Micro-Macro Stack Systems: A New Frontier of Elementary Decidability for Sequential Systems

Nir Piterman*
Department of Computer Science,
The Weizmann Institute of Science,
Email: nirp@wisdom.weizmann.ac.il

Moshe Y. Vardi†
Department of Computer Science,
Rice University,
Email: vardi@cs.rice.edu

Abstract

We define the class of micro-macro stack graphs, a new class of graphs modeling infinite-state sequential systems with a decidable model-checking problem. Micro-macro stack graphs are the configuration graphs of stack automata whose states are partitioned into micro and macro states. Nodes of the graph are configurations of the stack automaton where the state is a macro state. Edges of the graph correspond to the sequence of micro steps that the automaton makes between macro states. We prove that this class strictly contains the class of prefix-recognizable graphs. We give a direct automata-theoretic algorithm for model checking μ-calculus formulas over micro-macro stack graphs.

1 Introduction

One of the most significant developments in the area of formal design verification is the discovery of algorithmic methods for verifying on-going behaviors of reactive systems [18, 40, 37, 19, 45]. In *model-checking*, we verify the correctness of a system with respect to a desired behavior by checking whether a mathematical model of the system satisfies a formal specification of this behavior (for a survey, see [20]). Traditionally, model checking is applied to *finite-state* systems, typically modeled by labeled state-transition graphs, and to behaviors that are formally specified as *temporal-logic* formulas or *automata on infinite objects*. Symbolic methods that enable model-checking of very large state spaces, and the great ease of use of fully algorithmic methods, led to industrial acceptance of model-checking [2, 21].

In recent years, researchers extended the applicability of model-checking to infinite-state systems. An active field of research is model-checking of infinite-state *sequential systems*. These are systems in which each state carries a finite, but unbounded, amount of information, e.g., a pushdown store. This research origins in The origin of this research is the result of Müller and Schupp that the monadic second-order theory of *context-free* graphs is decidable [39]. As the complexity involved in that decidability result is nonelementary, researchers sought decidability results of elementary complexity. At the same time, researchers sought decidability results for larger classes of systems. Algorithms for simpler logics and more general systems have been proposed. The most powerful results are an exponential-time algorithm by Burkart for model checking the μ-calculus with respect to *prefix-recognizable* graphs [8] and decidability of monadic second-order theory of *high order pushdown graphs* [31]. See also [12, 15, 47, 10, 11, 4, 7, 25, 13, 9, 34]

The class of high order pushdown graphs strictly contains the class of prefix-recognizable graphs [31], and the class of prefix-recognizable graphs strictly contains the class of *pushdown* graphs [15], which in turn strictly contains the class of context-free graphs [17]. These classes are defined in terms of certain rewrite rules. More powerful notion of rewrite rules yield even larger classes of graphs. The class of *synchronized* rational graphs strictly contains the class of high order pushdown graphs and in turn is strictly contained in the class of *rational* graphs. Only the first-order theory of synchronized rational graphs is, however, decidable (cf. [6, 43]). It is undecidable even to determine if some vertex is reachable from another vertex (cf. [43]). For rational graphs even first-order theory is undecidable [38]. To the best of our knowledge the class of prefix-recognizable graphs is the largest class of graphs modeling sequential systems for which there is an elementary model checking algorithm of μ-calculus.

In this paper we present the class of *micro-macro stack graphs*, which strictly contains the class of prefix-recognizable graphs and for which model checking μ-

*Supported in part by the European Commission (FET project ADVANCE, contract No IST-1999-29082). This work was carried out at the John von Neumann Minerva Center for Verification of Reactive Systems.

†Supported in part by NSF grants CCR-9988322, IIS-9908435, IIS-9978135, and EIA-0086264, by BSF grant 9800096, and by a grant from the Intel Corporation.

calculus formulas is decidable in elementary time. Every graph in our class has a simple finite representation in terms of natural rewrite rules. The extension from prefix-recognizable graphs to micro-macro stack graphs is analogous to the extension from pushdown graphs to prefix-recognizable graphs, as we now explain.

The nodes of a pushdown graph are words over some finite alphabet. Such a word represents the store content and the internal state of a pushdown automaton. A transition corresponds to the move of a pushdown automaton when reading some input letter (i.e. popping the top of the store and pushing a finite sequence). The nodes of prefix-recognizable graphs are again words representing the internal state with the content of the pushdown store. Every transition is a triplet of regular languages. Application of the rewrite rule $\langle\alpha,\beta,\gamma\rangle$ on a node x consists of finding a partition zz' of x such that z is in the regular language α and z' is in the regular language β, and then replacing the prefix z by some prefix y in the regular language γ, reaching node yz'. Prefix-recognizable graphs correspond to the configuration graphs of pushdown automata when the ϵ-transitions are factored out [41, 3]. Indeed, a pushdown automaton can do a series of ϵ-transitions that remove z from the pushdown store while checking that z is in the language α. Making sure that the suffix z' is in the language β can be done by adding information to the store. Finally, another sequence of ϵ-transitions adds y to the store. We can think of the ϵ-transitions as *micro* steps that are not exhibited in the prefix-recognizable graph. Then, advancing transitions of the pushdown automaton are *macro* steps that are exhibited in the prefix-recognizable graph.

There is another way to let the pushdown automaton check that the suffix z' is in the language β. This is by allowing the automaton to read the entire contents of the store. This is the type of behavior of *stack* automata [27, 26, 29]. Just like pushdown automata, stack automata have a finite but unbounded store, they can change only the top of the store by either removing the letter on top of the store or by adding a finite sequence of letters on top of the store. Unlike pushdown automata, stack automata can read the entire contents of their store. A stack automaton can navigate on its store, checking its entire contents. It can change the contents of the store only when it visits the top of the store. Thus a prefix-recognizable graph can be viewed also as the pruned configuration graph of a stack automaton. The following sequences of micro steps are removed: (a) removing the prefix of the pushdown store while checking that it is in the regular language α, (b) going to the bottom of the store and checking that the remaining suffix is in the language β, and (c) adding a sequence of letters from the regular language γ to the store.

In our framework we offer a more flexible partition into micro and macro states. We let the automaton itself designate which states should be left unnoticed and which states correspond to a change in the graph. We refer to such stack automata whose state set is partitioned into micro and macro states as micro-macro stack automata. The nodes of a micro-macro stack graph correspond to configurations of a micro-macro stack automaton whose state is a macro state. Edges of the graph correspond to the change performed via a sequence of micro states.

We show that micro-macro stack graphs strictly contains the class of prefix-recognizable graphs. We give two examples of micro-macro stack graphs that are not prefix-recognizable. First, as mentioned, prefix-recognizable graphs are the configuration graphs of pushdown automata. Thus, the prefix-recognizable graphs, when considered as acceptors of languages, recognize the context-free languages [41]. We give a micro-macro stack system whose graph accepts a language that is not context free. Second, under some conditions on prefix-recognizable graphs, Blumensath gives information on the size of the encoding of the nodes of a prefix-recognizable graph [3]. We give a micro-macro stack system whose graph does not meet Blumensath' characterization. The class of micro-macro stack graphs is (strictly) contained in the class of synchronized rational graphs. Indeed, we show that model checking the μ-calculus over micro-macro stack graphs is decidable in elementary time. Thus, micro-macro stack graphs constitute a new frontier of elementary decidability for sequential systems. of decidability for sequential systems.

Our model checking algorithms are automata based. The automata-theoretic approach to verification uses the theory of automata as a unifying paradigm for program specification, verification, and synthesis [46, 36, 35]. Automata enables the separation of the logical and the algorithmic aspects of reasoning about systems, yielding clean and in many cases asymptotically optimal algorithms. The automata-theoretic framework for reasoning about finite-state systems has proven to be very versatile. Recently, the automata-theoretic approach to verification has been extended to infinite-state sequential systems [34, 32]. Our model-checking algorithms for micro-macro stack graphs extend the algorithms in [34, 32]. In general, the automata-theoretic approach to branching-time model checking uses a reduction to the emptiness problem of alternating tree automata. In the following we show that for micro-macro stack graphs, the model checking of branching-time specifications can be reduced to the emptiness problem of alternating stack tree automata.

The class of micro-macro stack graphs is contained in the class of high order pushdown graphs. In order to check the contents of the stack a high order pushdown puts a fresh copy of the first order pushdown on the second order pushdown and removes the top of the first order pushdown. The information is not lost, it is stored in the old copy in the second order pushdown. It follows that the monadic second-order theory of micro-macro stack graphs is decidable.

2 Transition Graphs and Rewrite Systems

A *labeled transition graph* is $G = \langle \Sigma, S, L, \rho, s_0 \rangle$, where Σ is a finite set of labels, S is a (possibly infinite) set of states, $L : S \to \Sigma$ is a labeling function, $\rho \subseteq S \times S$ is a transition relation, and $s_0 \in S$ is an initial state. When $\rho(s, s')$, we say that s' is a *successor* of s, and s is a *predecessor* of s'. For a state $s \in S$, we denote by $G^s = \langle \Sigma, S, L, \rho, s \rangle$, the graph G with s as its initial state. An *s-computation* is an infinite sequence $s_0, s_1, \ldots \in S^\omega$ such that $s_0 = s$ and for all $i \geq 0$, we have $\rho(s_i, s_{i+1})$. An s-computation $s_0, s_1, \ldots$ induces the *s-trace* $L(s_0) \cdot L(s_1) \cdots$. The set $\mathcal{T}_s$ is the set of all s-traces and $\mathcal{T}_G = \mathcal{T}_{s_0}$ is the set of all *initialized* traces in G.

A *rewrite system* is $R = \langle \Sigma, V, Q, L, T, q_0, \top, \bot \rangle$, where Σ is a finite set of labels, V is a finite alphabet, Q is a finite set of states, $L : Q \times V^+ \to \Sigma$ is a labeling function, T is a finite set of rewrite rules, to be defined below, $q_0 \in Q$ is an initial state, $\top \in V$ is a store-top symbol, and $\bot \in V$ is a store-bottom symbol. We assume that the store-top symbol is moved whenever the store is extended. We assume that both store-bottom and store-top cannot be removed from nor added to the store and that the system does not try to go below the store-bottom or above the store-top. The set of *configurations* of the system is a subset of $Q \times V^+$. Intuitively, the system has finitely many control states and an unbounded store. Thus, in a configuration $(q, x) \in Q \times V^+$ we refer to q as the *control state* and to x as the *store*.

We consider here the well known *prefix-recognizable* systems and introduce *micro-macro stack systems*. In a *prefix-recognizable* system, each rewrite rule is $\langle q, \alpha, \beta, \gamma, q' \rangle \in Q \times \mathit{reg}(V) \times \mathit{reg}(V) \times \mathit{reg}(V) \times Q$, where $\mathit{reg}(V)$ is the set of regular expressions over V. Thus, $T \subseteq Q \times \mathit{reg}(V) \times \mathit{reg}(V) \times \mathit{reg}(V) \times Q$. For a word $w \in V^*$ and a regular expression $r \in \mathit{reg}(V)$ we write $w \in r$ to denote that w is in the language of the regular expression r. We note that the standard definition of prefix-recognizable systems does not include control states [15]. Indeed, a prefix-recognizable system without states can simulate a prefix-recognizable system with states by having the state as the first letter of the unbounded store. We use prefix-recognizable systems with control states for the sake of uniform notation.

