is NP-complete by a reduction to satisfiability of existential Presburger formulas.

We give several further applications of this upper bound in Section 5: (1) another poly-time algorithm for global reachability for GTRS and (2) NP-completeness for repeated reachability and the complement of LTL_{det} model checking for wGTRS.

In the sequel, when deriving upper bounds, we allow infinitely many rewrite rules in the input (w)GTRS compactly represented by means of tree automata.

Other related work. There are two other approaches to extend pushdown systems with dynamic thread spawning. Process rewrite systems hierarchy proposed by Mayr [17]. Some classes of systems in this hierarchy (including PA and PAD processes) are intimately connected to GTRS [11]. Another approach was considered by Bouajjani *et al.* [3] in their work on networks of pushdown systems (called CPDN).

The authors of [13] studied the extension of process rewrite systems and other classes in the hierarchy with 1-weak finite control unit. They showed that decidability can still be retained for reachability, among others. Decidability and undecidability of fragments of LTL have also been fully classified [4]. The techniques considered in this paper can be easily adapted to show that reachability, repeated reachability for weakly extended PA and PAD are NP-complete, while $\mathrm{LTL}_{\mathrm{det}}$ model checking for weakly extended PA and PAD are coNP-complete.

Context-bounded model checking over multipushdown systems was first studied in [18] and is shown to be NP-complete for multipushdown systems. Various extensions have been proposed including phase-bounds [22], ordered multi-stack machines [1], bounded languages [8, 10], dynamic thread creation [2], and more general approach [15]. The work of [2] considers a different style of multithreading than what we consider in this paper. The difference is akin to the difference between GTRS and CPDN, which are unexplored. We leave it as future work to explore the connections.

2 Preliminaries

General notations For two given natural numbers $i \leq j$, we define $[i, j] = \{i, i+1, \ldots, j\}$. Define $[k] = [0, k]$. Given a function $f : S_1 \to S_2$ and a subset $S'_1 \subseteq S_1$, the notation $f_{|_{S'_1}}$ is used to denote the restriction of f to the domain S'_1. Vectors $\mathbf{v}$ over a set S are simply elements of S^k for some positive integer k. An example is when $S = \mathbb{N}$, which gives us vectors of naturals. In the sequel, vectors are also thought of as a function $\mathbf{v} : I \to S$, where I is some nonempty finite index set. Vectors in the standard sense use $I = [1, k]$ for some positive integer k. When comparing two vectors of naturals $\mathbf{u}, \mathbf{v}$ over the same index set I, we use the component-wise ordering. We write $\mathbf{u} \leq \mathbf{v}$ iff, for each $i \in I$, $\mathbf{u}(i) \leq \mathbf{v}(i)$. This partial ordering $\leq$ is well-known to be well-founded.

Transition systems Let ACT be a finite set of *action symbols*. A *transition system* over ACT is a tuple $\mathfrak{S} = \langle S, \{\to_a\}_{a \in \mathsf{ACT}} \rangle$, where S is a set of *configurations*, and $\to_a \subseteq S \times S$ is a binary relation over S. We use $\to$ to denote the relation $\left(\bigcup_{a \in \mathsf{ACT}} \to_a\right)$. The notation $\to^+$ (resp. $\to^*$) is used to denote the transitive (resp. transitive-reflexive) closure of $\to$. Given two sets $S_1, S_2 \subseteq S$ of configurations, we write $S_1 \to^+ S_2$ if $s_1 \to^+ s_2$ for some $s_1 \in S_1$ and $s_2 \in S_2$. The notations $S_1 \to^* S_2$ and $S_1 \to S_2$ are defined likewise. We say that a sequence $s_1 \to \cdots \to s_n$ is a *path* (or *run*) in $\mathfrak{S}$ (or in $\to$). Given two paths $\pi_1 : s_1 \to^* s_2$ and $\pi_2 : s_2 \to^* s_3$ in $\to$, we may concatenate them to obtain $\pi_1 \odot \pi_2$ (by gluing together s_2). Given a subset $S' \subseteq S$, denote by $\textsc{Rec}^{\to}(S')$ to be the set of elements $s_0 \in S$ for which there exists an infinite path $s_0 \to s_1 \to \cdots$ visiting S' infinitely often, i.e., $s_j \in S'$ for infinitely many $j \in \mathbb{N}$. A transition system $\mathfrak{S} = \langle S, \{\to_a\}_{a \in \mathsf{ACT}} \rangle$ is said to be *1-weak* if each path $s_1 \to \cdots \to s_n$ in $\mathfrak{S}$ with

$s_1 = s_n$ satisfies $s_i = s_{i+1}$ for all $i \in [1, n-1]$. In other words, every cycle in $\mathfrak{S}$ is a self-loop.

Word languages and Parikh images An alphabet Σ is a finite set of symbols. We use standard notations from word language theory (e.g. [12]). Given a word $w \in \Sigma^*$ and $a \in \Sigma$, we use $|w|_a$ to denote the number of occurrences of a in w (e.g. $|abaa|_a = 3$). The *Parikh image of* w, denoted by $\mathbb{P}(w)$, is the integral vector $\mathbf{v} : \Sigma \to \mathbb{N}$ such that $\mathbf{v}(a) = |w|_a$. Given a language $\mathcal{L} \subseteq \Sigma^*$, its *Parikh image* is $\mathbb{P}(\mathcal{L}) = \{\mathbb{P}(w) : w \in \mathcal{L}\}$.

Tree automata and languages A *ranked alphabet* is a nonempty finite set of symbols Σ equipped with a rank function $\mathtt{rank} : \Sigma \to \mathbb{N}$. When the context is clear, a ranked alphabet will simply be referred to as an alphabet. Let $\mathtt{rank}(\Sigma)$ denote $\max\{\mathtt{rank}(a) : a \in \Sigma\}$. A *tree domain* D is a nonempty finite subset of $\mathbb{N}^*$ satisfying (1) *prefix closure*, i.e., if $vi \in D$ with $v \in \mathbb{N}^*$ and $i \in \mathbb{N}$, then $v \in D$, (2) *younger-sibling closure*, i.e., if $vi \in D$ with $v \in \mathbb{N}^*$ and $i \in \mathbb{N}$, then $vj \in D$ for each natural number $j < i$. Standard terminologies (e.g. nodes, parents, children, ancestors, descendants) will be used. A *tree* over a ranked alphabet Σ is a pair $T = (D, \lambda)$, where D is a tree domain and the *node-labeling* λ is a function mapping D to Σ such that, for each node $v \in D$, the number of children of v in D equals the rank $\mathtt{rank}(\lambda(v))$ of the node label of v. Write $\textsc{Tree}(\Sigma)$ for the set of all trees over Σ. In the sequel, we also use the standard term representations of trees (cf. [6]).

