From Multi to Single Stack Automata

Mohamed Faouzi Atig

LIAFA, CNRS & Univ. of Paris 7, Case 7014, 75205 Paris 13, France
atig@liafa.jussieu.fr

Abstract. We investigate the issue of reducing the verification problem of multi-stack machines to the one for single-stack machines. For instance, elegant (and practically efficient) algorithms for bounded-context switch analysis of multi-pushdown systems have been recently defined based on reductions to the reachability problem of (single-stack) pushdown systems [10,18]. In this paper, we extend this view to both bounded-phase visibly pushdown automata (BVMPA) [16] and ordered multi-pushdown automata (OMPA) [1] by showing that each of their emptiness problem can be reduced to the one for a class of single-stack machines. For these reductions, we introduce *effective generalized pushdown automata* (EGPA) where operations on stacks are (1) pop the top symbol of the stack, and (2) push a word in some (effectively) given set of words L over the stack alphabet, assuming that L is in some class of languages for which checking whether L intersects regular languages is decidable. We show that the automata-based saturation procedure for computing the set of predecessors in standard pushdown automata can be extended to prove that for EGPA too the set of all predecessors of a regular set of configurations is an effectively constructible regular set. Our reductions from OMPA and BVMPA to EGPA, together with the reachability analysis procedure for EGPA, allow to provide conceptually simple algorithms for checking the emptiness problem for each of these models, and to significantly simplify the proofs for their 2ETIME upper bounds (matching their lower-bounds).

1 Introduction

In the last few years, a lot of effort has been devoted to the verification problem for models of concurrent programs (see, e.g., [5,3,16,9,2]). Pushdown automata have been proposed as an adequate formalism to describe sequential programs with procedure calls [8,14]. Therefore, it is natural to model recursive concurrent programs as multi-stack automata. In general, multi-stack automata are Turing powerful and hence come along with undecidability of basic decision problems [13]. To overcome this barrier, several subclasses of pushdown automata with multiple stacks have been proposed and studied in the literature.

Context-bounding has been proposed in [12] as a suitable technique for the analysis of multi-stack automata. The idea is to consider only runs of the automaton that can be divided into a given number of contexts, where in each context pop and push operations are exclusive to one stack. Although the state space which may be explored is still unbounded in presence of recursive procedure calls, the context-bounded reachability

P. Gastin and F. Laroussinie (Eds.): CONCUR 2010, LNCS 6269, pp. 117–131, 2010.

problem is NP-complete even in this case [11]. In fact, context-bounding provides a very useful tradeoff between computational complexity and verification coverage.

In [16], La Torre et al. propose a more general definition of the notion of a context. For that, they define the class of *bounded-phase visibly multi-stack pushdown automata* (BVMPA) where only those runs are taken into consideration that can be split into a given number of phases, where each phase admits pop operations of one particular stack only. In the above case, the emptiness problem is decidable in double exponential time by reducing it to the emptiness problem for tree automata.

Another way to regain decidability is to impose some order on stack operations. In [6], Breveglieri et al. define *ordered multi-pushdown automata* (OMPA), which impose a linear ordering on stacks. Stack operations are constrained in such a way that a pop operation is reserved to the first non-empty stack. In [1], we show that the emptiness problem for OMPA is 2ETIME-complete[1]. The proof of this result lies in a complex encoding of OMPA into some class of grammars for which the emptiness problem is decidable.

In this paper, we investigate the issue of reducing the verification problem of multi-stack machines to the one for single-stack machines. We believe that this is a general paradigm for understanding the expressive power and for establishing decidability results for various classes of concurrent program models. For instance, elegant (and practically efficient) algorithms for bounded-context switch analysis of multi-pushdown systems have been recently defined based on reductions to the reachability problem of (single-stack) pushdown systems [10,18]. We extend this view to both OMPA and BVMPA by showing that each of their emptiness problem can be reduced to the one for a class of single-stack machines. For these reductions, we introduce *effective generalized pushdown automata* (EGPA) where operations on stacks are (1) pop the top symbol of the stack, and (2) push a word in some (effectively) given set of words L over the stack alphabet, assuming that L is in some class of languages for which checking whether L intersects a given regular language is decidable. Observe that L can be any finite union of languages defined by a class of automata closed under intersection with regular languages and for which the emptiness problem is decidable (e.g., pushdown automata, Petri nets, lossy channel machines, etc). Then, we show that the automata-based saturation procedure for computing the set of predecessors in standard pushdown automata [4] can be extended to prove that for EGPA too the set of all predecessors of a regular set of configurations is a regular set and effectively constructible. As an immediate consequence of this result, we obtain similar decidability results of the decision problems for EGPA like the ones obtained for pushdown automata.

Then, we show that, given an OMPA $\mathcal{M}$ with n stacks, it is possible to construct an EGPA $\mathcal{P}$, whose pushed languages are defined by OMPA with $(n-1)$ stacks, such that the emptiness problem for $\mathcal{M}$ is reducible to its corresponding problem for $\mathcal{P}$. The EGPA $\mathcal{P}$ is constructed such that the following invariant is preserved: The state and the content of the stack of $\mathcal{P}$ are the same as the state and the content of the n-th stack of $\mathcal{M}$ when its first $(n-1)$ stacks are empty. Then, we use the saturation procedure for

[1] Recall that 2ETIME is the class of all decision problems solvable by a deterministic Turing machine in time $2^{2^{dn}}$ for some constant d.

EGPA to show, by induction on the number of stacks n, that the emptiness problem of an OMPA is in 2ETIME with respect to n (matching its lower-bound [1]).

Another application of EGPA is to show that, given a k-phase BVMPA $\mathcal{M}$ with n stacks, it is possible to construct an EGPA $\mathcal{P}$, whose pushed languages are defined by $(k-1)$-phase BVMPA with n stacks, such that the emptiness problem of $\mathcal{M}$ can be reduced to compute the set of predecessors of a regular set of configurations of $\mathcal{P}$. Then, we exploit the saturation procedure for EGPA to show, by induction on the number of phases k, that the emptiness problem for a BVMPA is in 2ETIME with respect to k (matching its lower-bound [17]).