In a *micro-macro stack* system (or *mMs* system for short), the set V contains a special symbol $\uparrow$, the set of states Q is partitioned into the set m of micro states and the set M of macro states, and every configuration contains exactly one occurrence of the symbol $\uparrow$. More formally, denote $V' = V \setminus \{\uparrow, \top, \bot\}$ then the set of possible store contents is $STORES = \{\top\} \cdot (V')^* \cdot \{\uparrow\} \cdot (V')^* \cdot \{\bot\} \cup \{\uparrow\} \cdot \{\top\} \cdot (V')^* \cdot \{\bot\}$ (recall that $\bot$ and $\top$ cannot be removed from nor added to the store). Each rewrite rule of an mMs system is $\langle q, A, act, q' \rangle \in Q \times (V \setminus \{\uparrow\}) \times ACT \times Q$ where $ACT \subseteq \{pop, md, mu, sp\} \cup \{push(z) \mid z \in (V')\}$. Here, *md*, *mu*, and *sp* stand for *move down*, *move up*, and *stay put*, respectively and we demand that the set ACT is finite. A configuration in $m \times STORES$ is a *micro configuration* and a configuration in $M \times STORES$ is a *macro configuration*. The labeling function L associates with every configuration of a rewrite system a label. The labeling depends only on one letter of the store x, as we explain below. Thus, we may write $L : Q \times V \to \Sigma$. When a rewrite system R is in configuration (q, x) such that $L(q, x) = a$ for some $a \in \Sigma$ we say that R *signals* a.

A rewrite system R induces a labeled transition graph $G_R = \langle \Sigma, Q \times V^+, L', \rho_R, (q_0, x_0) \rangle$. A labeled transition graph that is induced by a prefix-recognizable system is called a *prefix-recognizable graph*. A labeled transition graph that is induced by an mMs system is called a *micro-macro stack graph* (or *mMs graph* for short). For a prefix-recognizable system the states of G_R are the configurations of R, the initial store content is $x_0 = \top\bot$, and $\langle (q, z), (q', z') \rangle \in \rho_R$ if there is a rewrite rule in T leading from configuration (q, z) to configuration (q', z'). For an mMs system the states of G_R are the macro configurations of R, the initial store content is $x_0 = \top \uparrow \bot$, and $\langle (q, z), (q', z') \rangle \in \rho_R$ if there is a sequence of rewrite rules in T leading from configuration (q, z) to configuration (q', z') by a series of micro configurations. Formally, if R is a prefix-recognizable system, then $\rho_R((q, x \cdot y), (q', x' \cdot y))$ if there are regular languages α, β, and γ such that $x \in \alpha$, $y \in \beta$, $x' \in \gamma$, and $\langle q, \alpha, \beta, \gamma, q' \rangle \in T$. The labeling of a state (q, Az) is $L(q, A)$. In order to consider the case of an mMs system we need a few definitions. Let $\mathcal{H}(w_2 \uparrow z w_1 \bot) = z$, where $z \in V$, $\mathcal{H}(\uparrow \top w \bot) = \top$, and $\mathcal{H}(w \uparrow \bot) = \bot$. This describes the letter in the store read by the mMs system. In order to define the effect of applying a rewrite rule on a configuration we define the partial function $\mathcal{B} : STORES \times ACT \to STORES$. This function gives the new content of the store and is defined below. We assume that the system never moves up when it is at the top of the store nor down when it is at the bottom of the store.

- $\mathcal{B}(\uparrow \top z w \bot, pop) = \uparrow \top w \bot$.
- $\mathcal{B}(\uparrow \top w \bot, push(z)) = \uparrow \top z w \bot$.
- $\mathcal{B}(w_2 \uparrow z w_1 \bot, md) = w_2 z \uparrow w_1 \bot$.
- $\mathcal{B}(w_2 z \uparrow w_1 \bot, mu) = w_2 \uparrow z w_1 \bot$.
- $\mathcal{B}(s, sp) = s$.

A sequence of micro steps from a macro configuration (q, z) to macro configuration (q', z') is a sequence (q_1, z_1),

$(q_2, z_2), \ldots, (q_n, z_n)$ such that $(q, z) = (q_1, z_1)$, $(q', z') = (q_n, z_n)$, forall $1 < i < n$ we have $q_i \in m$, and forall $1 \leq i < n$ there exists a rewrite rule $\langle q_i, \mathcal{H}(z_i), act, q_{i+1} \rangle \in T$ and $\mathcal{B}(z_i, act) = z_{i+1}$. Finally, for a pair of macro configurations we have $\langle (q, z), (q', z') \rangle \in \rho_R$ if there exists a sequence of micro steps from (q, z) to (q', z'). Note that the mMs system collects information on the content of the stack and stores it in its control state. In particular, when standing on top of the store the mMs system always reads the symbol $\top$ and relies solely on the control state to choose its next action. The labeling of a state (q, z) is $L(q, \mathcal{H}(z))$.[1] We demand that the initial state of an mMs system be a macro state. This way we are ensured that the induced mMs graph has a single initial state. The work in this paper can be easily generalized for the case that there are many initial states.

Example 2.1 *Consider the following mMs* $R = \langle \{a, b, c, d\}, \{A, \top, \bot\}, Q, L, T, q_0, \top, \bot \rangle$ *where* $Q = \{q_0, q_g, q_a, q_b, q_c, q_d\}$*, the state* q_g *is the only micro state,* $L(q_\gamma, A) = L(q_\gamma, \top) = L(q_\gamma, \bot) = \gamma$ *for* $\gamma \in \{a, b, c, d\}$ *and* $L(q_0, A) = L(q_0, \bot) = d$*. Finally,* T *includes the following transitions.*

- $\langle q_0, \bot, mu, q_g \rangle$ *- signal* d *(state* q_0*) and move to a guessing state.*
- $\langle q_g, \top, push(A), q_g \rangle$ *- guess a number of* A *and put it on the store.*
- $\langle q_g, \top, md, q_a \rangle$ *- nondeterministically decide to start signaling* a*.*
- $\langle q_a, A, md, q_a \rangle$ *- go down the store while reading* A *and signaling* a*.*
- $\langle q_a, \bot, mu, q_b \rangle$ *- when reaching the bottom of the store start signaling* b*.*
- $\langle q_b, A, mu, q_b \rangle$ *- go up the store while reading* A *and signaling* b*.*
- $\langle q_b, \top, md, q_c \rangle$ *- when reaching the top of the store start signaling* c*.*
- $\langle q_c, A, md, q_c \rangle$ *- go down the store while reading* A *and signaling* c*.*
- $\langle q_c, \bot, sp, q_d \rangle$ *- when reaching the bottom of the store start signaling* d*.*
- $\langle q_d, \bot, sp, q_d \rangle$ *- signal* d *indefinitely.*

This system produces a 'star' of infinite degree. The center of this star is the initial state and each 'ray' is a sequence $da^n b^n c^n d^\omega$ *for some* n*. It does so by guessing some number of* A *and put them on the store. Then it moves up and down the store while signaling* a*,* b*, and* c*.*

Example 2.2 *Consider the mMs system* $R = \langle \{a, b\}, \{\top, \bot, A\}, \{q_0, r, l, p\}, L, T, q_0, \top, \bot \rangle$ *where* $m = \{r\}$*,* $M = \{q_0, l, p\}$*. The labeling function* L *associates with state* l *the symbol* a *and with states* q_0 *and* p *the symbol* b*. The set of rewrite rules* T *contains the following rules.*

- $\langle q_0, \bot, sp, r \rangle$ *- move to micro state* r*.*
- $\langle r, \bot, mu, l \rangle$ *- bottom of the store, switch to state* l*.*
- $\langle l, \top, sp, p \rangle$ *- top of the store, switch to state 'push'.*
- $\langle l, A, mu, l \rangle$ *- move up the store.*
- $\langle p, \top, push(A), r \rangle$ *- extend the store, switch to state* r*.*
- $\langle r, \top, md, r \rangle, \langle r, A, md, r \rangle$ *- move down.*

This system is in fact a unary counter. It 'outputs' 1, then 2, then 3, and so on ad-infinitum. It does so by going to the bottom of the store, then while it goes to the top of the store it signals a *with every move. When reaching the top of the store it extends the store with one symbol and signals one* b*. Thus, the first two* b *symbols are separated by one* a*. Then before the third* b *symbol there are 2* a*s, and generally the* i*th* b *is separated from the next* b *by the sequence* a^i*.*

It is quite easy to see that given a prefix-recognizable system R we can construct an mMs system R' such that G_R and $G_{R'}$ are isomorphic. The set of macro states of R' correspond to the set of states of R and the set of micro states of R' correspond to the states of automata that recognize the regular languages in the transitions of R. The mMs system R' mimics a transition $\langle q, \alpha, \beta, \gamma, q' \rangle$ of R by removing a sequence of letters from the store while simulating a run of the automaton for the regular language α on the removed sequence. Then the mMs system goes to the bottom of the stack and checks that what is left on the stack is a word in the regular language β. Finally, the mMs system guesses a word in γ (letter by letter) and adds it to the store[2]. In the sequel we show that the opposite is not true. There are mMs graphs with no isomorphic prefix-recognizable graph. The fact that an mMs system remembers in addition to the stack contents a location in the stack is enough to give it extra power over prefix-recognizable systems.

[1]The choice to set the labeling according to $\mathcal{H}(z)$ is the most general. We can emulate labeling according to the top of the store by remembering the top of the store as part of the control. Similarly, regular labeling, that depends on the membership of the entire store content in some regular language can be emulated by remembering the state of a deterministic finite automaton for the regular language on the store.

[2]A similar construction appears in Section 3.1.

3 Non prefix-recognizable mMs graphs

We give two examples of mMs graphs for which there exist no isomorphic prefix-recognizable graphs[3]. We use two methods to prove that these graphs are not prefix-recognizable. First, we use simple language considerations. We show that our graph, if represented by a prefix-recognizable system, induces a pushdown automaton over finite words that accepts a language that is not context-free. Second, we use a characterization of prefix-recognizable graphs by Blumensath [3]. We give another graph and prove that it does not satisfy the requirements of Blumensath. As not many proof techniques for proving that systems are not prefix-recognizable are known we include both proofs.

We define isomorphism with respect to two graphs $G = \langle \Sigma, S, L, \rho, s_0 \rangle$ and $G' = \langle \Sigma, S', L', \rho', s'_0 \rangle$, with the same alphabet. A bijection $I : S \to S'$ is an *isomorphism* between G and G' iff forall $s, t \in S$ we have $L(s) = L(I(s))$ and $\rho(s,t)$ iff $\rho'(I(s), I(t))$.

3.1 An mMs graph recognizing a non context free language

We use the well known non context-free language $\{a^n b^n c^n \mid n \in \mathcal{N}\}$ to prove that a graph is not prefix-recognizable. In fact, for many languages, if L is not context free but it can be recognized by a finite stack automaton, we can construct an mMs graph $G_{_L}$ whose 'language' is L. The graph $G_{_L}$ should be another example of an mMs graph that has no isomorphic prefix-recognizable graph.

We first need some definitions. A *pushdown automaton over finite words* (or *PD-NFW* for short) is $P = \langle \Sigma, \Gamma, Q, \rho, q_0, \perp, F \rangle$ where Σ is a finite input alphabet, Γ is a finite pushdown alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, $\perp \in \Gamma$ is a store-bottom symbol, and $F \subseteq Q$ is a set of accepting states. The transition function $\rho : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$ associates with every state $q \in Q$, input letter $\sigma \in (\Sigma \cup \{\epsilon\})$, and pushdown letter $A \in \Gamma$ a finite set of possible transitions $\rho(q, \sigma, A)$.