A (bottom-up) nondeterministic tree-automaton (NTA) over a ranked alphabet Σ is a tuple $\mathcal{A} = \langle Q, \Delta, F\rangle$, where Q is a finite nonempty set of states, Δ is a finite set of rules of the form $(q_1, \ldots, q_r) \overset{a}{\hookrightarrow} q$, where $a \in \Sigma$, $r = \mathtt{rank}(a)$, and $q, q_1, \ldots, q_r \in Q$, and $F \subseteq Q$ is a set of final states. A rule of the form $() \overset{a}{\hookrightarrow} q$ is also written as $\overset{a}{\hookrightarrow} q$. For a state $q \in Q$, the notation $\mathcal{A}^q$ is used to denote the NTA $\langle Q, \Delta, \{q\}\rangle$. A *run* of $\mathcal{A}$ on a tree $T = (D, \lambda)$ is a mapping ρ from D to Q such that, for each node $v \in D$ (with label $a = \lambda(v)$) with its all children $v_1, \ldots, v_r$, it is the case that $(\lambda(v_1), \ldots, \lambda(v_r)) \overset{a}{\hookrightarrow} \lambda(v)$ is a transition in Δ. For a subset $Q' \subseteq Q$, the run is said to be *accepting* if $\rho(\epsilon) \in F$. The NTA is said to *accept* T if it has an accepting run on T. The *language* $\mathcal{L}(\mathcal{A})$ *of* $\mathcal{A}$ is the set of trees which are accepted by $\mathcal{A}$. A language L is said to be *regular* if there exists an NTA accepting L.

A *context tree* with *(context) variables* $x_1, \ldots, x_n$ is a tree $T = (D, \lambda)$ over the alphabet $\Sigma \cup \{x_1, \ldots, x_n\}$, where $\Sigma \cap \{x_1, \ldots, x_n\} = \emptyset$ and for each $i = 1, \ldots, n$, it is the case that $\mathtt{rank}(x_i) = 0$ and there exists a unique *context node* u_i with $\lambda(u_i) = x_i$. In the sequel, we will often denote such a context tree as $T[x_1, \ldots, x_n]$ and, by convention, assume that $u_1, \ldots, u_n$ appear in an inorder tree traversal ordering. Given trees $T_1 = (D_1, \lambda_1), \ldots, T_n = (D_n, \lambda_n)$ over Σ, we use the notation $T[T_1, \ldots, T_n]$ to denote the tree (D', λ') obtained by filling each hole x_i by T_i, i.e., $D' = D \cup \bigcup_{i=1}^{n} u_i \cdot D_i$ and $\lambda'(u_i v) = \lambda_i(v)$ for each $i = 1, \ldots, n$ and $v \in D_i$. Given a tree T, if $T = C[t]$ for some context tree $C[x]$ and a tree t, then t is called a *subtree* of T.

Notation for Context-free Grammars. A *context-free grammar* (CFG) over an alphabet Σ is a tuple $\mathcal{G} = (\mathtt{Nt}, \mathtt{Tt}, \mathtt{Rules}, \mathtt{Start})$, where $\mathtt{Nt}$ is a finite set of *nonterminals*, $\mathtt{Tt} = \Sigma$ is a finite set *terminals*, $\mathtt{Rules}$ is a finite set of *production rules* of the form $X \to \alpha$ where $X \in \mathtt{Nt}$ and $\alpha \in (\mathtt{Nt} \cup \mathtt{Tt})^*$, and $\mathtt{Start} \in \mathtt{Nt}$ is the *start nonterminal.* In the sequel, for each $X \in \mathtt{Nt}$, we use the notation $\mathcal{G}^X$ to denote the CFG $(\mathtt{Nt}, \mathtt{Tt}, \mathtt{Rules}, X)$. We denote by $\mathcal{L}(\mathcal{G})$ the language of words generated by $\mathcal{G}$.

Existential Presburger formulas Existential Presburger formulas are formulas in the existential fragment of Presburger arithmetic, i.e., first-order theory over $\langle \mathbb{N}, + \rangle$. If $\varphi(\mathbf{x})$ is a formula with the vector $\mathbf{x}$ of free variables, where $\mathbf{x} : I \to \{x_1, \ldots, x_m\}$ is a vector with some index set I, and $\mathbf{v} : I \to \mathbb{N}$ is a vector over natural numbers, we write $\langle \mathbb{N}, + \rangle \models \varphi(\mathbf{v})$ if φ is a true formula in $\langle \mathbb{N}, + \rangle$ under the interpretation that maps each variable $\mathbf{x}(i)$ to $\mathbf{v}(i)$. A formula $\varphi(\mathbf{x})$ is said to be *satisfiable in* $\langle \mathbb{N}, + \rangle$ if there exists $\mathbf{v}$ such that $\langle \mathbb{N}, + \rangle \models \varphi(\mathbf{v})$. It is well-known that deciding satisfiability over $\langle \mathbb{N}, + \rangle$ is NP-complete [20], for which there are highly optimized solvers (e.g. Z3 [7]).

3 Weakly extended ground tree rewrite systems

A *state-extended ground tree rewrite systems (sGTRS)* over a finite set ACT of action symbols is a tuple $\mathcal{P} = \langle Q, \Sigma, \Delta \rangle$, where Q is a nonempty finite set of control states, Σ is a ranked alphabet, and Δ is a finite set of rules of the form $(q_1, \mathcal{A}_1) \overset{\alpha}{\hookrightarrow} (q_2, \mathcal{A}_2)$, where $q_1, q_2 \in Q$, $\alpha \in \mathsf{ACT}$, and $\mathcal{A}_1, \mathcal{A}_2$ are NTA over Σ. A configuration of $\mathcal{P}$ is a tuple (q, T), where $q \in Q$ and $T \in \textsc{Tree}(\Sigma)$. Let $\mathrm{Conf}(\mathcal{P})$ denote the set of configurations of $\mathcal{P}$. The transition system generated by $\mathcal{P}$ is $\mathfrak{S}_{\mathcal{P}} = \langle \mathrm{Conf}(\mathcal{P}), \{\to_a\}_{a \in \mathsf{ACT}} \rangle$, where $(q_1, T_1) \to_\alpha (q_2, T_2)$ iff there exist a context tree $T[x]$, a rule $(q_1, \mathcal{A}_1) \overset{\alpha}{\hookrightarrow} (q_2, \mathcal{A}_2)$ in Δ, and trees $t_1 \in \mathcal{L}(\mathcal{A}_1)$ and $t_2 \in \mathcal{L}(\mathcal{A}_2)$ such that $T_1 = T[t_1]$ and $T_2 = T[t_2]$. We define $\to_{\mathcal{P}}$ to be the union of all $\to_a$, where a ranges over ACT.