Related works: To the best of our knowledge, the class of effective generalized pushdown automata that we define in this paper is the first non-trivial extension of pushdown automata that allows to push a non-regular language (e.g., Petri nets languages) into the stack which is not the case of prefix-recognizable graphs [7]. Moreover, our reductions from OMPA and BVMPA to EGPA, together with the reachability analysis procedure for EGPA, provide conceptually simple algorithms for checking the emptiness problem for each of these models, and proving their 2ETIME upper bounds.

2 Preliminaries

In this section, we introduce some basic definitions and notations that will be used in the rest of the paper.

Integers: Let $\mathbb{N}$ be the set of natural numbers. For every $i, j \in \mathbb{N}$ such that $i \leq j$, we use $[i, j]$ (resp. $[i, j[$) to denote the set $\{k \in \mathbb{N} \mid i \leq k \leq j\}$ (resp. $\{k \in \mathbb{N} \mid i \leq k < j\}$).

Words and languages: Let Σ be a finite alphabet. We denote by Σ^* (resp. Σ^+) the set of all words (resp. non empty words) over Σ, and by ϵ the empty word. A language is a (possibly infinite) set of words. We use Σ_ϵ and $Lang(\Sigma)$ to denote respectively the set $\Sigma \cup \{\epsilon\}$ and the set of all languages over Σ. Let u be a word over Σ. The length of u is denoted by $|u|$. For every $j \in [1, |u|]$, we use $u(j)$ to denote the j^{th} letter of u. We denote by u^R the mirror of u.

Let Θ be a subset of Σ. Given a word $v \in \Sigma^*$, we denote by $v|_\Theta$ the projection of v over Θ, i.e., the word obtained from v by erasing all the symbols that are not in Θ. This definition is extended to languages as follows: If L is a language over Σ, then $L|_\Theta = \{v|_\Theta \mid v \in L\}$.

Transition systems: A transition system is a triplet $\mathcal{T} = (C, \Sigma, \rightarrow)$ where: (1) C is a (possibly infinite) set of configurations, (2) Σ is a finite set of labels (or actions) such that $C \cap \Sigma = \emptyset$, and (3) $\rightarrow \subseteq C \times \Sigma_\epsilon \times C$ is a transition relation. We write $c \xrightarrow{a}_{\mathcal{T}} c'$ whenever c and c' are two configurations and a is an action such that $(c, a, c') \in \rightarrow$.

Given two configurations $c, c' \in C$, a finite run ρ of $\mathcal{T}$ from c to c' is a finite sequence $c_0 a_1 c_1 \cdots a_n c_n$, for some $n \geq 1$, such that: (1) $c_0 = c$ and $c_n = c'$, and (2) $c_i \xrightarrow{a_{i+1}}_{\mathcal{T}} c_{i+1}$ for all $i \in [0, n[$. In this case, we say that ρ has length n and is labelled by the word $a_1 a_2 \cdots a_n$.

Let $c, c' \in C$ and $u \in \Sigma^*$. We write $c \xRightarrow[n]{u}_{\mathcal{T}} c'$ if one of the following two cases holds: (1) $n = 0$, $c = c'$, and $u = \epsilon$, and (2) there is a run ρ of length n from c to

c' labelled by u. We also write $c \overset{u}{\Longrightarrow}^*_{\mathcal{T}} c'$ (resp. $c \overset{u}{\Longrightarrow}^+_{\mathcal{T}} c'$) to denote that $c \overset{u}{\underset{n}{\Longrightarrow}}_{\mathcal{T}} c'$ for some $n \geq 0$ (resp. $n > 0$).

For every $C_1, C_2 \subseteq C$, let $Traces_{\mathcal{T}}(C_1, C_2) = \{u \in \Sigma^* \mid \exists (c_1, c_2) \in C_1 \times C_2 \,,\, c_1 \overset{u}{\Longrightarrow}^*_{\mathcal{T}} c_2\}$ be the set of sequences of actions generated by the runs of $\mathcal{T}$ from a configuration in C_1 to a configuration in C_2, and let $Pre^*_{\mathcal{T}}(C_1) = \{c \in C \mid \exists (c', u) \in C_1 \times \Sigma^* \,,\, c \overset{u}{\Longrightarrow}^*_{\mathcal{T}} c'\}$ be the set of predecessor configurations of C_1.

Finite state automata: A finite state automaton (FSA) is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ where: (1) Q is the finite non-empty set of states, (2) Σ is the finite input alphabet, (3) $\Delta \subseteq (Q \times \Sigma_\epsilon \times Q)$ is the transition relation, (4) $I \subseteq Q$ is the set of initial states, and (5) $F \subseteq Q$ is the set of final states. We represent a transition (q, a, q') in Δ by $q \xrightarrow{a}_{\mathcal{A}} q'$. Moreover, if I' and F' are two subsets of Q, then we use $\mathcal{A}(I', F')$ to denote the finite state automaton defined by the tuple $(Q, \Sigma, \Delta, I', F')$.

The size of $\mathcal{A}$ is defined by $|\mathcal{A}| = (|Q| + |\Sigma|)$. We use $\mathcal{T}(\mathcal{A}) = (Q, \Sigma, \Delta)$ to denote the transition system associated with $\mathcal{A}$. The language accepted (or recognized) by $\mathcal{A}$ is given by $L(\mathcal{A}) = Traces_{\mathcal{T}(\mathcal{A})}(I, F)$.