A *configuration* of a PD-NFW is a pair $(q, w) \in Q \times \Gamma^*$ where q is the state of the automaton and w is the content of the pushdown store. A *run* of a PD-NFW over a finite word $w \in \Sigma^*$ is a sequence of configurations and locations $r \in (Q \times \Gamma^* \times \{0, \ldots, |w|+1\})^*$ such that $r_0 = \langle (q_0, \perp), 0 \rangle$ and for every $0 \leq i < |r|$ we have $r_i = \langle (q, A \cdot x), j \rangle$, $r_{i+1} = \langle (q', y \cdot x), j + \Delta \rangle$ and either $\langle q', y \rangle \in \rho(q, w_j, A)$ and $\Delta = 1$, or $\langle q', y \rangle \in \rho(q, \epsilon, A)$ and $\Delta = 0$. A transition from $\langle (q,x), j \rangle$ to $\langle (q', x'), j \rangle$ is called an *ϵ-transition*. Otherwise it is an *advancing transition*. A run is accepting if $r_{|r|} = \langle f, x, |w|+1 \rangle$ for some state $f \in F$ and forall $j < |r|$ we have that $r_j = \langle s, y, k \rangle$ such that $k \leq |w|$. A word w is *accepted* by P if there exists an accepting run of P on w. The *language* $\mathcal{L}(P)$ of P is the set of words accepted by P.

An automaton is an NFW if $\Gamma = \{\perp\}$ and the transition function is restricted to $\rho : Q \times \Sigma \times \{\perp\} \to 2^{Q \times \{\perp\}}$. In this case we write $N = \langle \Sigma, Q, \rho, q_0, F \rangle$ and $\rho : Q \times \Sigma \to 2^Q$. In case that for every letter $\sigma \in \Sigma$ and state $q \in Q$ we have $|\rho(q, \sigma)| = 1$ we say that N is deterministic (DFW).

Consider the mMs system $R = \langle \{a,b,c,d\}, \{A, \top, \perp\}, Q, L, T, q_0, \top, \perp \rangle$ from Example 2.1.

Claim 3.1 *There is no prefix-recognizable system S such that G_S is isomorphic to G_R.*

Proof: Given a PD-NFW $P = \langle \Sigma, \Gamma, Q, \rho, q_0, \perp, F \rangle$ we construct the graph G_P where the set of vertices is the set of configurations of P from which there exists an advancing transition. An edge connects (q, w) to (q', w') if there exists an advancing transition followed by a sequence of ϵ transition leading from (q, w) to (q', w'). Finally, we label a configuration (q, Aw) by the set of letters $\sigma \in \Sigma$ such that $\rho(q, \sigma, A) \neq \emptyset$. Formally, let $G_P = \langle 2^\Sigma, S, L, \rho_P, s_0 \rangle$ where $S = \{(q, A \cdot w) \in Q \times \Gamma^+ \mid \exists \sigma \in \Sigma \text{ s.t. } \rho(q, \sigma, A) \neq \emptyset\}$, $s_0 = (q_0, \perp)$, and $L(q, Aw) = \{\sigma \mid \rho(q, \sigma, A) \neq \emptyset\}$. We have $((q, Aw), (q', w')) \in \rho_P$ if there exists a run $(q, Aw, 0), (q_1, w_1, 1), \ldots, (q_n, w_n, 1), (q', w', 1)$ on a word $\sigma \in L(q, Aw)$ (note that $|\sigma| = 1$ and that this run is not necessarily accepting).

It is known that the graph of a prefix-recognizable system is isomorphic to the configuration graph of a PD-NFW [41, 3]. We extend this result by showing that given a prefix-recognizable system $R_{pr} = \langle \Sigma_{pr}, V_{pr}, Q_{pr}, L_{pr}, T_{pr}, q_0^{pr}, \top, \perp \rangle$ we can construct a PD-NFW $P = \langle \Sigma, \Gamma, Q_{pd}, \rho, q_0^{pd}, \perp, F \rangle$ such that every state s of G_P has $|L(s)| = 1$ and if we restrict our attention to states reachable from $(q_0^{pd}, \perp)$ in G_P and from $(q_0^{pr}, \top\perp)$ in $G_{R_{pr}}$ the two are isomorphic.

We first need a few definitions Consider the prefix-recognizable system $R_{pr} = \langle \Sigma_{pr}, V_{pr}, Q_{pr}, L_{pr}, T_{pr}, q_0^{pr}, \top, \perp \rangle$. For a rewrite rule $t_i = \langle q, \alpha_i, \beta_i, \gamma_i, q' \rangle \in T_{pr}$, let $\mathcal{U}_\lambda = \langle V_{pr}, Q_\lambda, \eta_\lambda, q_\lambda^0, F_\lambda \rangle$, for $\lambda \in \{\alpha_i, \beta_i, \gamma_i\}$, be the NFW for the language of the regular expression λ. We assume that all initial states have no incoming edges and that all accepting states have no outgoing edges. We collect all the states of all the automata for α, β, and γ regular expressions. Formally, $Q_\alpha = \bigcup_{t_i \in T} Q_{\alpha_i}$, $Q_\beta = \bigcup_{t_i \in T} Q_{\beta_i}$, and $Q_\gamma = \bigcup_{t_i \in T} Q_{\gamma_i}$. Let $\{\beta_1, \ldots, \beta_n\}$ be the set of regular expressions β_i such that there is a rewrite rule $\langle q, \alpha_i, \beta_i, \gamma_i, q' \rangle \in T_{pr}$. Let $\mathcal{D}_{\beta_i} = \langle V_{pr}, D_{\beta_i}, \eta_{\beta_i}, q_{\beta_i}^0, F_{\beta_i} \rangle$ be the deterministic automaton for the **reverse** language of β_i. For a word $x \in V_{pr}^*$, we denote by $\eta_{\beta_i}(x)$ the unique state that $\mathcal{D}_{\beta_i}$ reaches after reading the word x. Let $\Upsilon = V_{pr} \times \Pi_{1 \leq i \leq n} D_{\beta_i}$. For a letter $\upsilon \in \Upsilon$, let $\upsilon[i]$, for $i \in \{0, \ldots n\}$, denote the i-th element in υ (that is, $\upsilon[0] \in V_{pr}$ and $\upsilon[i] \in D_{\beta_i}$ for

[3]In fact, for our graph there does not exist a *bisimilar* prefix-recognizable graph. Our proof can be extended for the case of bisimilarity.

$i > 0$). Let $\eta_{\mathcal{D}} : (\Upsilon \times V) \to \Upsilon$ denote the effect of reading a letter V on the states of the automata $\mathcal{D}_{\beta_i}$. Formally, $\eta_{\mathcal{D}}(v, A) = \langle A, \eta_{\beta_1}(v[1], A), \ldots, \eta_{\beta_n}(v[n], A)\rangle$. We use $\perp = \langle \perp, q_0^{\beta_1}, \ldots, q_0^{\beta_n}\rangle$ as store-bottom symbol of the PD-NFW. Given a word $w = w_0, \ldots, w_m \in V_{pr}^* \cdot \perp$ we denote by $r^{\beta_i}(w) = r^0_{\beta_i}, \ldots, r^m_{\beta_i}$ the unique run of $\mathcal{D}_{\beta_i}$ on the reverse of w. We denote by $w \times r^{\beta_1}(w) \times \cdots \times r^{\beta_n}(w)$ the word $v_0, v_1, \ldots, v_m$ such that forall $0 \leq j < m$ we have $v_j[0] = w_j$ and $v_m[0] = \perp$ and forall $0 \leq j \leq m$ and $1 \leq i \leq n$ we have $v_j[i] = r^{m-j}_{\beta_i}$.

Consider the PD-NFW $P = \langle \Sigma_{pr}, \Upsilon, Q_{pr} \cup (T_{pr} \times (Q_\alpha \cup Q_\gamma)), \rho, q_0^{pr}, \perp, \emptyset\rangle$ where the transition function is defined as follows. For a state $q \in Q_{pr}$ and $v \in \Upsilon$ we have $\rho(q, \sigma, v) = \{\langle (t, s), v\rangle \mid t = \langle q, \alpha, \beta, \gamma, q'\rangle \text{ and } s = q^0_\alpha\}$ for $\sigma = L_{pr}(q, v[0])$ and $\rho(q, \sigma, v) = \emptyset$ for $\sigma \neq L_{pr}(q, v[0])$. For a state $(t, s) \in T_{pr} \times Q_\alpha$ where $t = \langle q, \alpha, \beta, \gamma, q'\rangle$ we have $\rho((t, s), \epsilon, v) = \{\langle (t, s'), \epsilon\rangle \mid s' \in \eta_\alpha(s, v[0])\} \cup \{\langle (t, s'), v\rangle \mid s \in F_\alpha, s' \in F_\gamma, \text{ and } v[i] \in F_\beta\}$ and $\rho((t, s), \sigma, v) = \emptyset$ for $\sigma \in \Sigma_{pr}$. For a state $(t, s) \in T_{pr} \times Q_\gamma$ where $t = \langle q, \alpha, \beta, \gamma, q'\rangle$ we have $\rho((t, s), \epsilon, v) = \{\langle (t, s'), v'v\rangle \mid s \in \eta_\gamma(s', v'[0]) \text{ and } v' = \eta_{\mathcal{D}}(v, v'[0])\} \cup \{\langle q', v\rangle \mid s = q^0_\gamma\}$ and $\rho((t, s), \sigma, v) = \emptyset$ for $\sigma \in \Sigma_{pr}$. We show that the subgraphs of G_P and $G_{R_{pr}}$ that are reachable from the respective initial states are isomorphic. Consider the function $I : Q_{pr} \times V^* \to Q_{pd} \times \Upsilon^*$ where $I(q, x) = (q, x \times r^{\beta_1}(x) \times \cdots \times r^{\beta_n}(x))$.

For every state $q \in Q_{pr}$ the only letter $\sigma \in \Sigma_{pr}$ for which $\rho(q, \sigma, v)$ is defined is $L(q, v[0])$, hence $L_{pr}(q, w) = L_{G_P}(I(q, w))$. For every state $(t, q) \in T_{pr} \times (Q_\alpha \cup Q_\gamma)$ we have $\delta(q, \sigma, v) = \emptyset$ for $\sigma \neq \epsilon$. By definition it is clear that I is an injection. We show that I is a surjection. Notice that, the only way to extend the store is by a transition $\langle (t, s'), v'v\rangle$. However, we demand that $v' = \eta_{\mathcal{D}}(v, v'[0])$. Hence, the only states reachable from $(q_0^{pd}, \perp)$ are states in the range of I. We show by induction on the distance from $(q_0^{pd}, \perp)$ that every configuration (q, w') of P is in the range of I. For $(q_0^{pd}, \perp)$ the proof is immediate. Assume that all configurations whose distance is k are in the range of I. Consider a state (q, w) of G_P of distance $k + 1$ from $(q_0^{pd}, \perp)$. By induction there exists a reachable state (q', w') of $G_{R_{pr}}$ such that $(I(q', w'), (q, w)) \in \rho_P$. Denote $s = I(q', w')$ and $s' = (q, w)$. Then there is a sequence s, $\langle (t, q^0_\alpha), w_0\rangle$, $\langle (t, q^1_\alpha), w_1\rangle$, …, $\langle (t, q^m_\alpha), w_m\rangle$, $\langle (t, q^{m'}_\gamma), x_{m'}\rangle$, …, $\langle (t, q^0_\gamma), x_0\rangle$, s' such that w_0 is the store content of s, $x_0 = w$ is the store content of s', $w_m = x_{m'}$ is a common suffix of w_m and x_0 such that $w_m \in \beta$ (note that $\mathcal{D}_\beta$ runs from the end of w_m to the start of w_m and recognizes words whose reverse is in β). and the runs $q^0_\alpha, \ldots, q^m_\alpha$ and $q^0_\gamma, \ldots, q^{m'}_\gamma$ are witnesses that the prefixes of w_0 and x_0 are in α and γ respectively. We conclude that the state $(q', w[0])$ is reachable in $G_{R_{pr}}$ and $I(q', w[0]) = s'$. In a similar manner we show that $(s, s') \in \rho_{T_{pr}}$ iff $(I(s), I(s')) \in \rho_P$.