The *underlying control graph* of $\mathcal{P} = \langle Q, \Sigma, \Delta \rangle$ is the finite transition system $\mathfrak{S} = \langle Q, \{\to_a\}_{a \in \mathsf{ACT}} \rangle$, where $q_1 \to_a q_2$ iff $(q_1, \mathcal{A}_1) \overset{a}{\hookrightarrow} (q_2, \mathcal{A}_2)$ is a rule in Δ. We say that $\mathcal{P}$ is a *weakly-synchronized (or weakly-extended) ground tree rewrite systems (wGTRS)* if its underlying control graph is 1-weak. In the case of wGTRS, we often denote edge relation of this underlying control graph as $\prec$. We say that $\mathcal{P}$ is a *ground tree rewrite systems (GTRS)* if its underlying control graph is $\langle \{q\}, \{\to_a\}_{a \in \mathsf{ACT}} \rangle$, where $q \to_a q$, for each $a \in \mathsf{ACT}$. In this case, a GTRS $\mathcal{P} = \langle Q, \Sigma, \Delta \rangle$ is also written as $\langle \Sigma, \Delta \rangle$ (or simply Δ) for simplicity. If each letter in Σ is of rank ≤ 1, $\mathcal{P}$ is also called a *pushdown system (PDS).*

Remark: "Ground tree rewrite systems" are often defined in a rather restrictive form, wherein each NTA $\mathcal{A}$ in the rewrite rule is explicitly given as a tree $t \in \textsc{Tree}(\Sigma)$ representing the singleton set $\{t\}$. Our definition of GTRS coincides with what is commonly referred to as "regular GTRS". See [14].

Notation: In the sequel, given $s \in Q$ and an NTA $\mathcal{A}$ over Σ, we shall use the notation $(s, \mathcal{L}(\mathcal{A}))$ to mean $\{s\} \times \mathcal{L}(\mathcal{A})$.

We define the *(global) reachability problem for sGTRSs* as follows: given two NTAs $\mathcal{A}_1, \mathcal{A}_2$ over the ranked alphabet Σ, an sGTRS $\mathcal{P} = \langle Q, \Sigma, \Delta \rangle$, and two states $q_1, q_2 \in Q$, decide if $(q_1, \mathcal{L}(\mathcal{A}_1)) \rightarrow^*_{\mathcal{P}} (q_2, \mathcal{L}(\mathcal{A}_2))$. When we restrict the input sGTRSs to wGTRSs (resp. GTRSs), we call the resulting subproblems to be *global reachability problem for wGTRSs (resp. GTRSs).*

An intuitive example Modeling sequential programs as PDS is standard (e.g. see [17]): the stack is used to record program points (function names and values of local variables), while the finite control is used to record the return values from the previous function call. Modeling with (s)GTRS is similar except that the tree structure can model multithreading.

Consider a program with two functions called `fun1` and `fun2` (among others). The function `fun2`, which we leave unspecified, inputs and outputs a boolean value. The function `fun1` is defined in as follows:

$$\texttt{bool b[5]; Par.For(0,4,i,=>)}\{\texttt{b[i] = fun2(a[i])}\}\texttt{;return} \bigwedge_{i=0}^{4} \texttt{b[i];}$$

In this example, we assume that the boolean array `a` is given as input to `fun1` and has size 5. This program simply executes the assignment `b[i] = fun2(a[i])` in parallel for each $i \in [0, 4]$, and afterwards outputs the boolean value obtained by taking the conjunction of all the boolean variables `b[i]`.

Assuming there is no global variables, the above program can be easily modeled as a GTRS. For example, a GTRS model may contain the following rules: (1) $\langle \texttt{fun1}, \mathrm{s}\ 2, a \rangle \rightarrow \langle \texttt{fun1}, \mathrm{s}\ 3, a \rangle(\langle \texttt{fun2}, \mathrm{s}\ 1, a[0] \rangle, \ldots, \langle \texttt{fun2}, \mathrm{s}\ 1, a[4] \rangle)$, reflecting the `Par.For` step (s i means step i), and (2) for all $j_1, \ldots, j_4 \in \{0, 1\}$, $\langle \texttt{fun1}, \mathrm{s}\ 3, a \rangle(j_1, \ldots, j_4) \rightarrow \bigwedge_{i=0}^{4} j_i$, reflecting the return step of `fun1`.

With the existence of global (shared) variables, the above GTRS does not suffice because after rule of type (1) has been applied, the five subthreads can no longer communicate in this GTRS. Communication in general can be captured by sGTRS by embedding synchronization in the finite control. wGTRS actually suffices provided that the vector of values of the shared variables can change only for a bounded number of times.

Modeling power of wGTRS wGTRS can be used to underapproximate sGTRS. Intuitively, given an sGTRS $\mathcal{P}$ and a "depth" parameter $d \in \mathbb{N}$, a wGTRS $\mathcal{P}_d$ is constructed in polynomial time that underapproximates $\mathcal{P}$ up to d switches of control states. We can also show that context-bounded analysis of multipushdown systems [18] can be efficiently reduced to analyzing wGTRS. Both are shown in the full version.

4 Reachability

Theorem 1. *Global reachability for wGTRS is NP-complete. In fact, it is poly-time reducible to satisfiability of existential Presburger formulas.*

NP-hardness follows from the proof of Proposition 5.4.6 in [21] (by a reduction from hamiltonian path problem). We now show the upper bound. The idea of the reduction to satisfiability of existential Presburger formulas is as follows: first construct a CFG $\mathcal{G}$ which "overapproximates" the given wGTRS $\mathcal{P}$; the behavior of $\mathcal{G}$ is then limited by adding an extra existential Presburger constraint ψ. Since there is a linear-time algorithm [24] for computing the Parikh image of $\mathcal{L}(\mathcal{P})$ as existential Presburger formulas Ψ, the desired formula will be $\Psi \wedge \psi$.