3 Generalized Pushdown Automata

In this section, we introduce the class of generalized pushdown automata where operations on stacks are (1) pop the top symbol of the stack, and (2) push a word in some (effectively) given set of words L over the stack alphabet. A transition t is of the form $\delta(p, \gamma, a, p') = L$ where L is a (possibly infinite) set of words. Being in a configuration (q, w) where q is a state and w is a stack content, t can be applied if both $p = q$ and the content of the stack is of the form $\gamma w'$ for some w'. Taking the transition and reading the input letter a (which may be the empty word), the system moves to the successor configuration (p', uw') where $u \in L$ (i.e., the new state is p', and γ is replaced with a word u belonging to the language L). Formally, we have:

Definition 1 (Generalized pushdown automata). *A generalized pushdown automaton (GPA for short) is a tuple $\mathcal{P} = (P, \Sigma, \Gamma, \delta, p_0, \gamma_0, F)$ where: (1) P is the finite non-empty set of states, (2) Σ is the input alphabet, (3) Γ is the stack alphabet, (4) $\delta : P \times \Gamma \times \Sigma_\epsilon \times P \rightarrow Lang(\Gamma)$ is the transition function, (5) $p_0 \in P$ is the initial state, (6) $\gamma_0 \in \Gamma$ is the initial stack symbol, and (7) $F \subseteq P$ is the set of final states.*

Definition 2 (Effectiveness Property). *A GPA $\mathcal{P} = (P, \Sigma, \Gamma, \delta, p_0, \gamma_0, F)$ is effective if and only if for every finite state automaton $\mathcal{A}$ over the alphabet Γ, it is decidable whether $L(\mathcal{A}) \cap \delta(p, \gamma, a, p') \neq \emptyset$ for all $p, p' \in P$, $\gamma \in \Gamma$, and $a \in \Sigma_\epsilon$.*

A configuration of a GPA $\mathcal{P} = (P, \Sigma, \Gamma, \delta, p_0, \gamma_0, F)$ is a pair (p, w) where $p \in P$ and $w \in \Gamma^*$. The set of all configurations of $\mathcal{P}$ is denoted by $Conf(\mathcal{P})$. Similarly to the case of pushdown automata [4], we use the class of $\mathcal{P}$-automata as finite symbolic representation of a set of configurations of GPA. Formally, a $\mathcal{P}$-automaton is a FSA $\mathcal{A} = (Q_{\mathcal{A}}, \Gamma, \Delta_{\mathcal{A}}, I_{\mathcal{A}}, F_{\mathcal{A}})$ such that $I_{\mathcal{A}} = P$. We say that a configuration (p, w) of $\mathcal{P}$ is accepted (or recognized) by $\mathcal{A}$ if $w \in L(\mathcal{A}(\{p\}, F_{\mathcal{A}}))$. The set of all configurations recognized by $\mathcal{A}$ is denoted by $L_{\mathcal{P}}(\mathcal{A})$. A set of configurations of $\mathcal{P}$ is said to be recognizable if and only if it is accepted by some $\mathcal{P}$-automaton.

The transition system $\mathcal{T}(\mathcal{P})$ associated with the generalized pushdown automaton $\mathcal{P}$ is defined by the tuple $(Conf(\mathcal{P}), \Sigma, \rightarrow)$ where $\rightarrow$ is the smallest transition relation such that: For every $p, p' \in P$, $\gamma \in \Gamma$, and $a \in \Sigma_\epsilon$, if $\delta(p, \gamma, a, p') \neq \emptyset$, then $(p, \gamma w) \xrightarrow{a}_{\mathcal{T}(\mathcal{P})} (p', uw)$ for all $u \in \delta(p, \gamma, a, p')$ and $w \in \Gamma^*$. Let $L(\mathcal{P}) = Traces_{\mathcal{T}(\mathcal{P})}(\{(p_0, \gamma_0)\}, F \times \{\epsilon\})$ denote the language accepted by $\mathcal{P}$.

Observe that pushdown automata can be seen as a particular class of effective GPA where $\delta(p, \gamma, a, p')$ is a finite set of words for all (p, γ, a, p').

On the other hand, we can show that the class of effective GPA is closed under concatenation, union, Kleene star, projection, homomorphism, and intersection with a regular language. However, effective GPA are not closed under intersection.

4 Computing the Set of Predecessors for a GPA

In this section, we show that the set of predecessors of a recognizable set of configurations of an effective GPA is recognizable and effectively constructible. This is done by adapting the construction given in [4]. On the other hand, it is easy to observe that the set of successors of a recognizable set of configurations of an effective GPA is not recognizable in general.

Theorem 1. *For every effective generalized pushdown automaton $\mathcal{P}$, and every $\mathcal{P}$-automaton $\mathcal{A}$, it is possible to construct a $\mathcal{P}$-automaton recognizing $Pre^*_{\mathcal{T}(\mathcal{P})}(L_{\mathcal{P}}(\mathcal{A}))$.*

Proof. Let $\mathcal{P} = (P, \Sigma, \Gamma, \delta, p_0, \gamma_0, F)$ be an effective generalized pushdown automata and $\mathcal{A} = (Q_{\mathcal{A}}, \Gamma, \Delta_{\mathcal{A}}, I_{\mathcal{A}}, F_{\mathcal{A}})$ be a $\mathcal{P}$-automaton. Without loss of generality, we assume that $\mathcal{A}$ has no transition leading to an initial state. We compute $Pre^*_{\mathcal{T}(\mathcal{P})}(L_{\mathcal{P}}(\mathcal{A}))$ as the set of configurations recognized by a $\mathcal{P}$-automaton $\mathcal{A}_{pre^*} = (Q_{\mathcal{A}}, \Gamma, \Delta_{pre^*}, I_{\mathcal{A}}, F_{\mathcal{A}})$ obtained from $\mathcal{A}$ by means of a saturation procedure. Initially, we have $\mathcal{A}_{pre^*} = \mathcal{A}$. Then, the procedure adds new transitions to $\mathcal{A}_{pre^*}$, but no new states. New transitions are added according to the following saturation rule:

For every $p, p' \in P$, $\gamma \in \Gamma$, and $a \in \Sigma_\epsilon$, if $\delta(p, \gamma, a, p') \neq \emptyset$, then for every $q \in Q_{\mathcal{A}}$ such that $\delta(p, \gamma, a, p') \cap L(\mathcal{A}_{pre^*}(\{p'\}, \{q\})) \neq \emptyset$, add the transition (p, γ, q) to $\mathcal{A}_{pre^*}$

It is easy to see that the saturation procedure eventually reaches a fixed point because the number of possible new transitions is finite. Moreover, the saturation procedure is well defined since the emptiness problem of the language $\big(\delta(p, \gamma, a, p') \cap L(\mathcal{A}_{pre^*}(\{p'\}, \{q\}))\big)$ is decidable ($\mathcal{P}$ is an effective GPA). Then, the relation between the set of configurations recognized by $\mathcal{A}_{pre^*}$ and the set $Pre^*_{\mathcal{T}(\mathcal{P})}(L_{\mathcal{P}}(\mathcal{A}))$ is established by Lemma 1.