Assume by contradiction that there exists a prefix-recognizable system whose graph is isomorphic to G_R. Let $R_{pr} = \langle \{a, b, c, d\}, V, Q, L, T, q_0, \top, \perp\rangle$ be the prefix-recognizable system that produces this graph. Let $P = \langle \{a, b, c, d\}, \Upsilon, Q', \rho, q_0', \perp, \emptyset\rangle$ be the PD-NFW such that G_P is isomorphic to $G_{R_{pr}}$. Consider the PD-NFW $P' = \langle \{a, b, c, d\}, \Upsilon, Q' \cup \{acc\}, \rho', q_0', \perp, \{acc\}\rangle$, where for every state $q \in Q'$ and letter $A \in \Upsilon$ we define $\rho'(q, d, A) = \rho(q, d, A) \cup \{\langle acc, A\rangle\}$ and $\rho'(q, \gamma, A) = \rho(q, \gamma, A)$ for $\gamma \neq d$. Thus, whenever P' reads the letter d it can enter an accepting state. Is is simple to see that the language of P' is $\{d\} \cdot \{a^n b^n c^n \mid n \geq 1\} \cdot \{d\}^+$. However, $\{a^n b^n c^n \mid n \in \mathcal{N}\}$ is not a context-free language [29]. contradiction. $\square$

3.2 An mMs graph violating prefix-recognizable representation characterization

We give another example of an mMs graph that is not prefix-recognizable. This time, we use characterization of prefix-recognizable graphs by Blumensath [3] to show that the graph is not prefix-recognizable.

We first need some definitions. Let $\mathcal{T}_2 = \{0, 1\}^*$ denote the full binary tree and let $succ_0(x, y)$ and $succ_1(x, y)$ denote the successor relations (that is, for $x, y \in \{0, 1\}^*$ we have $succ_0(x, y)$ true iff $y = x \cdot 0$, and similarly for $succ_1$). We define Monadic Second Order Logic (MSOL) over $\langle \mathcal{T}_2, succ_0, succ_1\rangle$ as follows. Let $x, y, \ldots$ denote state variables and $X, Y, \ldots$ denote set variables. *Atomic formulas* are $x \in X$ and $succ_i(x, y)$ for $i \in 0, 1$. *MSOL formulas* are the formulas obtained from atomic formulas by closing them under Boolean connectives and existential quantification over vertex or set variables. For a full exposition of MSOL we refer the reader to [42].

A graph $G = \langle \Sigma, S, L, \rho, s_0\rangle$ is *MSOL interpretable* in the structure $\langle \mathcal{T}_2, succ_0, succ_1\rangle$ if there exist MSOL formulas $\varphi_\rho(x, y)$, φ_{s_0}, and for every $\sigma \in \Sigma$ we have $\varphi_\sigma(x)$ such that the graph G' defined below is isomorphic to G. Let $G' = \langle \Sigma, S', L', \rho', s_0'\rangle$ such that $S' = \{t \in \mathcal{T}_2 \mid \exists x.\varphi_\rho(t, x) \vee \varphi_\rho(x, t)\}$, $S_0 = \{t \in \mathcal{T}_2 \mid \varphi_{s_0}(t)\}$ is a singleton $S_0 = \{s_0'\}$, $\rho = \{(x, y) | \varphi_\rho(x, y)\}$ and finally, for every state $s \in S'$ we have $l(s) = \{\sigma \mid \varphi_\sigma(s)\}$ is a singleton and $l(s) = \{L(s)\}$. We overload notation and refer to $\langle \varphi_\rho, \varphi_{s_0}, (\varphi_\sigma)_{\sigma \in \Sigma}\rangle$ both as the interpretation in $\mathcal{T}_2$ and the graph that it induces.

Theorem 3.2 [1, 3] *A graph G is prefix-recognizable iff it is MSOL interpretable in $\langle \mathcal{T}_2, succ_0, succ_1\rangle$.*

Consider a prefix-recognizable graph $G = \langle \Sigma, S, L, \rho, s_0\rangle$ and a state $s \in S$. Let $Pre^*(s)$ denote the set of states from which s is reachable. Formally, for $S' \subseteq S$ we define $Pre_0(S') = S'$, $Pre_{i+1}(S') =$

$\{t \mid \exists t' \in Pre_i(S') \text{ and } (t,t') \in \rho\}$, and $Pre^*(S') = \bigcup_{i\geq 0} Pre_i(S')$. If $S' = \{s\}$ is a singleton we write $Pre^*(s)$. We define a relation on the states of G, forall s,t we have $s \leq t$ iff $s \in Pre^*(t)$. Let $\langle \varphi_\rho, \varphi_{s_0}, (\varphi_\sigma)_{\sigma\in\Sigma}\rangle$ be some interpretation of G in $\mathcal{T}_2$ and let $I : S \to \mathcal{T}_2$ be the bijection associating G and $\langle \varphi_\rho, \varphi_{s_0}, (\varphi_\sigma)_{\sigma\in\Sigma}\rangle$. For a state $s \in S$, let $|I(s)|$ denote the length of $I(s) \in \mathcal{T}_2$.

Lemma 3.3 [3] *Let G be a prefix-recognizable graph. If the relation $s \leq t$ is a well order on the states of G, then for every interpretation of G in $\mathcal{T}_2$ with bijection I we have $|I(a_n)| = \Theta(n)$ where a_n is the nth element in the ordering.*

Consider the mMs system $R = \langle\ \{a,b\},\ \{\top,\bot,A\},\ \{q_0,r,l,p\},\ L,\ T,\ q_0,\ \top,\ \bot\ \rangle$ from Example 2.2.

Theorem 3.4 *The system R above is not a prefix-recognizable system.*

Proof: Lemma 3.3 gives information on the size of the encoding of states of a well ordered prefix-recognizable graph. The graph above is clearly well ordered. We show a subgraph that is also well ordered. Both the graph and its subgraph are embedded in $\mathcal{T}_2$ and share the same encoding of the states. In each graph we get information on the size of the encoding and this information is contradictory.

Suppose by contradiction that there exists a prefix-recognizable system inducing a graph isomorphic to G_R. Let $G = \langle \Sigma, S, L, \rho, s_0\rangle$ be the graph of this prefix-recognizable system and let $\langle \varphi_\rho, \varphi_{s_0}, \varphi_a, \varphi_b\rangle$ be some interpretation of G in $\mathcal{T}_2$. Let $I : S \to \mathcal{T}_2$ denote the isomorphism associating the two. As mentioned the system above produces an infinite path of nodes. Let b_i denote the state x such that $|Pre^*(x) \cap \{y \mid L(y) = b\}| = i$.

Denote by $G' = \langle\{b\}, S', L', \rho', s_0\rangle$ the graph consisting of the states $S' = \{b_i \mid i \geq 1\}$ and the edges $\rho' = \{(b_i, b_{i+1}) \mid i \geq 1\}$. For every state $b_i \in S'$ we have $L'(b_i) = b$ and as $s_0 = b_1$ we use the same initial state in G'. We show that this graph is also interpretable in $\mathcal{T}_2$. Hence, it is also a prefix-recognizable graph and Lemma 3.3 applies for it. Consider the MSOL formulas in Figure 1. For states x', y' we have $R_{a^*}(x', y')$ true iff y' is reachable from x' along a path that visits only states labeled a. For states x, y we have $R_{ba^*b}(x,y)$ true iff both x and y are b states and either y is a successor of x or y is reachable from x along a path that visits only states labeled a.

Consider the structure $\langle R_{ba^*b}, \varphi_{s_0}, \mathbf{true}\rangle$. It is isomorphic to the graph G'. Let $I' : S' \to \mathcal{T}_2$ denote the isomorphism between G' and $\langle R_{ba^*b}, \varphi_{s_0}, \mathbf{true}\rangle$ (actually, I' is I restricted to S'). From Theorem 3.2 it follows that G' is a prefix-recognizable graph. It is also the case that G' is well ordered. According to Lemma 3.3 we have $|I(b_i)| = \Theta(i)$.

As mentioned the graph G consists of a single path. Hence, the relation $\leq$ is a well order on the states of G. The state b_i is the $\frac{i(i+1)}{2}$ state in this order. According to Lemma 3.3 we have $|I(b_i)| = \Theta(i^2)$. Contradiction. □

4 Model checking over mMs systems

We use the automata-theoretic approach to model checking. In the branching time framework, the automata-theoretic approach reduces the model checking problem to the emptiness problem of an alternating tree automaton over 1-letter alphabet. The alternating tree automaton is the combination of the system and the specification [35]. In the linear time framework, the automata theoretic approach reduces the model checking problem into the emptiness problem of a nondeterministic word automaton. The nondeterministic word automaton is the combination of the system and the negation of the specification [45]. Similarly, for mMs systems we solve model checking by a reduction to the respective emptiness problem. As our systems have a stack we use stack automata. Due to lack of space we include only an algorithm for branching time model checking.

In order to give the most general algorithm we assume that the specification is given as an automaton. In the branching time framework the specification is given as an alternating graph automaton (that accepts all possible graphs that satisfy the specification, see below). Algorithms for converting μ-calculus, CTL, CTL*, and LTL and $S1S$ to automata can be found in the literature [5, 46, 30, 44, 35].

4.1 Definitions

Graph Automata. Given a finite set Υ of directions, an Υ-*tree* is a set $T \subseteq \Upsilon^*$ such that if $\upsilon \cdot x \in T$, where $\upsilon \in \Upsilon$ and $x \in \Upsilon^*$, then also $x \in T$. The elements of T are called *nodes*, and the empty word ε is the *root* of T. For every $\upsilon \in \Upsilon$ and $x \in T$, the node x is the *parent* of $\upsilon \cdot x$ and $\upsilon \cdot x$ is a *child* of x. If $z = x \cdot y \in T$ then z is a descendant of y. There is a natural partial order induced on the nodes of the tree, $x \leq y$ iff y is a descendant of x. An Υ-tree T is a *full infinite tree* if $T = \Upsilon^*$. A *path* π of a tree T is an infinite set $\pi \subseteq T$ such that $\varepsilon \in \pi$ and for every $x \in \pi$ there exists a unique $\upsilon \in \Upsilon$ such that $\upsilon \cdot x \in \pi$. Our definitions here reverse the standard definitions (e.g., when $\Upsilon = \{0,1\}$, the successors of 0 are 00 and 10, rather than 00 and 01).

Given two finite sets Υ and Σ, a Σ-*labeled* Υ-*tree* is a pair $\langle T, \tau\rangle$ where T is an Υ-tree and $\tau : T \to \Sigma$ maps each node of T to a letter in Σ. When Υ and Σ are not important or clear from the context, we call $\langle T, \tau\rangle$ a labeled tree.

For a finite set X, let $\mathcal{B}^+(X)$ be the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$), where we also allow the formulas $\mathbf{true}$ and $\mathbf{false}$, and, as usual, $\wedge$ has precedence over $\vee$. For a set $Y \subseteq X$ and a formula $\theta \in \mathcal{B}^+(X)$, we say that

$$R_{a^*}(x', y') = \forall X \left[\left(\begin{array}{c} (\forall z. z \in X \to \varphi_a(z)) \land \\ (\forall z, z'. z \in X \land \varphi_\rho(z, z') \land \varphi_a(z') \to z' \in X) \end{array} \right) \to (x' \in X \to y' \in X) \right] \land \varphi_a(x') \land \varphi_a(y')$$

$$R_{ba^*b}(x, y) = \varphi_b(x) \land \varphi_b(y) \land [\varphi_\rho(x, y) \lor \exists x', y' \, (R_a(x', y') \land \varphi_\rho(x, x') \land \varphi_\rho(y', y))]$$

Figure 1. Interpretation in $\mathcal{T}_2$

Y *satisfies* θ iff assigning **true** to elements in Y and assigning **false** to elements in $X \setminus Y$ makes θ true.