We now provide the details of the reduction. We are given a wGTRS $\mathcal{P} = \langle Q, \Sigma, \Delta \rangle$ over the action alphabet ACT, and two tuples $(s_0, \mathcal{S})$ and $(t_0, \mathcal{T})$ of states $s_0, t_0 \in Q$ and NTA $\mathcal{S}, \mathcal{T}$ over Σ representing, respectively, a set $\{(s_0, T) : T \in \mathcal{L}(\mathcal{S})\}$ of start configurations and a set $\{(t_0, T) : T \in \mathcal{L}(\mathcal{T})\}$ of target configurations. Denote by $\mathfrak{S}_{\mathcal{P}} = \langle \mathrm{Conf}(\mathcal{P}), \{\rightarrow_a\}_{a \in \mathsf{ACT}} \rangle$ the transition system generated by $\mathcal{P}$. The task is to decide whether $(s_0, \mathcal{L}(\mathcal{S})) \rightarrow^* (t_0, \mathcal{L}(\mathcal{T}))$.

Denote the a-labeled edge relation of the underlying control graph G of $\mathcal{P}$ by $\prec_a$, and the transitive closure (resp. transitive-reflexive closure) of $\prec := \bigcup_{a \in \mathsf{ACT}} \prec_a$ by $\prec^+$ (resp. $\prec^*$). Since G is a DAG possibly with self-loops, it follows that $\prec^+$ is antisymmetric, i.e., if $s_1 \prec^+ s_2$ and $s_2 \prec^+ s_1$, then $s_1 = s_2$. *Without loss of generality, we assume that: (1)* $s_0 \prec^* t_0$ (for, otherwise, we immediately have $(s_0, \mathcal{L}(\mathcal{S})) \not\rightarrow^* (t_0, \mathcal{L}(\mathcal{T}))$), and (2) each state $s \in Q$ satisfy $s_0 \prec^* s \prec^* t_0$ (for all $T, T' \in \textsc{Tree}(\Sigma)$, i.e., each path $(s_0, T) \rightarrow^* (t_0, T')$ cannot go via configurations of the form (s, T'') with either $s_0 \not\prec^* s$ or $s \not\prec^* t_0$).

We now define the CFG $\mathcal{G} = (\mathtt{Nt}, \mathtt{Tt}, \mathtt{Rules}, \mathtt{Start})$. In the following, we use the notation $\mathcal{M}$ (possibly with a subscript) to range over the NTAs $\mathcal{S}$, $\mathcal{T}$, or NTAs appearing in Δ. The notation $q^{\mathcal{M}}$ (possibly with a subscript) will be used to denote a state in the NTA $\mathcal{M}$. The notation $q_F^{\mathcal{M}}$ will be used to denote a final state of $\mathcal{M}$. The starting nonterminal $\mathtt{Start} \in \mathtt{Nt}$ is marked. Add the rule $\mathtt{Start} \rightarrow X_{(s_0, q_F^{\mathcal{S}}),(t_0, q_F^{\mathcal{T}})}$, for each $q_F^{\mathcal{S}}$ and each $q_F^{\mathcal{T}}$. The nonterminal $X_{(s_0, q_F^{\mathcal{S}}),(t_0, q_F^{\mathcal{T}})}$ is initially unmarked. We then repeat the following two rules until all elements of $\mathtt{Nt}$ have been marked. If $X_{(s, q^{\mathcal{M}_1}),(t, q^{\mathcal{M}_2})} \in \mathtt{Nt}$ is unmarked, then mark $X_{(s, q^{\mathcal{M}_1}),(t, q^{\mathcal{M}_2})}$ and apply the following rules:

(Rule I) For all transitions $(q_1^{\mathcal{M}_1}, \ldots, q_r^{\mathcal{M}_1}) \overset{a}{\hookrightarrow} q^{\mathcal{M}_1}$ and $(q_1^{\mathcal{M}_2}, \ldots, q_r^{\mathcal{M}_2}) \overset{a}{\hookrightarrow} q^{\mathcal{M}_2}$ in $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively, add the rule

$$X_{(s, q^{\mathcal{M}_1}),(t, q^{\mathcal{M}_2})} \quad \rightarrow \quad X_{(s, q_1^{\mathcal{M}_1}),(t, q_1^{\mathcal{M}_2})} \cdots X_{(s, q_r^{\mathcal{M}_1}),(t, q_r^{\mathcal{M}_2})}$$

to $\mathtt{Rules}$ and add each $X_{(s, q_i^{\mathcal{M}_1}),(t, q_i^{\mathcal{M}_2})}$ on the r.h.s. of the rule to $\mathtt{Nt}$ unmarked (if not already a member of $\mathtt{Nt}$).

(Rule II) For each wGTRS rule $r = (s', \mathcal{A}) \overset{\alpha}{\hookrightarrow} (t', \mathcal{B})$ with $s \prec^* s' \prec t' \prec^* t$, and each $q_F^{\mathcal{A}}$ and each $q_F^{\mathcal{B}}$, add the rule

$$X_{(s, q^{\mathcal{M}_1}),(t, q^{\mathcal{M}_2})} \quad \rightarrow \quad \alpha(s', t') X_{(s, q^{\mathcal{M}_1}),(s', q_F^{\mathcal{A}})} X_{(t', q_F^{\mathcal{B}}),(t, q^{\mathcal{M}_2})}$$

to $\mathtt{Rules}$, add $X_{(s, q^{\mathcal{M}_1}),(s', q_F^{\mathcal{A}})}$ and $X_{(t', q_F^{\mathcal{B}}),(t, q^{\mathcal{M}_2})}$ to $\mathtt{Nt}$ unmarked (if not already a member of $\mathtt{Nt}$), and add (s', t') and each letter in $\alpha \in \mathsf{ACT}^*$ to $\mathtt{Tt}$.

Observe that $s \prec^+ t$ for each $X_{(s, q^{\mathcal{M}_1}),(t, q^{\mathcal{M}_2})} \in \mathtt{Nt}$ is an invariant throughout the above procedure. Termination of this procedure is immediate since (1) $\mathtt{Nt} \subseteq \bigcup_{\mathcal{M}_1, \mathcal{M}_2} \{X_{(s, q^{\mathcal{M}_1}),(t, q^{\mathcal{M}_2})} : s \prec^+$

t, and for each $i = 1, 2$, $q^{\mathcal{M}_i}$ is a state of $\mathcal{M}_i\}$ and the size of the set on the r.h.s. is at most $|Q|^2 \times$ (total number of NTA states in $\mathcal{P})^2$, and (2) the r.h.s. of each production rule is a word of length at most $\max\{\texttt{rank}(\Sigma), 4\}$.