Lemma 1. $L_{\mathcal{P}}(\mathcal{A}_{pre^*}) = Pre^*_{\mathcal{T}(\mathcal{P})}(L_{\mathcal{P}}(\mathcal{A}))$. □

As an immediate consequence of Theorem 1, we obtain the decidability of the emptiness problem and the membership for effective generalized pushdown automata.

Theorem 2 (EMPTINESS, MEMBERSHIP). *The emptiness and the membership problems are decidable for effective generalized pushdown automata.*

5 Ordered Multi-Pushdown Automata

In this section, we first recall the definition of *multi-pushdown automata*. Then *ordered multi-pushdown automata* [6,1] appear as a special case of multi-pushdown automata.

5.1 Multi-pushdown Automata

Multi-pushdown automata have one read-only left to right input tape and $n \geq 1$ read-write memory tapes (stacks) with a last-in-first-out rewriting policy. A transition is of the form $t = \langle q, \gamma_1, \ldots, \gamma_n \rangle \xrightarrow{a} \langle q', \alpha_1, \ldots, \alpha_n \rangle$. Being in a configuration $(p, w_1, \ldots, w_n)$, which is composed of a state p and a stack content w_i for each stack i, t can be applied if both $q = p$ and the i-th stack is of the form $\gamma_i w_i'$ for some w_i'. Taking the transition and reading the a (which might be the empty word), the system moves to the successor configuration $(q', \alpha_1 w_1', \ldots, \alpha_n w_n')$.

Definition 3 (Multi-pushdown automata). *A* multi-pushdown automaton *(MPA) is a tuple* $\mathcal{M} = (n, Q, \Sigma, \Gamma, \Delta, q_0, \gamma_0, F)$ *where:*

- $n \geq 1$ *is the number of stacks.*
- Q *is the finite non-empty set of* states.
- Σ *is the finite set of* input symbols.
- Γ *is the finite set of* stack symbols *containing the special stack symbol* $\perp$.
- $\Delta \subseteq \big(Q \times (\Gamma_\epsilon)^n\big) \times \Sigma_\epsilon \times \big(Q \times (\Gamma^*)^n\big)$ *is the* transition relation *such that, for all* $((q, \gamma_1, \ldots, \gamma_n), a, (q', \alpha_1, \ldots, \alpha_n)) \in \Delta$ *and* $i \in [1, n]$*, we have:*
  - $|\alpha_i| \leq 2$.
  - *If* $\gamma_i \neq \perp$*, then* $\alpha_i \in (\Gamma \setminus \{\perp\})^*$.
  - *If* $\gamma_i = \perp$*, then* $\alpha_i = \alpha_i' \perp$ *for some* $\alpha_i' \in \Gamma_\epsilon$.
- $q_0 \in Q$ *is the* initial state.
- $\gamma_0 \in (\Gamma \setminus \{\perp\})$ *is the* initial stack symbol.
- $F \subseteq Q$ *is the set of* final states.

The size of $\mathcal{M}$, denoted by $|\mathcal{M}|$, is defined by $(n + |Q| + |\Sigma| + |\Gamma|)$. In the rest of this paper, we use $\langle q, \gamma_1, \ldots, \gamma_n \rangle \xrightarrow{a}_{\mathcal{M}} \langle q', \alpha_1, \ldots, \alpha_n \rangle$ to denote that the transition $((q, \gamma_1, \ldots, \gamma_n), a, (q', \alpha_1, \ldots, \alpha_n))$ is in Δ. Moreover, we denote by $\mathcal{M}(q, \gamma, q')$ the multi-pushdown automaton defined by the tuple $(n, Q, \Sigma, \Gamma, \Delta, q, \gamma, \{q'\})$.

A stack content of $\mathcal{M}$ is a sequence from $Stack(\mathcal{M}) = (\Gamma \setminus \{\perp\})^* \{\perp\}$. A configuration of $\mathcal{M}$ is a $(n+1)$-tuple $(q, w_1, \ldots, w_n)$ with $q \in Q$, and $w_1, \ldots, w_n \in Stack(\mathcal{M})$. A configuration $(q, w_1, \ldots, w_n)$ is final if $q \in F$ and $w_1 = \cdots = w_n = \perp$. The set of configurations of $\mathcal{M}$ is denoted by $Conf(\mathcal{M})$.

The behavior of the MPA $\mathcal{M}$ is described by its corresponding transition system $\mathcal{T}(\mathcal{M})$ defined by the tuple $(Conf(\mathcal{M}), \Sigma, \rightarrow)$ where $\rightarrow$ is the smallest transition relation satisfying the following condition: if $\langle q, \gamma_1, \ldots, \gamma_n \rangle \xrightarrow{a}_{\mathcal{M}} \langle q', \alpha_1, \ldots, \alpha_n \rangle$, then $(q, \gamma_1 w_1, \ldots, \gamma_n w_n) \xrightarrow{a}_{\mathcal{T}(\mathcal{M})} (q', \alpha_1 w_1, \ldots, \alpha_n w_n)$ for all $w_1, \ldots, w_n \in \Gamma^*$ such that $\gamma_1 w_1, \ldots, \gamma_n w_n \in Stack(\mathcal{M})$. Observe that the symbol $\perp$ marks the bottom of a stack. According to the transition relation, $\perp$ can never be popped.