An alternating automaton on labeled transition graphs (*graph automaton*, for short) [49] is a tuple $\mathcal{A} = \langle \Sigma, Q, \eta, q_0, \alpha \rangle$, where Σ, Q, q_0 are as in PD-NFW, α specifies the acceptance condition, and $\eta : Q \times \Sigma \to \mathcal{B}^+(\{\varepsilon, \Diamond, \Box\} \times Q)$ is the transition function. Intuitively, when $\mathcal{A}$ is in state q and it reads a state s of G, fulfilling an atom $\langle \Diamond, p \rangle$ (or $\Diamond p$, for short) requires $\mathcal{A}$ to send a copy in state p to some successor of s. Similarly, fulfilling an atom $\Box p$ requires $\mathcal{A}$ to send copies in state p to all the successors of s. The atom $\langle \varepsilon, p \rangle$ (or p, for short) requires $\mathcal{A}$ to send a copy in state p to the node s itself. Thus, like symmetric automata [22, 49], graph automata cannot distinguish between the various successors of a state and treat them in an existential or universal way.

A run of a graph automaton $\mathcal{A}$ on a labeled transition graph $G = \langle \Sigma, S, L, \rho, s_0 \rangle$ is a labeled tree in which every node is labeled by an element of $S \times Q$. A node labeled by (s, q), describes a copy of the automaton that is in the state q of $\mathcal{A}$ and reads the state s of G. Formally, a run is a Σ_r-labeled Γ-tree $\langle T_r, r \rangle$, where Γ is some set of directions, $\Sigma_r = S \times Q$, and $\langle T_r, r \rangle$ satisfies the following:

1. $\varepsilon \in T_r$ and $r(\varepsilon) = (s_0, q_0)$.
2. Consider $y \in T_r$ with $r(y) = (s, q)$ and $\delta(q, L(s)) = \theta$. Then there is a (possibly empty) set $Y \subseteq (\{\varepsilon, \Diamond, \Box\} \times Q)$, such that Y satisfies θ, and for all $\langle c, q' \rangle \in Y$, the following hold:
    - If $c = \varepsilon$, then there is $\gamma \in \Gamma$ such that $\gamma \cdot y \in T_r$ and $r(\gamma \cdot y) = (s, q')$.
    - If $c = \Box$, then for every successor s' of s, there is $\gamma \in \Gamma$ such that $\gamma \cdot y \in T_r$ and $r(\gamma \cdot y) = (s', q')$.
    - If $c = \Diamond$, then there is a successor s' of s and $\gamma \in \Gamma$ such that $\gamma \cdot y \in T_r$ and $r(\gamma \cdot y) = (s', q')$.

A run $\langle T_r, r \rangle$ is *accepting* if all its infinite paths satisfy the acceptance condition. We consider here *parity* acceptance conditions [23]. A parity condition $\alpha = \{F_1, F_2, \ldots, F_m\}$ is a partition of Q. The number m of sets is called the *index* of $\mathcal{A}$. Given a run $\langle T_r, r \rangle$ and an infinite path $\pi \subseteq T_r$, let $\mathit{inf}(\pi) \subseteq Q$ be such that $q \in \mathit{inf}(\pi)$ if and only if there are infinitely many $y \in \pi$ for which $r(y) \in S \times \{q\}$. That is, $\mathit{inf}(\pi)$ contains exactly all the states that appear infinitely often in π. A path π satisfies the condition F if the minimal i such that $\mathit{inf}(\pi) \cap F_i \neq \emptyset$ is even. A run $\langle T_r, r \rangle$ is accepting if all paths π in T_r are accepting. The graph G is accepted by $\mathcal{A}$ if there is an accepting run on it. We denote by $\mathcal{L}(\mathcal{A})$ the set of all graphs that $\mathcal{A}$ accepts.

We use graph automata as our branching-time specification language. We say that a labeled transition graph G satisfies a graph automaton $\mathcal{A}$, denoted $G \models \mathcal{A}$, if $\mathcal{A}$ accepts G. Graph automata are as expressive as μ-calculus [30, 49]. In particular, we have the following.

Theorem 4.1 *Given a μ-calculus formula ψ, of length n and alternation depth k, we can construct a graph parity automaton $\mathcal{A}_\psi$ such that $\mathcal{L}(\mathcal{A}_\psi)$ is exactly the set of graphs satisfying ψ. The automaton $\mathcal{A}_\psi$ has n states and index k.*

Stack Automata. A *stack alternating automaton* on Υ-trees (or ST-APT) is $M = \langle \Sigma, V, Q, \delta, q_0, \top, \bot, \alpha \rangle$ where Σ, Q, q_0, and α are as in graph automata, V is the stack alphabet, and $\top$ and $\bot$ are the store-top and store-bottom symbols (that cannot be removed from nor added to the stack). The transition function is $\delta : Q \times \Sigma \times V \to B^+(Q \times ACT \times \Upsilon)$ where ACT is the set of possible actions as defined for mMs systems. We also use the set of stack configurations $STORES$, the function $\mathcal{H}$ and the function $\mathcal{B}$ as defined for mMs systems.

A run of M on a Σ-labeled Υ-tree $\langle \Upsilon^*, \tau \rangle$ is a $\Upsilon^* \times Q \times STORES$-labeled Γ-tree $\langle T, r \rangle$ where Γ is some set of directions and $\langle T, r \rangle$ satisfies the following.

- $\varepsilon \in T$ and $r(\varepsilon) = (\epsilon, q_0, \top \uparrow \bot)$.
- Consider $x \in T$ with $r(x) = (y, q, z)$ and $\delta(q, \tau(y), \mathcal{H}(z)) = \theta$. Then there exists (a possibly empty) set $((q_1, act_1, \Delta_1), \ldots, (q_d, act_d, \Delta_d)) \models \theta$ and x has d successors $x_1, \ldots x_d$ such that $r(x_j) = (\Delta_j y, q_j, \mathcal{B}(z, act_j))$.

A run is accepting if every infinite path in $\langle T, r \rangle$ satisfies the parity acceptance condition. We are interested in alternating stack automata with 1-letter input alphabet.

An alternating stack automaton with 1-letter input alphabet, the location on the input tree and the structure of the input tree are not important. Hence, we can consider automata reading infinite words and we can write $\delta : Q \times V \to B^+(Q \times ACT)$. Accordingly, runs of such automata are

$Q \times STORES$-labeled trees. We denote alternating stack parity automata with 1-letter input alphabet as ST-APW$_1$.

Harel and Raz show that the emptiness problem of nondeterministic stack automata on infinite trees is decidable in quintuply exponential time [28]. Their methods can be easily extended to ST-APW$_1$. A combination of [28] and [33] gives a double exponential algorithm for the emptiness of ST-APW$_1$.

Theorem 4.2 *The emptiness of an ST-APW$_1$ N can be determined in time double exponential in the size of N.*

4.2 Branching time model checking

We use the automata-theoretic approach to branching time model checking [35]. Given an mMs system R and a graph automaton $\mathcal{A}$, we construct an ST-APW$_1$ N such that $\mathcal{L}(N) \neq \emptyset$ iff $G_R \models \mathcal{A}$.

The idea behind the construction is that the stack automaton N holds the control structure of $\mathcal{A}$ and the control structure of R within its finite control. When R is in a micro state, our stack automaton mimics R without changing the control state of $\mathcal{A}$. When the control of R is in a macro state, our stack automaton mimics a transition of $\mathcal{A}$ reading the new macro state. Formally we have the following.

Let $\mathcal{A} = \langle \Sigma, Q, \eta, q_0, \alpha \rangle$ be a graph automaton, $R = \langle \Sigma, V, S, L, T, s_0, \top, \bot \rangle$ be an mMs system where $S = m \cup M$, and $D = \{\varepsilon, \Diamond, \Box\}$. We construct the ST-APW$_1$ $N = \langle \{a\}, V, Q', \delta, q'_0, \top, \bot \alpha' \rangle$ where $Q' = (D \times Q \times S)$, the initial state $q'_0 = (\varepsilon, q_0, s_0)$ and if $\alpha = \{F_1, ..., F_k\}$ (wlog k is odd) then $\alpha' = \{D \times F_1 \times M, \ldots, D \times F_k \times M, \{\Box\} \times Q \times m, \{\Diamond\} \times Q \times m\}$. We shorten notations by writing $(\Diamond q, s)$ instead of $(\Diamond, q, s)$ and similarly $(\Box q, s)$ and (q, s). The transition function δ is defined for every state $(c, q, s) \in D \times Q \times S$ and letter $A \in V$ as follows.

- For $c \in \{\Diamond, \Box\}$, the transition function is in Figure 2.
- For $c = \varepsilon$, then $s \in M$ is a macro state and we obtain $\delta((c, q, s), A)$ from $\eta(q, L(s, A))$ by replacing every atom $\Diamond q'$ in $\eta(q, L(s, A))$ by $\langle (\Diamond q', s), sp \rangle$, every atom $\Box q'$ by $\langle (\Box q', s), sp \rangle$, and every atom q' by $\langle (q', s), sp \rangle$.

Note that by including $\{\Box\} \times Q \times m$ as the maximal even set in α' and $\{\Diamond\} \times Q \times m$ as the maximal odd set in α' we ensure that when quantifying universally over successors, our ST-APW$_1$ does not care about infinite sequences of micro states and when quantifying existentially over successors, our ST-APW$_1$ does not allow to follow an infinite sequence of micro states. In the full version, we show that $\mathcal{L}(N) \neq \emptyset$ iff $G_R \models \mathcal{A}$ by translating a run tree of N on the word a^ω to a run tree of $\mathcal{A}$ on G_R and vice versa.

Claim 4.3 *$\mathcal{L}(N) \neq \emptyset$ iff $G_R \models \mathcal{A}$.*

Corollary 4.4 *The model checking problem of an mMs system and a graph automaton can be solved in time double exponential in both the size of the system and the size of the automaton.*

5 Emptiness of stack automata

The emptiness problem of an automaton is to determine whether it accepts some input. In this section we solve the emptiness problem for ST-APW$_1$ by a reduction to the emptiness of PD-APW$_1$. Our reduction enhances the reduction of ST-NPT to PD-NPT given in [28]. An algorithm for checking the emptiness of PD-APW$_1$ is given in [33].

Given a ST-APW$_1$ we construct a PD-APW$_1$ that records extra information on the pushdown store. This information is the summary of the stack automaton's actions when it reads the stack's contents. We want to know which parity sets are visited by the stack automaton and in what state does it 'get out' from the stack. As the stack automaton is alternating, when reading the stack's contents it may spawn new copies of the automaton. Hence, for every state we record the possible sets of states with which the stack automaton can surface after reading the contents of the stack. We note that the top-of-store symbol is a buffer between the possibility to change the contents of the stack and the meaningful information in the stack. Our pushdown automaton uses basically the same states as the stack automaton. However, it duplicates the set of states. One copy mimics the behavior of the stack automaton when reading the top-of-store symbol. The other copy mimics the behavior of the stack automaton when standing just before the top of the store and reading the last meaningful letter on the stack. We start with the simpler problem of solving the emptiness of ST-NBW and then extend it to the emptiness of ST-APW$_1$.