Let $m := |\texttt{Tt}|$. We use the linear-time algorithm from [24] on the input $\mathcal{G}$ to produce an existential Presburger formula $\Psi(\mathbf{x})$, where $\mathbf{x} : \texttt{Tt} \to \{x_1, \ldots, x_m\}$, capturing the Parikh image of $\mathcal{L}(\mathcal{G})$, i.e., for each $\mathbf{v} : \texttt{Tt} \to \mathbb{N}$, we have $\mathbf{v} \in \mathbb{P}(\mathcal{L}(\mathcal{G}))$ iff $\langle \mathbb{N}, + \rangle \models \Psi(\mathbf{v})$. We also write $\mathbf{x}(s,t)$ to mean $\mathbf{x}((s,t))$, if $(s,t) \in \texttt{Tt}$.

We now define several constraints as quantifier-free Presburger formulas:

- $\textsc{Dom} := \bigwedge_{s \prec t, s \neq t} \mathbf{x}(s,t) \leq 1$. This "domain" formula asserts that advancing from control state s to its strict successor t can take place at most once.
- $\textsc{Out}_{\leq 1} := \bigwedge_{s \prec t_1, s \prec t_2, \text{distinct}(s,t_1,t_2)} (\mathbf{x}(s,t_1) = 1 \to \mathbf{x}(s,t_2) = 0)$. This formula asserts that from control state s, the system can only advance to *at most one* of its successors.
- $\textsc{NoIncomp} := \bigwedge_{s : s_0 \prec^+ s \prec^+ t_0} \left(\textsc{Inv}_s \to \bigwedge_{s' : s' \neq s, s \not\prec^+ s', s' \not\prec^+ s} \neg \textsc{Inv}_{s'} \right)$, where $\textsc{Inv}_t$ is a shorthand for the formula $\bigvee_{s \prec t} \mathbf{x}(s,t) = 1 \vee \bigvee_{t \prec s} \mathbf{x}(t,s) = 1$. Intuitively, if a control state s is involved in a path, then no control states that are incomparable to s (with respect to $\prec^+$) can be involved in this path.
- if $s_0 = t_0$, then $\textsc{Init} := \top$, and if $s_0 \neq t_0$, then $\textsc{Init} := \bigvee_{s_0 \prec t, s_0 \neq t} \mathbf{x}(s_0, t) = 1$. This formula states that the system must advance from the initial state.
- $\textsc{Progress} := \bigwedge_{t \in Q, s_0 \prec^+ t \prec^+ t_0, \text{distinct}(s_0,t,t_0)}$ $\left(\bigvee_{s \prec t, s \neq t} \mathbf{x}(s,t) = 1 \to \bigvee_{t \prec t', t \neq t'} \mathbf{x}(t,t') = 1 \right)$. This formula states that the system must advance from a control state it is in to one of its successors.

In the sequel, we let φ_1 denote the formula $\textsc{Dom} \wedge \textsc{Out}_{\leq 1} \wedge \textsc{NoIncomp}$, and φ_2 denote the formula $\textsc{Init} \wedge \textsc{Progress}$. The desired existential Presburger formula is $\varphi := \Psi \wedge \varphi_1 \wedge \varphi_2$. Correctness of the reduction follows from the following lemma.

Lemma 2 (Correctness of Reduction). *For each $w \in \mathsf{ACT}^*$, $(s_0, \mathcal{L}(\mathcal{S})) \to_v (t_0, \mathcal{L}(\mathcal{T}))$ for some $v \in \mathsf{ACT}^*$ with $\mathbb{P}(v) = \mathbb{P}(w)$ iff there exists $\mathbf{v} : \mathtt{Tt} \to \mathbb{N}$ such that $\mathbf{v}(a) = |w|_a$ for each $a \in \mathsf{ACT}$ and $\langle \mathbb{N}, + \rangle \models \varphi(\mathbf{v})$.*

It is not hard to show that the reduction takes polynomial time; more precisely, $O(|Q|^3 + (\texttt{rank}(\Sigma) \times N))$ time, where $N := (|Q|^2 \times N_{\max}^2) \times (M_{\max}^2 + |\Delta|)$, $N_{\max}$ be the maximum number of automata states in any given NTA appearing in $\mathcal{P}$ or $\mathcal{S}$ or $\mathcal{T}$, and $M_{\max}$ be the maximum number of automata transitions in any given NTA in $\mathcal{P}$ or $\mathcal{S}$ or $\mathcal{T}$. The analysis is given in the full version.

Remark: Adding a "counting constraint" on the path as an existential Presburger formula is easy. Such a counting constraint is simply an existential Presburger formula $\psi(\mathbf{x}')$, where $\mathbf{x}' = \mathbf{x}_{|\mathsf{ACT}}$. In this case, the desired formula is simply $\varphi \wedge \psi$.

Correctness: Proof of Lemma 2 The proofs for both directions of Lemma 2 are done by induction. However, in both cases, we will have to strengthen the statements; for, otherwise, the induction hypothesis will not get us off the ground. To this end, we will define a slight variant of the transition system $\mathfrak{S}_{\mathcal{P}}$ generated by $\mathcal{P}$ (which we call $\mathfrak{S}'_{\mathcal{P}}$).

Let $\mathfrak{S}'_{\mathcal{P}}$ be the transition system obtained by adding to $\mathfrak{S}_{\mathcal{P}}$ each ϵ-transition $(s,T) \to_\epsilon (t,T)$, for each $T \in \textsc{Tree}(\Sigma)$ and $s \prec^+ t$. Given a path

$$\pi = (p_0, T_0) \to_{a_1} \cdots \to_{a_n} (p_n, T_n)$$

in $\mathfrak{S}'_{\mathcal{P}}$ and $w = a_1 \cdots a_n$, we define $\chi(\pi)$ to be the vector $\mathbf{v} : \mathtt{Tt} \to \mathbb{N}$ such that (1) if $a \in \mathsf{ACT}$, then $\mathbf{v}(a) = |w|_a$, and (2) if $a = (s,t)$ with $s \prec t$, then $\mathbf{v}(a)$ is the number of indices $i \in [1,n]$ with $(p_{i-1}, p_i) = (s,t)$ and $a_i \neq \epsilon$.