The language accepted (or recognized) by $\mathcal{M}$ is defined by the set $L(\mathcal{M}) = \{\tau \in \Sigma^* \mid (q_0, \gamma_0 \perp, \perp, \ldots, \perp) \overset{\tau}{\Longrightarrow}^*_{\mathcal{T}(\mathcal{M})} c \text{ for some final configuration } c\}$.

5.2 Ordered Multi-pushdown Automata

An ordered multi-pushdown automaton is a multi-pushdown automaton in which one can pop only from the first non-empty stack (i.e., all preceding stacks are equal to $\bot$). In the following, we consider only ordered multi-pushdown automata in normal form with respect to the definitions in [1].

Definition 4 (Ordered multi-pushdown automata). *An ordered multi-pushdown automaton (OMPA for short) is a multi-pushdown automaton* $(n, Q, \Sigma, \Gamma, \Delta, q_0, \gamma_0, F)$ *such that* Δ *contains only the following types of transitions:*

- $\langle q, \gamma, \epsilon, \ldots, \epsilon\rangle \xrightarrow{a}_{\mathcal{M}} \langle q', \gamma''\gamma', \epsilon, \ldots, \epsilon\rangle$ *for some* $q, q' \in Q$, $\gamma, \gamma', \gamma'' \in (\Gamma \setminus \{\bot\})$ *and* $a \in \Sigma_\epsilon$.
- $\langle q, \gamma, \epsilon, \ldots, \epsilon\rangle \xrightarrow{a}_{\mathcal{M}} \langle q', \epsilon, \ldots, \epsilon, \gamma', \epsilon, \ldots, \epsilon\rangle$ *for some* $q, q' \in Q$, $\gamma, \gamma' \in (\Gamma\setminus\{\bot\})$ *and* $a \in \Sigma_\epsilon$ *(γ' is pushed on one of stacks 2 to n).*
- $\langle q, \bot, \ldots, \bot, \gamma, \epsilon, \ldots, \epsilon\rangle \xrightarrow{a}_{\mathcal{M}} \langle q', \gamma'\bot, \bot, \ldots, \bot, \epsilon, \epsilon, \ldots, \epsilon\rangle$ *for some* $q, q' \in Q$, $\gamma, \gamma' \in (\Gamma \setminus \{\bot\})$ *and* $a \in \Sigma_\epsilon$ *(γ is popped from one of the stacks 2 to n).*
- $\langle q, \gamma, \epsilon, \ldots, \epsilon\rangle \xrightarrow{a} \langle q', \epsilon, \ldots, \epsilon\rangle$ *for some* $q, q' \in Q$, $\gamma \in (\Gamma \setminus \{\bot\})$ *and* $a \in \Sigma_\epsilon$

We introduce the following abbreviations: (1) For $n \geq 1$, we call a MPA/OMPA a n-MPA/n-OMPA, respectively, if its number of stacks is n, and (2) A MPA *over* Σ is a MPA with input alphabet Σ.

Next, we recall some properties of the class of languages recognized by n-OMPA.

Lemma 2 ([6]). *If $\mathcal{M}_1$ and $\mathcal{M}_2$ are two n-OMPAs over an alphabet Σ, then it is possible to construct a n-OMPA $\mathcal{M}$ over Σ such that:* (1) $L(\mathcal{M}) = L(\mathcal{M}_1) \cup L(\mathcal{M}_2)$ *and* $|\mathcal{M}| = O(|\mathcal{M}_1| + |\mathcal{M}_2|)$.

Lemma 3 ([6]). *Let Σ be an alphabet. Given a n-OMPA $\mathcal{M}$ over Σ and a finite state automaton $\mathcal{A}$ over Σ, then it is possible to construct a n-OMPA $\mathcal{M}'$ such that:* $L(\mathcal{M}') = L(\mathcal{M}) \cap L(\mathcal{A})$ *and* $|\mathcal{M}'| = O(|\mathcal{M}| \cdot |\mathcal{A}|)$.

6 The Emptiness Problem for a n-OMPA Is in 2ETIME

In this section, we show that the emptiness problem for ordered pushdown automata is in 2ETIME. (We provide here a simpler proof of the 2ETIME upper bound than the one given in [1].) To this aim, we proceed as follows:

- First, we show that, given a n-OMPA $\mathcal{M}$ with $n > 1$, it is possible to construct a GPA $\mathcal{P}$ with transition languages defined by $(n-1)$-OMPAs of size $O(|\mathcal{M}|^2)$ such that the emptiness problem of $\mathcal{M}$ can be reduced to the emptiness problem of $\mathcal{P}$. Let us present the main steps of this construction. For that, let us consider an accepting run ρ of $\mathcal{M}$. This run can be seen as a sequence of runs of the form $\varsigma_1\sigma_1\varsigma_2\sigma_2\cdots\varsigma_m\sigma_m$ such that pop operations are exclusive to the first $(n-1)$-stacks (resp. the n-th stack) of $\mathcal{M}$ during the sequence of runs $\varsigma_1, \varsigma_2, \ldots, \varsigma_m$ (resp. $\sigma_1, \sigma_2, \ldots, \sigma_m$). Observe that, by definition, the first $(n-1)$-stacks of $\mathcal{M}$ are empty along the runs $\sigma_1, \sigma_2, \ldots, \sigma_m$. Moreover, at the beginning of the runs

$\varsigma_1, \varsigma_2, \ldots, \varsigma_m$, the OMPA $\mathcal{M}$ is in some configuration c where the first stack of $\mathcal{M}$ contains just one symbol and the stacks from 2 to $n-1$ are empty (i.e., c of the form $(q, \gamma\bot, \bot, \ldots, \bot, w)$).