We reduce the emptiness of an ST-NBW to the emptiness of PD-NBW. Consider an ST-NBW $S = \langle \Sigma, V, Q, \delta, q_0, \top, \bot, \alpha \rangle$ where $\alpha \subseteq Q$ and $\delta : Q \times \Sigma \times V \to 2^{Q \times ACT}$. It is easy to see that the emptiness problem of an ST-NBW can be reduced to that of an ST-NBW with one letter input alphabet. We assume that $|\Sigma| = 1$ and ignore this component. We construct a PD-NBW P such that $L(P) = \emptyset$ iff $L(S) = \emptyset$. The pushdown alphabet of P is $V \times \Gamma$ where $\Gamma = 2^{Q \times \{0,1\} \times (Q \cup \{acc\})}$. Thus, every letter in Γ is a $\{0, 1\}$ labeled graph G on the set of vertices Q. For a state $q \in Q$ the set of nodes that are neighbors of q in G are the nodes reachable from q after a finite detour in the store. An infinite detour into the store is either accepting, in which case we add *acc* to the options of q or rejecting in which case we ignore it. Whenever state s chooses to enter the store in state q we choose one of the neighbours of q in G and move to it. Visits to the acceptance set are monitored using the label on the edges. Label 0 indicates that the path does not visit α and label 1 indicates that the path does visit

$$\delta((c,q,s),A) = \begin{cases} \bigvee_{\substack{\langle s,A,act,s'\rangle \in T \\ \text{and } s' \in m}} \langle(\Diamond q,s'),act\rangle \vee \bigvee_{\substack{\langle s,A,act,s'\rangle \in T \\ \text{and } s' \in M}} \langle(q,s'),act\rangle & \text{If } c = \Diamond \\ \bigwedge_{\substack{\langle s,A,act,s'\rangle \in T \\ \text{and } s' \in m}} \langle(\Box q,s'),act\rangle \wedge \bigwedge_{\substack{\langle s,A,act,s'\rangle \in T \\ \text{and } s' \in M}} \langle\langle q,s'\rangle,act\rangle & \text{If } c = \Box \end{cases}$$

Figure 2. Transition of micro state

α. More formally, we have the following.

Let $P = \langle \Sigma, V \times \Gamma, Q', \delta', q'_0, (\perp, \emptyset), \alpha' \rangle$ where the set of states is $Q' = (\{in, on\} \times Q \times \{0,1\}) \cup \{acc\}$, the initial state is $q'_0 = (in, q_0, 0)$, and the set of accepting states is $\alpha' = (\{in, on\} \times Q \times \{1\}) \cup \{acc\}$. The states marked by in serve as states that stand just before the top of the stack, the states marked by on serve as states that read the top-of-store symbol. In order to define the transition function δ' we define a function $f : V \times \Gamma \times V \to \Gamma$ that tells us how to extend the pushdown store. In case that on top of the pushdown store stands the pair (v, γ) and we want to emulate the extension of the stack with v' then we add to the pushdown store the pair $(v', f(v, \gamma, v'))$. Let $f(v, \gamma, v') = \gamma' \subseteq Q \times \{0,1\} \times (Q \cup \{acc\})$ such that $(t, i, t') \in \gamma'$ if there exists a sequence $r' = (j_1, s_1), (j_2, s_2), \ldots$ such that all the following hold.

- $(s_1, md) \in \delta(t, v')$ and $j_1 = 1$ iff $s_1 \in \alpha$.
- For every $1 \leq m < |r'|$ either $(s_{m+1}, sp) \in \delta(s_m, v)$ and $j_{m+1} = 1$ iff $s_{m+1} \in \alpha$ or $(s_m, j, s_{m+1}) \in \gamma$ and $j_{m+1} = j$.
- If r' is finite then either $(t', mu) \in \delta(s_{|r'|}, v)$ or $s_{|r'|} = acc$ and $t' = acc$.
- If r' is infinite then there are infinitely many m such that $j_m = 1$ and $t' = acc$.
- If there is some m such that $j_m = 1$ then $i = 1$. Otherwise $i = 0$.

The transition function $\delta' : Q' \times V \times \Gamma \to 2^{Q' \times \{sp, pop, push(z) \,:\, z \in V \times \Gamma\}}$ is defined as follows. For every $v \in V$ and $\gamma \in \Gamma$ we have $\delta'(acc, (v, \gamma)) = \{(acc, sp)\}$. For $(on, q, i) \in Q'$, $v \in V$ and $\gamma \in \Gamma$ we have

- If $(q', sp) \in \delta(q, \top)$ then $((on, q', i'), sp) \in \delta'((on, q, i), (v, \gamma))$ and $i' = 1$ iff $q' \in \alpha$.
- If $(q', pop) \in \delta(q, \top)$ then $((on, q', i'), pop) \in \delta'((on, q, i), (v, \gamma))$ and $i' = 1$ iff $q' \in \alpha$.
- If $(q', push(v')) \in \delta(q, \top)$ then $((on, q', i'), push(v', f(v, \gamma, v'))) \in \delta'((on, q, i), (v, \gamma))$ and $i' = 1$ iff $q' \in \alpha$.
- If $(q', md) \in \delta(q, \top)$ then $((in, q', i'), sp) \in \delta((on, q, i), (v, \gamma))$ and $i' = 1$ iff $q' \in \alpha$.

For $(in, q, i) \in Q'$, $v \in V$, and $\gamma \in \Gamma$ we have

- If $(q', sp) \in \delta(q, v)$ then $((in, q', i'), sp) \in \delta'((in, q, i), (v, \gamma))$ and $i' = 1$ iff $q' \in \alpha$.
- If $(q', mu) \in \delta(q, v)$ then $((on, q', i'), sp) \in \delta'((in, q, i), (v, \gamma))$ and $i' = 1$ iff $q' \in \alpha$.
- If $(q, i', q') \in \gamma$ then
  - if $q' = acc$ then $(acc, sp) \in \delta'((in, q, i), (v, \gamma))$.
  - if $q' \neq acc$ then $((in, q', i'), sp) \in \delta'((in, q, i), (v, \gamma))$.

Prior to showing that S is empty iff P is empty we show that whenever the contents of the pushdown store is $w \in (V \times \Gamma)^*$ with the letter (v, γ) on top of the pushdown store (the first letter of w) and $(q, i, q') \in \gamma$ then we can find a partial run that connects the configuration (q, w') to configuration (q', w') where w' is the projection of w on its component in V (with $\perp \uparrow$ concatenated on top). Formally, we have the following.

We extend the function f to a function $f : STORES \to (V \times \Gamma)^*$. For every $w = w_0, \ldots, w_m \in STORES$ we set $f(w)$ as follows.

- If $m = 3$ then $f(\top \uparrow \perp) = f(\uparrow \top \perp) = (\perp, \emptyset)$.
- If $m > 3$ where $w_i = \uparrow$ then $f(w) = (w_1, \gamma_1), \ldots, (w_{i-1}, \gamma_{i-1}), (w_{i+1}, \gamma_{i+1}), \ldots, (w_m, \gamma_m)$ where $\gamma_m = \emptyset$, $\gamma_{i-1} = f(w_{j+1}, \gamma_{j+1}, w_{j-1})$, and for every $j \neq i$ we have $\gamma_{j-1} = f(w_j, \gamma_j, w_{j-1})$. Note that the top-of-store symbol $\perp$ and the head indicator $\uparrow$ are removed.

For every $w \in STORES$ where $w = w' \uparrow w''$ we say that the *head location* in w is $|w'|$, denoted $l(w)$. A sequence $(q_0, w_0), (q_1, w_1), \ldots$ is a *partial run* of S if for every i we have $(q_{i+1}, act) \in \delta(q_i, \mathcal{H}(w_i))$ and $w_{i+1} = \mathcal{B}(w_i, act)$. We say that pair $(i, q') \in \{0,1\} \times (Q \cup \{acc\})$ is *reachable* from configuration (q, w) if there exists a partial run $r = (q_0, w_0), (q_1, w_1), \ldots$ such that $(q_0, w_0) = (q, w)$ and all the following holds.

- For all m we have $f(w_m) = f(w)$ (this means that w and w_m are the same but the location of $\uparrow$).
- If the run is finite then $q_{|r|} = q'$, and $i = 1$ iff there exists some m such that $q_m \in \alpha$.
- If the run is infinite then it visits α infinitely often, and $q' = acc$.

We say that (i, q') is *l-reachable* from (q, w) if in addition all the following hold.

- $l(w_0) = l$
- For every $1 \leq m < |r|$ we have $l(w_i) > l$.
- If the run is finite then $l(w_{|r|}) = l$.

Claim 5.1 *For every configuration (q, w) and pair $(i, q') \in \{0,1\} \times (Q \cup \{acc\})$ such that $f(w) = (w_1, \gamma_1), \ldots, (w_m, \gamma_m)$ we have (i, q') is $l(w)$-reachable from (q, w) iff $(q, i, q') \in \gamma_{l(w)}$.*

Proof: We prove the claim by induction on $m - l(w)$. If $m = l(w)$ then it must be the case that $(q', sp) \in \delta(q, \mathcal{H}(w))$. Suppose that $m > l(w)$. Let $r = (q_0, w_0), (q_1, w_1), \ldots$ denote the partial run. Let $r = (q_{j_0}, w_{j_0}), (q_{j_1}, w_{j_1}), \ldots$ denote the subsequence of locations where $l(w_{j_k}) = l(w) + 1$. For every k either $j_{k+1} = j_k + 1$ and $(q_{j_{k+1}}, sp) \in \delta(q_{j_k}, \mathcal{H}(w_{j_k}))$ or $j_{k+1} > j_k + 1$ and by induction $(q_{j_k}, i, q_{j_{k+1}}) \in \gamma_{l(w_{j_k})}$. We conclude that $(q, i, q') \in \gamma_{l(w)}$.

In the other direction we prove the claim by induction on $m - l(w)$. If $m = l(w)$ then we know that $\gamma_m = \emptyset$ and the claim follows. If $m > l(w)$ then we concatenate the partial runs promised by induction on $l(w) + 1$ to create a partial run connecting (q, w) and (i, q'). □

We are now ready to prove the following claim.

Claim 5.2 *$\mathcal{L}(S) = \emptyset$ iff $\mathcal{L}(P) = \emptyset$.*

Proof: Suppose that $\mathcal{L}(S) \neq \emptyset$. Then there exists an accepting run $r = (q_0, w_0), (q_1, w_1), \ldots$ on the word a^ω. We show that the subsequence of configurations where S stands on top of the store or just below the top of the store is an accepting run of P on a^ω. If the subsequence is finite then the run of P ends in an infinite chain of *acc*. More formally, we have the following.

Given a run $r = (q_0, w_0), (q_1, w_1), \ldots$ let $r' = (q_{i_0}, w_{i_0}), (q_{i_1}, w_{i_1}), \ldots$ denote the subsequence of configurations where S stands on top of the stack or one location before the end of the stack (i.e. $w_{i_j} = \uparrow \top w \bot$ or $w_{i_j} = \top \uparrow w \bot$ for some $w \in V^*$). We translate this subsequence into a run of P. Set $l_{i_j} = in$ if $w_{i_j} = \top \uparrow w \bot$ for some $w \in V^*$ and $l_{i_j} = on$ otherwise. Set $\alpha_{i_0} = 0$. Set $\alpha_{i_j} = 1$ if there exists k such that $i_{j-1} < k \leq i_j$ and $q_k \in \alpha$. Otherwise, set $\alpha_{i_j} = 0$. We claim that the sequence $r' = \langle (l_{i_0}, q_{i_0}, \alpha_{i_0}), f(w_{i_0}) \rangle, \langle (l_{i_1}, q_{i_1}, \alpha_{i_1}), f(w_{i_1}) \rangle, \ldots$ is a valid and accepting run of P on a^ω.