Lemma 3. *For all $X_{(s,q^{\mathcal{M}_1}),(t,q^{\mathcal{M}_2})} \in \mathtt{Nt}$, if $w \in \mathcal{L}(\mathcal{G}^{X_{(s,q^{\mathcal{M}_1}),(t,q^{\mathcal{M}_2})}})$ with $\langle \mathbb{N}, + \rangle \models \varphi_1(\mathbb{P}(w))$, then there exists a path $\pi : (s, \mathcal{L}(\mathcal{M}_1^{q^{\mathcal{M}_1}})) \to_v (t, \mathcal{L}(\mathcal{M}_2^{q^{\mathcal{M}_2}}))$ in $\mathfrak{S}'_{\mathcal{P}}$ with $\chi(\pi) = \mathbb{P}(w)$.*

It is not hard to see that Lemma 3 implies the direction ($\Leftarrow$) of Lemma 2. A proof can be found in the full version.

Proof (of Lemma 3). As we previously saw, we have $s \prec^+ t$ for each $X_{(s,q^{\mathcal{M}_1}),(t,q^{\mathcal{M}_2})} \in \mathtt{Nt}$. The proof is by induction on the length of derivations of the word w from $X_{(s,q^{\mathcal{M}_1}),(t,q^{\mathcal{M}_2})}$.

The base case is when $X_{(s,q^{\mathcal{M}_1}),(t,q^{\mathcal{M}_2})} \to \epsilon$, which is a production rule generated by Rule I. This means that there exists $a \in \Sigma$ such that $\overset{a}{\hookrightarrow} q^{\mathcal{M}_1}$ and $\overset{a}{\hookrightarrow} q^{\mathcal{M}_2}$ are transitions of $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively, and thus $a \in \mathcal{L}(\mathcal{M}_1^{q^{\mathcal{M}_1}}) \cap \mathcal{L}(\mathcal{M}_2^{q^{\mathcal{M}_2}})$. Since $s \prec^+ t$, it follows that $(s,a) \to_\epsilon (t,a)$ is path π in $\mathfrak{S}'_{\mathcal{P}}$. It is also easy to see that $\mathbb{P}(\epsilon) = \chi(\pi) = \mathbf{0}$, and that $\langle \mathbb{N}, + \rangle \models \varphi_1(\mathbf{0})$.

We now proceed to the induction cases. There are two cases. We only consider the first case; the second case is considered in the full version.

The first induction case is when the first production rule applied in the derivation of w from $X_{(p_1,q^{\mathcal{M}_1}),(p_2,q^{\mathcal{M}_2})}$ (with $p_1 = s$ and $p_2 = t$) is

$$X_{(p_1,q^{\mathcal{M}_1}),(p_2,q^{\mathcal{M}_2})} \quad \to \quad \alpha(p_3,p_4) X_{(p_1,q^{\mathcal{M}_1}),(p_3,q^{\mathcal{M}_3})} X_{(p_4,q^{\mathcal{M}_4}),(p_2,q^{\mathcal{M}_2})}$$

where $q^{\mathcal{M}_3}$ and $q^{\mathcal{M}_4}$ are final states of $\mathcal{M}_3$ and $\mathcal{M}_4$, respectively. This production rule is generated by Rule II, which means that there exists a rule $r = (p_3, \mathcal{M}_3) \overset{\alpha}{\hookrightarrow} (p_4, \mathcal{M}_4)$ with $p_1 \prec^* p_3 \prec p_4 \prec^* p_2$ in $\mathcal{P}$. We may also write w as $\alpha(p_3,p_4) w_1 w_2$, where $w_1 \in \mathcal{L}(\mathcal{G}^{X_{(p_1,q^{\mathcal{M}_1}),(p_3,q^{\mathcal{M}_3})}})$ and $w_2 \in \mathcal{L}(\mathcal{G}^{X_{(p_4,q^{\mathcal{M}_4}),(p_2,q^{\mathcal{M}_2})}})$. Furthermore, since $\langle \mathbb{N}, + \rangle \models \varphi_1(\mathbb{P}(w))$ and $\mathbb{P}(w_1), \mathbb{P}(w_2) \leq \mathbb{P}(w)$, it follows that $\langle \mathbb{N}, + \rangle \models \varphi_1(\mathbb{P}(w_i))$ for each $i = 1,2$. By induction, there exist paths $\pi_1 : (p_1, T_1) \to^* (p_3, T_3)$ and $\pi_2 : (p_4, T_4) \to^* (p_2, T_2)$ in $\mathfrak{S}'_{\mathcal{P}}$ such that $T_i \in \mathcal{L}(\mathcal{M}_i^{q^{\mathcal{M}_i}})$, for each $i \in [1,4]$, and $\chi(\pi_j) = \mathbb{P}(w_j)$, for each $j = 1,2$. By applying the rule r above, we also see that $\pi_3 : (p_3, T_3) \to_\alpha (p_4, T_4)$ is a transition in $\mathfrak{S}_{\mathcal{P}}$. Therefore, $\pi := \pi_1 \odot \pi_3 \odot \pi_2$ is a path from (p_1, T_1) to (p_2, T_2) in $\mathfrak{S}'_{\mathcal{P}}$. We have $\chi(\pi) = \sum_{i=1}^{3} \chi(\pi_i) = \mathbb{P}(w_1) + \mathbb{P}(w_2) + \mathbb{P}(\alpha(p_3,p_4)) = \mathbb{P}(w)$. This completes the proof for the second induction case. ☐

It remains to prove the direction ($\Rightarrow$) of Lemma 2. To this end, we need the following lemma.

Lemma 4. *For all* $X_{(s,q^{\mathcal{M}_1}),(t,q^{\mathcal{M}_2})} \in \mathtt{Nt}$, *if there exists a path* $\pi : (s, \mathcal{L}(\mathcal{M}_1^{q^{\mathcal{M}_1}})) \rightarrow_v (t, \mathcal{L}(\mathcal{M}_2^{q^{\mathcal{M}_2}}))$ *in* $\mathfrak{S}'_{\mathcal{P}}$, *then there exists* $w \in \mathtt{Tt}^*$ *with* $\mathbb{P}(w) = \chi(\pi)$ *such that* $w \in \mathcal{L}(\mathcal{G}^{X_{(s,q^{\mathcal{M}_1}),(t,q^{\mathcal{M}_2})}})$.