Then, we construct $\mathcal{P}$ such that the following invariant is preserved during the simulation of $\mathcal{M}$: The state and the content of the stack of $\mathcal{P}$ are the same as the state and the content of the n-th stack of $\mathcal{M}$ when its first $(n-1)$-stacks are empty (and so, $L(\mathcal{P}) \neq \emptyset$ if and only if $L(\mathcal{M}) \neq \emptyset$). To this aim, a pushdown operation of $\mathcal{M}$ that pops a symbol γ from the n-th stack is simply simulated by a pushdown operation of $\mathcal{P}$ that pops the symbol γ. This implies that a run of the form σ_i, with $1 \leq i \leq m$, that pops the word u_i from the n-th stack of $\mathcal{M}$ is simulated by a run of $\mathcal{P}$ that pops the same word u_i. Now, for every $j \in [1, m]$, we need to compute the pushed word v_j into the n-th stack of $\mathcal{M}$ during the run ς_j in order to be pushed also by $\mathcal{P}$. For that, let $L_{(q,\gamma,q')}$ be the set of all possible pushed words u into the n-th stack of $\mathcal{M}$ by a run $(q, \gamma\bot, \bot, \ldots, \bot, w) \stackrel{\tau}{\Longrightarrow}^*_{\mathcal{T}(\mathcal{M})} (q', \bot, \bot, \ldots, \bot, uw)$ where pop operations are exclusive to the first $(n-1)$-stacks of $\mathcal{M}$. We show that this language $L_{(q,\gamma,q')}$ can be defined by a $(n-1)$-OMPA $\mathcal{M}'(q, \gamma, q')$ over the stack alphabet of $\mathcal{M}$ that: (1) performs the same operations on its state and $(n-1)$-stacks as the one performed by $\mathcal{M}$ on its state and its first $(n-1)$ stacks while discarding the pop operations of $\mathcal{M}$ over the n^{th} stack, and (2) makes visible as transition labels the pushed symbols over the n^{th} stack of $\mathcal{M}$. Now, to simulate the run $\varsigma_j = (q_j, \gamma_j\bot, \bot, \ldots, \bot, w_j) \stackrel{\tau_j}{\Longrightarrow}^*_{\mathcal{T}(\mathcal{M})} (q'_j, \bot, \bot, \ldots, \bot, u_j w_j)$ of $\mathcal{M}$, $\mathcal{P}$ can push into its stack the word $u_j \in L(\mathcal{M}'(q_j, \gamma_j, q'_j))$.

– Then, we prove, by induction on n, that the emptiness problem for the n-OMPA $\mathcal{M}$ is in 2ETIME with respect to the number of stacks. For that, we assume that the emptiness problem for $(n-1)$-OMPAs can be solved in 2ETIME. This implies that the GPA $\mathcal{P}$ (that simulates $\mathcal{M}$) is effective (see Definition 2 and Lemma 3). Now, we can use Theorem 2 to prove the decidability of the emptiness problem of the effective GPA $\mathcal{P}$ (and so, of the n-OMPA $\mathcal{M}$). To show that the emptiness problem of $\mathcal{P}$ and $\mathcal{M}$ is in 2ETIME, we estimate the running time of our saturation procedure, given in section 4, under the assumption that the emptiness problem for $(n-1)$-OMPAs can be solved in 2ETIME.

Let us give in more details of the proof described above.

6.1 Simulation of a n-OMPA by a GPA

In the following, we prove that, given an OMPA $\mathcal{M}$, we can construct a GPA $\mathcal{P}$, with transition languages defined by $(n-1)$-OMPAs of size $O(|\mathcal{M}|^2)$, such that the emptiness problem for $\mathcal{M}$ is reducible to the emptiness problem for $\mathcal{P}$.

Theorem 3. *Given an OMPA $\mathcal{M} = (n, Q, \Sigma, \Gamma, \Delta, q_0, \gamma_0, F)$ with $n > 1$, it is possible to construct a GPA $\mathcal{P} = (P, \Sigma', \Gamma, \delta, p_0, \bot, \{p_f\})$ such that $P = Q \cup \{p_0, p_f\}$, $\Sigma' = Q$, and we have:*

– *$L(\mathcal{M}) \neq \emptyset$ if and only if $L(\mathcal{P}) \neq \emptyset$, and*

– *For every* $p_1, p_2 \in P$, $a \in \Sigma'_\epsilon$, *and* $\gamma \in \Gamma$, *there is a* $(n-1)$*-OMPA* $\mathcal{M}_{(p_1,\gamma,a,p_2)}$ *over* Γ *s.t.* $L(\mathcal{M}_{(p_1,\gamma,a,p_2)}) = \big(\delta(p_1,\gamma,a,p_2)\big)^R$ *and* $|\mathcal{M}_{(p_1,\gamma,a,p_2)}| = O(|\mathcal{M}|^2)$.

The proof of Theorem 3 is structured as follows. First, we define a $(n-1)$-OMPA $\mathcal{M}'$ over the alphabet Γ that: (1) performs the same operations on its state and $(n-1)$-stacks as the one performed by $\mathcal{M}$ on its state and its first $(n-1)$ stacks while discarding the pop operations of $\mathcal{M}$ on the n^{th} stack, and (2) makes visible as transition labels the pushed symbols over the n^{th} stack of $\mathcal{M}$. Intuitively, depending on the initial and final configurations of $\mathcal{M}'$, the "language" of $\mathcal{M}'$ summarizes the effect of a sequence of pop operations of $\mathcal{M}$ over the first $(n-1)$-stacks on the n^{th} stack of $\mathcal{M}$. So, if we are interested only by the configurations of $\mathcal{M}$ where the first $(n-1)$ stacks are empty, a run of $\mathcal{M}$ can be seen as a sequence of alternations of a pop operation of $\mathcal{M}$ over the n^{th} stack and a push operation over the n^{th} stack of a word in the "language" of $\mathcal{M}'$.