We first show that the transition $(\langle (l_{i_j}, q_{i_j}, \alpha_{i_j}), f(w_{i_j}) \rangle, \langle (l_{i_{j+1}}, q_{i_{j+1}}, \alpha_{i_{j+1}}), f(w_{i_{j+1}}) \rangle)$ is a legal transition of P. Clearly, if $i_{j+1} = i_j + 1$ then it is a legal transition. Suppose that $i_{j+1} > i_j + 1$. It follows from Claim 5.1 that it is a legal transition. We show that the run is accepting. If the run ends in an infinite sequence of *acc* then it is clearly accepting. Otherwise, α_{i_j} is set to 1 iff some state in $q_{i_j+1}, \ldots, q_{i_{j+1}}$ is accepting. As r is accepting so is r'.

In the other direction we extend the partial run of P to a full run of S using the promised partial runs from Claim 5.1. □

Theorem 5.3 [24, 32] *The emptiness problem of a PD-NBW can be determined in time $O(|Q|^2 \cdot |\delta| \cdot |V|)$.*

Theorem 5.4 *The emptiness problem of a ST-NBW can be determined in time $O(|Q|^2 \cdot |\delta| \cdot |V|) \cdot 2^{O(|Q|^2)}$.*

Corollary 5.5 *Given an LTL formula φ and a micro-macro stack system R the model checking problem is solvable in time $2^{O(|R|^2) \cdot 2^{O(|\varphi|)}}$.*

We advance now to the general case of ST-APW$_1$. First let us denote by *top-configurations* configurations in which the head indicator $\uparrow$ is either to the left or to the right of the top-of-store symbol $\top$ (i.e. configurations (q, w) where $w = \top \uparrow w' \bot$ or $w = \uparrow \top w' \bot$ for some $w' \in V^*$). Otherwise the configuration is denoted a *middle-configuration*. A nondeterministic word automaton has a single copy reading the input word at all times. Hence, when a nondeterministic automaton 'ventures' into the stack it comes out in a single copy (if at all). When a copy of the alternating automaton reads the last meaningful letter in the store, the run continues to form a finite tree whose internal nodes are all middle-configurations and whose leaves are top-configurations. On the pushdown store, we record for each state the possible sets of leaves of such trees (corresponding to sp and mu actions). The trees may be infinite if some path stays indefinitely inside the store. However we care only about the possible sets of leaves.

Consider an ST-APW$_1$ $S = \langle \{a\}, V, Q, \delta, q_0, \top, \bot, \alpha \rangle$ where $\alpha = \langle F_1, \ldots, F_k \rangle$ and $\delta : Q \times \{a\} \times V \to B^+(Q \times ACT)$. For a state $q \in Q$ let $r(q)$ denote the index i such that $q \in F_i$. We construct a PD-APW$_1$ P such that $\mathcal{L}(P) = \emptyset$ iff $\mathcal{L}(S) = \emptyset$. The pushdown alphabet of P is $V \times \Gamma$ where $\Gamma = 2^{Q \times 2^{([k] \times Q \times \{sp, mu\})}}$. Thus, every letter in Γ associates with every state in Q subsets of $[k] \times Q \times \{sp, mu\}$. For a state $q \in Q$ and $\gamma \in \Gamma$, the sets U such that $(q, U) \in \gamma$ are the possible transitions of state q. Again the automaton can enter the stack and stay there indefinitely. We just make sure that the set of states U participates in **some** run such that the paths that stay indefinitely in the stack are accepting. The minimal parity set visited is monitored via the labels attached to the states in Q. Label r indicates that the least rank visited along the detour is r. More formally, we have the following.

For a set $U \subseteq Q \times \{sp, mu\}$ let $r(U)$ denote the set $\{(r(s), s, \Delta) \mid (s, \Delta) \in U\}$. Let $\flat \subseteq Q \times 2^{[k]\times Q\times\{sp,mu\}}$ denote the set $\{(q, r(U)) \mid U \models \delta(q, \bot)\}$ of assignments that satisfy the transition of states in Q reading the bottom-of-store symbol $\bot$. The symbol $\flat$ is added to the store-bottom-symbol $\bot$ as the store-bottom-symbol of the pushdown automaton. Let $P = \langle \{a\}, V \times \Gamma, Q', \delta', q'_0, (\bot, \flat), \alpha' \rangle$ where the set of states is $Q' = \{in, on\} \times Q \times [k]$, the initial state is $q'_0 = (in, q_0, r(q_0))$, and the parity acceptance condition is $\alpha' = \langle \{in, on\} \times Q \times \{1\}, \ldots, \{in, on\} \times Q \times \{k\} \rangle$. The in and on states are just as in nondeterministic automata. The definition of the transition function and the extension of the pushdown store is quite technical. Details follow.

In order to define the transition function δ' we define a function $f : V \times \Gamma \times V \to \Gamma$ that tells us how to extend the pushdown store. In case that on top of the pushdown store stands the pair (v, γ) and we want to emulate the extension of the stack with v' then we add to the pushdown store the pair $(v', f(v, \gamma, v'))$. For every state $s \in Q$, in order to test whether to add the pair (s, U) to γ' where $U \subseteq [k] \times Q \times \{sp, mu\}$ we construct an APW that moves between the letters v and v'. The transition of states in U that read v' is set to $\mathbf{true}$ and states not in T that read v' are set to **false**. The pair (s, U) is added if the APW has an accepting run[4]. Formally, for every state $t \in Q$ and every subset $U \subseteq [k] \times Q \times \{sp, mu\}$ we define the APW $A^{t,U}_{v,\gamma,v'} = \langle \{a\}, Q'', \rho, t, \alpha'' \rangle$ where the set of states is $Q'' = \{t\} \cup (\{v\} \times Q \times [k]) \cup (\{v'\} \times Q \times [k] \times \{sp, mu\})$, the parity acceptance condition is $\alpha'' = \langle \{v\} \times Q \times \{1\}, \ldots, \{v\} \times Q \times \{k\} \rangle$, and the transition function ρ is as follows[5].

- $\rho(t, a)$ is obtained from $\delta(t, v')$ by replacing (s, mu) by $(v', s, r(s), mu)$, replacing (s, sp) by $(v', s, r(S), sp)$, and replacing (s, md) by $(v, s, r(s))$.
- $\rho(v', s, r, \Delta) = \begin{cases} \mathbf{true} & \text{if } (r, s, \Delta) \in U \\ \mathbf{false} & \text{if } (r, s, \Delta) \notin U \end{cases}$
- $\rho(v, s, r) = \bigvee_{(s,U')\in\gamma} \bigwedge_{(r',s',\Delta)\in U'} c_r(r', s', \Delta)$ where $c_r : [k] \times Q \times \{sp, mu\} \to Q''$ is
$$c_r(r', s', \Delta) = \begin{cases} (v', min(r, r'), s', sp) & \text{if } \Delta = mu \\ (v, min(r, r'), s') & \text{if } \Delta = sp \end{cases}$$

Finally, $f(v, \gamma, v') = \gamma'$ such that for every state $s \in Q$ and set $U \subseteq [k] \times Q \times \{sp, mu\}$ we have $(s, U) \in \gamma'$ iff $A^{s,U}_{v,\gamma,v'}$ has some accepting run [6].

The transition function $\delta' : Q' \times V \times \Gamma \to B^+(Q' \times \{sp, pop, push(z) : z \in V \times \Gamma\})$ is defined as follows. For every $v \in V$, and $\gamma \in \Gamma$ we have

- $\delta'((on, q, i), (v, \gamma))$ is obtained from $\delta(q, v)$ by replacing (s, sp) by $((on, s, r(s)), sp)$, replacing $(s, push(v'))$ by $((on, s, r(s)), push(v', f(v, \gamma, v')))$, replacing (s, pop) by $((on, s, r(s)), pop)$, and replacing (s, md) by $((in, s, r(s)), sp)$.
- $\delta'((in, q, i), (v, \gamma)) = \bigvee_{(q,U)\in\gamma} \bigwedge_{(r,s,\Delta)\in U} c(r, s, \Delta)$ where $c : [k] \times Q \times \{sp, mu\} \to Q'$ is as follows.
$$c(r, s, \Delta) = \begin{cases} (in, s, r) & \Delta = sp \\ (on, s, r) & \Delta = mu \end{cases}$$

Prior to showing that S is empty iff P is empty we show that whenever the contents of the pushdown store is $w \in (V \times \Gamma)^*$ with the letter (v, γ) on top of the pushdown store (the first letter of w) and $(q, U) \in \gamma$ then we can find a run tree whose root is labeled by (q, w') and all its leaves are labeled by configurations $(s, \mathcal{B}(w', act))$ for $(r, s, act) \in U$ where w' is the projection of w on its component in V (with $\bot \uparrow$ concatenated on top) and all infinite paths in the tree are accepting according to α. Formally, we have the following.

We extend the function f to a function $f : STORES \to (V \times \Gamma)^*$ just like we did for ST-NBW. and we use the head location as defined for ST-NBW. A $Q \times STORES$-labeled tree $\langle T, r \rangle$ is a *partial run* of S if for every node $x \in T$ with $r(x) = (q, w)$ and $\delta(q, \mathcal{H}(w)) = \theta$ there exists a (possibly empty) set $\{(q_1, act_1), \ldots, (q_d, act_d)\} \models \theta$ and x has d successors $x_1, \ldots, x_d$ such that $r(x_j) = (q_j, \mathcal{B}(w, act_j))$. We say that a set $U \subseteq [k] \times Q \times \{sp, mu\}$ is *l-reachable* from configuration (q, w) if there exists a partial run $\langle T, r \rangle$ such that all the following hold.

- The root of T is labeled by $r(\epsilon) = (q, w)$.
- For every node $x \in T$ such that $r(x) = (s, w')$ then $f(w) = f(w')$ (this means that w and w' are the same but for the location of the head indicator $\uparrow$).
- For every node $x \in T$ such that $r(x) = (s, w')$ one of the following holds.
    - x is a leaf or $x = \epsilon$ and $l(w') = l(w)$ or $l(w') = l(w) - 1$.

[4]This construction resembles the transformation of APW with ϵ-moves to APW without ϵ-moves in [49].

[5]Note that the parity acceptance condition does not include states in $t \cup (\{v'\} \times Q \times [k] \times \{sp, mu\})$. These states cannot occur more than once on a path and adding / removing them from α'' does not matter.

[6]Notice that once a pair (s, U) is added to γ' then for every superset U' such that $U \subseteq U'$ we have $(s, U') \in \gamma'$. In practice, it is sufficient to add (s, U), however this would complicate the proof beyond necessary.

  - x is internal and $x \neq \epsilon$ and $l(w') > l(w)$.

- For every leaf $x \in T$ such that $r(x) = (s, w')$ and the minimal rank on the path from the root to this leaf is r we have $(s, r, act) \in U$ and $w' = \mathcal{B}(w, act)$.

- Every infinite path in T is accepting according to α.

Claim 5.6 *For every configuration (q, w) and set $U \subseteq [k] \times Q \times \{sp, mu\}$ we have U is $l(w)$-reachable from (q, w) iff $(q, U) \in \gamma_{l(w)}$.*

Proof: Consider a configuration (q, w) and a set $U \subseteq [k] \times Q \times \{sp, mu\}$. Assume that U is $l(w)$-reachable from (q, w). There exists a partial run $\langle T, r\rangle$ such that all the leaves of T are labeled by states in U. We show by induction on $m - l(w)$ that $(q, U) \in \gamma_{l(w)}$. If $m = l(w)$ then the transition of (q, w) has to be supplied by states labeled by sp and mu. In this case, for the set U of q's successors we have $(q, U) \in \gamma_m = \flat$. Suppose $m > l(w)$. We define an equivalence relation on the nodes of T. Each equivalence class corresponds to the partial run that shows that some set U' is $l(w) + 1$-reachable from some configuration (q', w'). Then we use the induction assumption to show that $(q', U') \in \gamma_{l(w)+1}$ and prove that $(q, U) \in \gamma_{l(w)}$. Formally, we have the following.