To show the direction ($\Rightarrow$) of Lemma 2, first observe that each path $\pi : (s_0, \mathcal{L}(\mathcal{S})) \rightarrow_v (t_0, \mathcal{L}(\mathcal{T}))$ in $\mathfrak{S}_{\mathcal{P}}$ is also a path in $\mathfrak{S}'_{\mathcal{P}}$. By Lemma 4, there exists a word $w \in \mathcal{L}(\mathcal{G})$ with $\chi(\pi) = \mathbb{P}(w)$. Let $\mathbf{v} := \mathbb{P}(w)$. It follows that $\langle \mathbb{N}, + \rangle \models \Psi(\mathbf{v})$. It suffices to show that $\langle \mathbb{N}, + \rangle \models \varphi_1(\mathbf{v}) \wedge \varphi_2(\mathbf{v})$. If $s_0 = t_0$, it is easy to see that $\langle \mathbb{N}, + \rangle \models \varphi_1(\mathbf{v}) \wedge \varphi_2(\mathbf{v})$. Therefore, assume that $s_0 \neq t_0$. In this case, we take the projection of π on to its first component (i.e. control states), say, $p_0, \ldots, p_k$ such that $p_0 = s$ and $p_k = t$. By removing duplicates, we may assume that $p_0, \ldots, p_k$ are pairwise distinct control states. This means that for all distinct $p, p' \in Q$, we have $\mathbf{v}(p, p') = 1$ iff $p = p_{i-1}$ and p_i for some $i \in [1, k]$. Since $p_0, \ldots, p_k$ is a path in the underlying graph of $\mathcal{P}$, it is easy to check now that $\langle \mathbb{N}, + \rangle \models \varphi_1(\mathbf{v}) \wedge \varphi_2(\mathbf{v})$ by exhausting all the five subconjuncts in the formula $\varphi_1 \wedge \varphi_2$.

We now prove Lemma 4. To this end, we need the following technical lemma about path decompositions for wGTRS.

Lemma 5. *For each path* $\pi : (p, T_1) \rightarrow_v (q, T_2)$ *in* $\mathfrak{S}'_{\mathcal{P}}$, *there exists a context tree* $C[x_1, \ldots, x_n]$ *for some* $n \in \mathbb{N}$ *such that:*

1. $T_1 = C[t_1, \ldots, t_n]$ *for some trees* $t_1, \ldots, t_n \in \textsc{Tree}(\Sigma)$,
2. $T_2 = C[t'_1, \ldots, t'_n]$ *for some trees* $t'_1, \ldots, t'_n \in \textsc{Tree}(\Sigma)$,
3. *for each* $i \in [1, n]$, *there exists a path* $\pi_i : (p, t_i) \rightarrow^* (p_i^1, t_i^1) \rightarrow_{\alpha_i} (p_i^2, t_i^2) \rightarrow^* (q, t'_i)$ *in* $\mathfrak{S}'_{\mathcal{P}}$ *for some rewrite rule* $(p_i^1, \mathcal{A}_i) \overset{\alpha_i}{\hookrightarrow} (p_i^2, \mathcal{B}_i)$ *in* $\mathcal{P}$ *such that* $t_i^1 \in \mathcal{L}(\mathcal{A}_i)$ *and* $t_i^2 \in \mathcal{L}(\mathcal{B}_i)$.
4. $\chi(\pi) = \sum_{i=1}^{n} \chi(\pi_i)$.

It is not hard to see that Lemma 5 implies Lemma 4. The proof is done by induction on $\chi(\pi)$ (with componentwise ordering $\leq$). The above path decomposition lemma is used to prove the inductive case. The proof is not hard but tedious, and so is relegated into the full version. For space reasons, we also relegate the proof of Lemma 5 into the full version.

5 Applications

A polynomial-time algorithm for GTRS reachability The reduction from the previous section can be easily modified to give another polynomial-time algorithm for GTRS global reachability. Given the GTRS $\mathcal{P}$ and two tuples $(s_0, \mathcal{S})$ and $(t_0, \mathcal{T})$ of control states and NTAs, consider the CFG $\mathcal{G}$ and formula $\varphi = \Psi \wedge \varphi_1 \wedge \varphi_2$ produced by the reduction. Observe that, in the case of GTRS, we have $\varphi_1 \wedge \varphi_2 \equiv \top$. Therefore, Lemma 2 implies that $(s_0, \mathcal{L}(\mathcal{S})) \rightarrow^*_{\mathcal{P}} (t_0, \mathcal{L}(\mathcal{T}))$ iff Ψ is a satisfiable Presburger formula iff $\mathcal{L}(\mathcal{G}) \neq \emptyset$. Therefore, we have reduced global GTRS reachability to language emptiness of CFG, which is solvable in polynomial time. This gives us another proof of the following proposition.

Proposition 6. *GTRS global reachability is solvable in polynomial-time.*

Repeated reachability *Repeated reachability for sGTRS* is the following problem: given an sGTRS $\mathcal{P} = \langle Q, \Sigma, \Delta \rangle$, an initial set $(s_0, \mathcal{L}(\mathcal{S}))$ of configurations of $\mathcal{P}$ given as $(s_0, \mathcal{S})$, a final set $\mathcal{C}$ of target configurations of $\mathcal{P}$ given as a function mapping a control state $p \in Q$ to an NTA $\mathcal{A}_p$ over Σ (here, $\mathcal{C}$ consists of configurations of the form (p, T) for some $p \in Q$ and $T \in \mathcal{L}(\mathcal{A}_p)$), decide whether $(s_0, \mathcal{L}(\mathcal{S})) \cap \textsc{Rec}^{\to_{\mathcal{P}}}(\mathcal{C}) \neq \emptyset$. As for reachability problem, this problem is undecidable.

Theorem 7. *Repeated reachability for wGTRS is NP-complete. In fact, it is poly-time reducible to satisfiablity of existential Presburger formulas.*

NP-hardness follows from the proof of Proposition 5.4.6 in [21] (by a reduction from hamiltonian path problem). To obtain the upper bound for this theorem, we reduce this problem to satisfiability of existential Presburger formulas in polynomial time. To this end, for each $p \in Q$, we define $\mathcal{P}_p$ to be GTRS obtained by restricting $\mathcal{P}$ to control state p, i.e., $\mathcal{P}_p = \langle \{p\}, \Sigma, \Delta_p \rangle$, where Δ_p consists of all rules in Δ of the form $(p, \mathcal{A}) \to_a (p, \mathcal{B})$. We first use Löding's result [14] that an NTA $\mathcal{A}'_p$ representing $\textsc{Rec}^{\to_{\mathcal{P}_p}}((p, \mathcal{L}(\mathcal{A}_p)))$ is computable in time polynomial in $\|\mathcal{A}_p\| + \|\mathcal{P}_p\|$, for each $p \in Q$. It follows that $(s_0, \mathcal{L}(\mathcal{S})) \cap \textsc{Rec}^{\to_{\mathcal{P}}}(\mathcal{C}) \neq \emptyset$ iff, for some $p \in Q$, $(s_0, \mathcal{L}(\mathcal{S})) \to^*_{\mathcal{P}} (p, \mathcal{L}(\mathcal{A}'_p))$. Applying our poly-time reduction from the previous section for the problem instance $(s_0, \mathcal{L}(\mathcal{S})) \to^*_{\mathcal{P}} (p, \mathcal{L}(\mathcal{A}'_p))$, for each p, and existentially quantifying all free variables, we obtained existential Presburger sentences φ_p such that $(s_0, \mathcal{L}(\mathcal{S})) \to^*_{\mathcal{P}} (p, \mathcal{L}(\mathcal{A}'_p))$ iff $\langle \mathbb{N}, + \rangle \models \varphi_p$. It immediately follows that $(s_0, \mathcal{L}(\mathcal{S})) \cap \textsc{Rec}^{\to_{\mathcal{P}}}(\mathcal{C}) \neq \emptyset$ iff $\langle \mathbb{N}, + \rangle \models \bigvee_{q \in Q} \varphi_p$. This shows that the desired existential Presburger sentence is $\bigvee_{q \in Q} \varphi_p$.