Then, we construct a generalized pushdown automaton $\mathcal{P}$ such that the state and the stack content of $\mathcal{P}$ are the same as the state and the n^{th}-stack content of $\mathcal{M}$ when the first $(n-1)$ stacks of $\mathcal{M}$ are empty. In the definition of $\mathcal{P}$, we use the $(n-1)$-OMPA $\mathcal{M}'$ to characterize the pushed word on the n^{th} stack of $\mathcal{M}$ due to a sequence of pop operations of $\mathcal{M}$ on the $(n-1)$ first stacks of $\mathcal{M}$. This implies that the emptiness problem for $\mathcal{M}$ is reducible to its corresponding problem for $\mathcal{P}$

Constructing the $(n-1)$-OMPA $\mathcal{M}'$: Let us introduce the following the n-OMPA $\mathcal{M}_{[1,n[} = (n, Q, \Sigma, \Gamma, \Delta_{[1,n[}, q_0, \gamma_0, F)$ such that $\Delta_{[1,n[} = \Delta \cap \big(Q \times (\Gamma_\epsilon)^{n-1} \times \{\epsilon\}) \times \Sigma_\epsilon \times (Q \times (\Gamma^*)^n)\big)$. Intuitively, $\mathcal{M}_{[1,n[}$ is built up from $\mathcal{M}$ by discarding pop operations of $\mathcal{M}$ over the n^{th} stack. Then, let $\mathcal{M}' = (n-1, Q, \Gamma, \Gamma, \Delta', q_0, \gamma_0, F)$ be the $(n-1)$-OMPA, built out from $\mathcal{M}_{[1,n[}$, which (1) performs the same operations on the first $n-1$ stacks of $\mathcal{M}_{[1,n[}$, and (2) makes visible as transition label the pushed stack symbol over the n^{th} stack of $\mathcal{M}_{[1,n[}$. Formally, Δ' is defined as the smallest transition relation satisfying the following conditions:

– If $\langle q, \gamma_1, \ldots, \gamma_{n-1}, \epsilon\rangle \xrightarrow{a}_{\mathcal{M}_{[1,n[}} \langle q', \alpha_1, \ldots, \alpha_{n-1}, \epsilon\rangle$ for some $q, q' \in Q$, $\gamma_1, \ldots, \gamma_{n-1} \in \Gamma_\epsilon$, $a \in \Sigma_\epsilon$, and $\alpha_1, \ldots, \alpha_{n-1} \in \Gamma^*$, then $\langle q, \gamma_1, \ldots, \gamma_{n-1}\rangle \xrightarrow{\epsilon}_{\mathcal{M}'} \langle q', \alpha_1, \ldots, \alpha_{n-1}\rangle$.
– If $\langle q, \gamma, \epsilon, \ldots, \epsilon\rangle \xrightarrow{a}_{\mathcal{M}_{[1,n[}} \langle q', \epsilon, \ldots, \epsilon, \gamma'\rangle$ for some $q, q' \in Q$, $a \in \Sigma_\epsilon$, and $\gamma, \gamma' \in (\Gamma \setminus \{\bot\})$, then $\langle q, \gamma, \epsilon, \ldots, \epsilon\rangle \xrightarrow{\gamma'}_{\mathcal{M}'} \langle q', \epsilon, \ldots, \epsilon\rangle$.

Let us now give the relation between the effect of a sequence of operations of $\mathcal{M}_{[1,n[}$ on the n^{th}-stack and the language of $\mathcal{M}'$.

Lemma 4. *For every* $q, q' \in Q$, *and* $w_1, w'_1, \ldots, w_n, w'_n \in Stack(\mathcal{M}_{[1,n[})$, $(q, w_1, \ldots, w_n) \overset{\tau}{\Longrightarrow}^*_{\mathcal{T}(\mathcal{M}_{[1,n[})} (q', w'_1, \ldots, w'_n)$ *for some* $\tau \in \Sigma^*$ *if and only if there is* $u \in \Gamma^*$ *such that* $(q, w_1, \ldots, w_{n-1}) \overset{u}{\Longrightarrow}^*_{\mathcal{T}(\mathcal{M}')} (q', w'_1, \ldots, w'_{n-1})$ *and* $w'_n = u^R w_n$.

Constructing the GPA $\mathcal{P}$: We are ready now to define the generalized pushdown automaton $\mathcal{P} = (P, \Sigma', \Gamma, \delta, p_0, \bot, \{p_f\})$, with $P = Q \cup \{p_0, p_f\}$ and $\Sigma' = Q$, that keeps track of the state and the content of the n-th stack of $\mathcal{M}$ when the first $(n-1)$ stacks are empty. Formally, $\mathcal{P}$ is built from $\mathcal{M}$ as follows: For every $p, p' \in P$, $q \in \Sigma'_\epsilon$, and $\gamma \in \Gamma$, we have:

- If $p = p_0$, $\gamma = \bot$, $q = q_0$, and $p' \in Q$, then $\delta(p, \gamma, q, p') = \{u^R \bot \mid u \in L(\mathcal{M}'(q_0, \gamma_0, p'))\}$.
- If $p \in F$, $\gamma = \bot$, $q = \epsilon$, and $p' = p_f$, then $\delta(p, \gamma, q, p') = \{\epsilon\}$.
- If $p, p' \in Q$, $\gamma \neq \bot$, and $q \in Q$, then $\delta(p, \gamma, q, p') = \bigcup_{\gamma' \in \Gamma'} \left(L(\mathcal{M}'(q, \gamma', p'))\right)^R$ where $\Gamma' = \{\gamma' \in \Gamma \mid \exists a \in \Sigma_\epsilon,\ \langle p, \bot, \ldots, \bot, \gamma\rangle \xrightarrow{a}_{\mathcal{M}} \langle q, \gamma' \bot, \bot, \ldots, \bot, \epsilon\rangle\}$.
- Otherwise, $\delta(p, \gamma, q, p') = \emptyset$.