Let $f(w) = (w_1, \gamma_1), \ldots, (w_m, \gamma_m)$. We abuse notation and for a node x such that $r(x) = (q', w')$ we write $l(x)$ for $l(w')$. Recall that every node x such that $l(x) \leq l(w)$ is either the root of T or a leaf. For every node x we add an annotation to x another node in T. If $l(x) \leq l(w) + 1$ we annotate x by itself. If $l(x) > l(w) + 1$ then we annotate x by the least node x' such that $l(x') = l(w) + 1$ and there exists no $x' < x'' < x$ such that $l(x'') \leq l(w) + 1$. We say that two nodes x and x' are equivalent if the annotation of x and x' is equal.

For a node x such that $l(x) = l(w) + 1$ consider the tree T_x consisting of all the nodes in the equivalence class of x and their immediate descendants. That is, T_x includes internal nodes with head location greater than $l(w) + 1$ and leaves with head location at most $l(w) + 1$. Let $r(x) = (q', w')$ and let $U \subseteq [k] \times Q \times \{sp, mu\}$ be a set such that for every leaf $y \in T_x$ such that $r(y) = (q'', w'')$, r is the minimal rank on the path from the root to y, and $w'' = \mathcal{B}(w', act)$ then $(q'', r, act) \in U$. If y is a leaf in T_x then $l(y) = l(w) + 1$ or $l(y) = l(w)$. If follows that T_x is a partial run connecting (q', w') to U manifesting the fact that U is $l(w')$-reachable from (q', w'). As $l(w') = l(w) + 1$ we conclude that $(q', U) \in \gamma_{l(w)+1}$.

From above it is obvious that with every node x such that $l(x) = l(w) + 1$ we can associate a set $U_x \subseteq [k] \times Q \times \{sp, mu\}$ that 'labels' all the leaves of T_x. For every triplet (r, s, act) in U_x we choose one leaf $y^x_{r,s,act}$ in T_x that is 'labeled' by this triplet. Let T' be the minimal tree such that $\epsilon \in T'$ and for every node $x \in T'$ and every triplet $(r, s, act) \in U_x$ the leaf $y^x_{r,s,act} \in T'$. Let $U \subseteq [k] \times Q \times \{sp, mu\}$ denote the set of labels of leaves in T with the minimal ranks from the root to them. We claim that $\langle T', r'\rangle$ where r' is the restriction of r to T' is a run of $A^{q,U}_{w_{l(w)},\gamma_{l(w)},w_{l(w)+1}}$. From the explanation above it is clear that it is a valid run of $A^{q,U}_{w_{l(w)},\gamma_{l(w)},w_{l(w)+1}}$. We show that it is accepting. Every infinite path in T' visits infinitely many nodes x such that $l(x) = l(w) + 1$. However, an infinite path in T' corresponds to an infinite path in T. As the path in T is accepting we conclude that the path in T' is also accepting.

In the other direction, assume that $(q, U) \in \gamma_{l(w)}$. We prove by induction on $m - l(w)$ that U is $l(w)$-reachable from (q, w). For $m = l(w)$, we know that $\gamma_m = \flat$ and every pair $(t, U) \in \flat$ corresponds to a partial run that shows m-reachability of U from (q, w). Suppose $m > l(w)$. We use the induction assumption to replace every transition of $A^{q,U}_{w_{l(w)+1},\gamma_{l(w)+1},w_{l(w)}}$ by the partial run that is promised from the membership of (q, U) in $\gamma_{l(w)+1}$. This is clearly a legal partial run that connects (q, w) to U. We have to show that the run is accepting. An infinite path that remains from some point onwards inside some partial run is definitely accepting. An infinite path that is the result of the concatenation of infinitely many partial runs is also accepting because every node is marked by the minimal rank between the root and the leaf. ☐

We are now ready to prove the following claim.

Claim 5.7 *$\mathcal{L}(S) = \emptyset$ iff $\mathcal{L}(P) = \emptyset$.*

Proof: From the previous claim it is clear that we can convert a run of S on a^ω to a legal run of P. Showing that this run is also accepting is not different from the arguments used in Claim 5.6.

The other direction is also similar. By popping the partial runs promised by Claim 5.6 we convert a run of P to a valid and accepting run of S. ☐

Theorem 5.8 [33] *The emptiness problem of a PD-APW$_1$ $P = \langle \{a\}, V, Q, \delta, q_0, \top, \bot, \alpha\rangle$ with n states and index k can be determined in time $(|V|nk)^{O((nk)^2)}$.*

Theorem 5.9 *The emptinesss problem of a ST-APW$_1$ with n states and index k can be determined in time $2^{(nk)^2 \cdot 2^{O(n^2k)}}$.*

Corollary 5.10 *The model checking problem of a graph automaton A and a micro-macro stack system R can be determined in time $2^{2^{O((|A||R|)^2)}}$.*

6 Conclusions and Future Work

We propose a class of graphs called micro-macro stack graphs that strictly contains the class of prefix-recognizable graphs. We give direct automata-theoretic algorithms for model checking μ-calculus over micro-macro stack graphs. Our model checking algorithms is double exponential.

Since their introduction in [15], prefix-recognizable graphs have been thoroughly studied. As a few examples we mention, games on prefix-recognizable graphs [14], characterization of languages accepted by prefix-recognizable graphs [41], and prefix-recognizable structures [3]. There are many equivalent ways to represent prefix-recognizable graphs, using rewrite rules, as the outcome of regular restriction and inverse regular substitution on the infinite binary tree [15], as monadic second order logic interpretations in the infinite binary tree [3], and as graph equations [15, 1]. All these issues need to be studied for mMs graphs.

As mentioned, the class of micro-macro stack graphs is contained in the class of high order pushdown graphs. As the monadic second-order theory of the latter is decidable [31], it follows that the monadic second-order theory of micro-macro stack graphs is decidable.

7 Acknowledements

We thank T. Cachat for bringing [31] to our attention and for clarifying the connection between mMs systems and high order pushdown systems.

References

[1] K. Barthelmann. On equational simple graphs. Tech. report, Universität Mainz, 1997.

[2] P. Biesse, T. Leonard, and A. Mokkedem. Finding bugs in an alpha microprocessors using satisfiability solvers. In *13th CAV*, LNCS 2102, 454–464. Springer-Verlag. 2001.

[3] A. Blumensath. Prefix-recognisable graphs and monadic second-order logic. Tech. Report, RWTH Aachen, 2001.

[4] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *8th Concur*, LNCS 1243, 135–150, Springer-Velag, 1997.

[5] J. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Internat. Congr. Logic, Method. and Philos. Sci. 1960*, pages 1–12, Stanford, 1962.

[6] J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundladen der Mathematik*, 6:66–92, 1960.

[7] O. Burkart. Automatic verification of sequential infinite-state processes. In *LNCS 1354*, Springer-Verlag. 1997.

[8] O. Burkart. Model checking rationally restricted right closures of recognizable graphs. In *Proc. 2nd Infinity*, 1997.

[9] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. Unpublished manuscript, 2000.

[10] O. Burkart and J. Esparza. More infinite results. In *1st Infinity*, *ENTCS*, 6. Elsevier, 1996.

[11] O. Burkart and Y.-M. Quemener. Model checking of infinite graphs defined by graph grammers. In *1st Infinity*, 1996.

[12] O. Burkart and B. Steffen. Composition, decomposition and model checking of pushdown processes. *Nordic J. Comut.*, 2:89–125, 1995.

[13] O. Burkart and B. Steffen. Model checking the full modal μ-calculus for infinite sequential processes. *TCS*, 221. 1999.

[14] T. Cachat. Uniform solution of parity games on prefix-recognizable graphs. In *4th Infinity*, 2002.

[15] D. Caucal. On infinite transition graphs having a decidable monadic theory. In *23rd ICALP*, LNCS 1099. 1996.

[16] D. Caucal. On infinite terms having a decidable monadic theory. In *27th MFCS*, LNCS 2420, 165–176. 2002.

[17] D. Caucal and R. Monfort. On the transition graphs of automata and grammars. In *16th WG*, LNCS 484. 1990.

[18] E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logic of Programs*, LNCS 131, 1981.

[19] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM TOPLAS*, 8(2):244–263, January 1986.

[20] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[21] F. Copty, L. Fix, R. Fraer, E. Giunchiglia, G. Kamhi, A. Tacchella, and M. Vardi. Benefits of bounded model checking at an industrial setting. In *13th CAV*, LNCS 2102, 2001.

[22] M. Dickhfer and T. Wilke. Timed alternating tree automata: the automata-theoretic solution to the TCTL model checking problem. In *ICALP*, LNCS 1644, 281–290. 1999.

[23] E. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd FOCS*, 368–377, 1991.

[24] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *12th CAV*, LNCS 1855, 232–247, 2000. Springer-Verlag.

[25] A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown automata. In *2nd Infinity*, 1997.

[26] S. Ginsburg, S. Greibach, and M. Harrison. One-way stack automata. *JACM*, 14(2):381–418, 1967.

[27] S. Ginsburg, S. Greibach, and M. Harrison. Stack automata and compiling. *JACM*, 14(1):172–201, 1967.

[28] D. Harel and D. Raz. Deciding emptiness for stack automata on infinite trees. *IC*, 113(2):278–299, 1994.

[29] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[30] D. Janin and I. Walukiewicz. Automata for the modal μ-calculus and related results. In *20th MFCS*, 1995.

[31] T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-order pushdown trees are easy. In *5th FOSSACS*, LNCS 2303, 205–222, 2003. Springer-Verlag.

[32] O. Kupferman, N. Piterman, and M. Vardi. Model checking linear properties of prefix-recognizable systems. In *14th CAV*, LNCS 2404, 371–386. Springer-Verlag. 2002.

[33] O. Kupferman, N. Piterman, and M. Vardi. Pushdown specifications. In *9th LPAR*, LNCS 2514, 262–277. 2002.

[34] O. Kupferman and M. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *12th CAV*, LNCS 1855, 36–52. Springer-Verlag. 2000.

[35] O. Kupferman, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *JACM*, 47(2):312–360, 2000.

[36] R. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.

[37] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *12th POPL*, 97–107, 1985.

[38] C. Morvan. On rational graphs. In *3rd FSSCS*, LNCS 1784, 252–266, Springer-Verlag. 2000.

[39] D. Muller and P. Schupp. The theory of ends, pushdown automata, and second-order logic. *TCS*, 37:51–75, 1985.

[40] J. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Proc. 5th International Symp. on Programming*, LNCS 137, 337–351. 1981.

[41] C. Stirgling. Decidability of bisimulation equivalence for pushdown processes. Unpublished manuscript, 2000.

[42] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 165–191, 1990.

[43] W. Thomas. A short introduction to infinite automata. In *5th. DLT*, LNCS 2295, 130–144. Springer-Verlag, 2001.

[44] M. Vardi. Reasoning about the past with two-way automata. In *25th ICALP*, LNCS 1443, 628–641. 1998.

[45] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *1st LICS*, 332–344, 1986.

[46] M. Vardi and P. Wolper. Reasoning about infinite computations. *IC*, 115(1):1–37, 1994.

[47] I. Walukiewicz. Pushdown processes: games and model checking. In *8th CAV*, LNCS 1102, 62–74. 1996.

[48] I. Walukiewicz. Monadic second-order logic on tree-like structures. *TCS*, 275(1-2):311–346, 2002.

[49] T. Wilke. CTL$^+$ is exponentially more succinct than CTL. In *19th FSTTCS*, LNCS 1738, 110–121. 1999.