Model checking deterministic LTL over wGTRS Deterministic LTL (LTL_{det}) over ACT is the fragment of LTL with the following syntax:

$$\varphi, \varphi' := p \mid \mathbf{X}\varphi \mid \varphi \wedge \varphi' \mid (p \wedge \varphi) \vee (\neg p \wedge \varphi') \mid (p \wedge \varphi)\mathbf{Op}(\neg p \wedge \varphi')$$

where p ranges over boolean combinations of ACT, and $\mathbf{Op}$ ranges over $\{\mathbf{U}, \mathbf{W}\}$. The semantics $[\![\varphi]\!] \subseteq \mathsf{ACT}^\omega$ of an LTL formula φ can be defined in the same way as for LTL, which is standard (e.g. see [23]). Given a transition system $\mathfrak{S} = \langle S, \{\to_a\}_{a \in \mathsf{ACT}} \rangle$ over ACT, we write $[\![\varphi]\!]_{\mathfrak{S}}$ to denote the set of configurations $s_0 \in S$ from which all infinite paths $\pi : s_0 \to_{a_1} s_1 \to_{a_2} \cdots$ in $\mathfrak{S}$ satisfy $a_1 a_2 \ldots \in [\![\varphi]\!]$. The problem of *model checking deterministic LTL for sGTRS* is defined as follows: given an LTL_{det} formula ψ and a sGTRS $\mathcal{P} = \langle Q, \Sigma, \Delta \rangle$ over the same set ACT of action symbols, and an initial set of configurations $(s_0, \mathcal{L}(\mathcal{S}))$ of $\mathcal{P}$ represented as the tuple $(s_0, \mathcal{S})$ of control state $s_0 \in Q$ and NTA $\mathcal{S}$ over Σ, decide if $(s_0, \mathcal{L}(\mathcal{S})) \subseteq [\![\psi]\!]_{\mathfrak{S}}$.

Theorem 8. LTL_{det} *model checking over wGTRS is coNP-complete. In fact, it is poly-time reducible to non-satisfiablity of existential Presburger formulas.*

coNP-hardness for the problem is known [21]. For space reasons, we relegate the proof for the upper bound into the full version.

Acknowledgment. I thank Matthew Hague and anonymous reviewers for their helpful feedback. I am grateful to EPSRC (EP/H026878/1) for their support.

References

1. M. F. Atig, B. Bollig, and P. Habermehl. Emptiness of multi-pushdown automata is 2ETIME-complete. In *DLT*, pages 121–133, 2008.
2. M. F. Atig, A. Bouajjani, and S. Qadeer. Context-bounded analysis for concurrent programs with dynamic creation of threads. *LMCS*, 7(4), 2011.
3. A. Bouajjani, M. Müller-Olm, and T. Touili. Regular symbolic analysis of dynamic networks of pushdown systems. In *CONCUR*, pages 473–487, 2005.
4. L. Bozzelli, M. Kretínský, V. Rehák, and J. Strejcek. On decidability of LTL model checking for process rewrite systems. *Acta Inf.*, 46(1):1–28, 2009.
5. O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. In *Handbook of process algebra*, pages 545–623. Elsevier, North-Holland, 2001.
6. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications, 2007.
7. L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, pages 337–340, 2008.
8. J. Esparza and P. Ganty. Complexity of pattern-based verification for multi-threaded programs. In *POPL*, pages 499–510, 2011.
9. K. Fisler. Toward diagrammability and efficiency in event-sequence languages. *Int. J. Softw. Tools Technol. Transf.*, 8(4):431–447, Aug. 2006.
10. P. Ganty, R. Majumdar, and B. Monmege. Bounded underapproximations. *Formal Methods in System Design*, 40(2):206–231, 2012.
11. S. Göller and A. W. Lin. Refining the process rewrite systems hierarchy via ground tree rewrite systems. In *CONCUR*, pages 543–558, 2011.
12. D. C. Kozen. *Automata and Computability.* Springer, 2007.
13. M. Kretínský, V. Rehák, and J. Strejcek. Extended process rewrite systems: Expressiveness and reachability. In *CONCUR*, pages 355–370, 2004.
14. C. Löding. *Infinite Graphs Generated by Tree Rewriting.* PhD thesis, RWTH Aachen, 2003.
15. P. Madhusudan and G. Parlato. The tree width of auxiliary storage. In *POPL*, pages 283–294, 2011.
16. M. Maidl. The common fragment of CTL and LTL. In *FOCS*, pages 643–652, 2000.
17. R. Mayr. *Decidability and Complexity of Model Checking Problems for Infinite-State Systems.* PhD thesis, TU-Munich, 1998.
18. S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *TACAS*, pages 93–107, 2005.
19. G. Ramalingam. Context-sensitive synchronization-sensitive analysis is undecidable. *ACM Trans. Program. Lang. Syst.*, 22(2):416–430, Mar. 2000.
20. B. Scarpellini. Complexity of subcases of presburger arithmetic. *Trans. of AMS*, 284(1):203–218, 1984.
21. A. W. To. *Model Checking Infinite-State Systems: Generic and Specific Approaches.* PhD thesis, LFCS, School of Informatics, University of Edinburgh, 2010.
22. S. L. Torre, P. Madhusudan, and G. Parlato. A robust class of context-sensitive languages. In *LICS*, pages 161–170, 2007.
23. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS*, pages 332–344, 1986.
24. K. N. Verma, H. Seidl, and T. Schwentick. On the complexity of equational horn clauses. In *CADE*, pages 337–352, 2005.