Observe that for every $p_1, p_2 \in P$, $q \in Q_\epsilon$, and $\gamma \in \Gamma$, we can construct an $(n-1)$-OMPA $\mathcal{M}_{(p_1,\gamma,q,p_2)}$ over Γ such that $L(M_{(p_1,\gamma,q,p_2)}) = \left(\delta(p_1, \gamma, q, p_2)\right)^R$ and $|\mathcal{M}_{(p_1,\gamma,q,p_2)}| = O(|\mathcal{M}|^2)$. This can be easily proved using Lemma 2.

To complete the proof of Theorem 3, it remains to show that the emptiness problem for $\mathcal{M}$ is reducible to its corresponding problem for $\mathcal{P}$. This is stated by Lemma 5.

Lemma 5. *$L(\mathcal{M}) \neq \emptyset$ if and only if $L(\mathcal{P}) \neq \emptyset$.*

6.2 Emptiness of a n-OMPA Is in 2ETIME

In the following, we show that the emptiness problem for a n-OMPA is in 2ETIME.

Theorem 4. *The emptiness problem for a n-OMPA $\mathcal{M}$ can be solved in time $O(|\mathcal{M}|^{2^{dn}})$ for some constant d.*

Proof. Let $\mathcal{M} = (n, Q, \Sigma, \Gamma, \Delta, q_0, \gamma_0, F)$ be a n-OMPA. To prove Theorem 4, we proceed by induction on n.

Basis. $n = 1$. Then, $\mathcal{M}$ is a pushdown automaton. From [4], we know that the emptiness problem for $\mathcal{M}$ can be solved in time polynomial in $|\mathcal{M}|$.

Step. $n > 1$. Then, we can apply Theorem 3 to construct a generalized pushdown automaton $\mathcal{P} = (P, Q, \Gamma, \delta, p_0, \bot, \{p_f\})$, with $P = Q \cup \{p_0, p_f\}$, such that:

- $L(\mathcal{P}) \neq \emptyset$ if and only if $L(\mathcal{M}) \neq \emptyset$, and
- For every $p_1, p_2 \in P$, $a \in Q_\epsilon$, and $\gamma \in \Gamma$, there is a $(n-1)$-OMPA $\mathcal{M}_{(p_1,\gamma,a,p_2)}$ over Γ s.t. $L(M_{(p_1,\gamma,a,p_2)}) = \left(\delta(p_1, \gamma, a, p_2)\right)^R$ and $|\mathcal{M}_{(p_1,\gamma,a,p_2)}| = O(|\mathcal{M}|^2)$.

It is easy to observe that $\mathcal{P}$ is an effective generalized pushdown automaton. This is established by the following lemma.

Lemma 6. *$\mathcal{P}$ is an effective generalized pushdown automaton.*

From Theorem 2, Theorem 3, and Lemma 6, we deduce that the emptiness problem for the n-OMPA $\mathcal{M}$ is decidable. Let us now estimate the running time of the decision procedure. From Theorem 2, we know that the emptiness problem of $\mathcal{P}$ is reducible to compute the set of predecessors of the configuration (p_f, ϵ) since $L(\mathcal{P}) \neq \emptyset$ if and only if $(p_0, \bot) \in Pre^*_{\mathcal{T}(\mathcal{P})}(\{p_f\} \times \{\epsilon\})$.

Let $\mathcal{A}$ be the $\mathcal{P}$-automaton that recognizes the configuration (p_f, ϵ) of $\mathcal{P}$. It is easy to see that such $\mathcal{P}$-automaton $\mathcal{A}$, with $|\mathcal{A}| = O(|\mathcal{M}|)$, is effectively constructible. Now,

we need to analysis the running time of the saturation procedure (given in section 4) applied to $\mathcal{A}$. For that, let $\mathcal{A}_0, \ldots, \mathcal{A}_i$ be the sequence of $\mathcal{P}$-automaton obtained from the saturation procedure such that $\mathcal{A}_0 = \mathcal{A}$ and $L_{\mathcal{P}}(\mathcal{A}_i) = Pre^*_{\mathcal{T}(\mathcal{P})}(L_{\mathcal{P}}(\mathcal{A}))$. Then, we have $i = O(|\mathcal{M}|^3)$ since the number of possible new transitions of $\mathcal{A}$ is finite. Moreover, at each step j, with $0 \leq j \leq i$, we need to check, for every $q \in Q_{\mathcal{A}}$, $p, p' \in P$, $\gamma \in \Gamma$, and $a \in Q_\epsilon$, whether $L(\mathcal{A}_j)(\{p'\}, \{q\}) \cap \delta(p, \gamma, a, p') \neq \emptyset$.

Using Lemma 3, we can construct, in time polynomial in $|\mathcal{M}|$, a $(n-1)$-OMPA $\mathcal{M}'_{(q,p,\gamma,a,p')}$ such that $L(\mathcal{M}'_{(q,p,\gamma,a,p')}) = (L(\mathcal{A}_j)(\{p'\}, \{q\}))^R \cap L(M_{(p,\gamma,a,p')})$ and $|\mathcal{M}'_{(q,p,\gamma,a,p')}| \leq c(|\mathcal{M}|^3)$ for some constant c. Now, we can apply the induction hypothesis to $\mathcal{M}'$, and we obtain that the problem of checking whether $L(\mathcal{M}'_{(q,p,\gamma,a,p')}) \neq \emptyset$ can be solved in time $O\big((c\,|\mathcal{M}|^3)^{2^{d(n-1)}}\big)$. Putting together all these equations, we obtain that the problem of checking whether $(p_0, \bot) \in Pre^*_{\mathcal{T}(\mathcal{P})}(\{p_f\} \times \{\epsilon\})$ can be solved in time $O\big(|\mathcal{M}|^3|\mathcal{M}|^5(c\,|\mathcal{M}|^3)^{2^{d(n-1)}}\big)$. By taking a constant d as big as needed, we can show that the problem of checking whether $L(\mathcal{M}) \neq \emptyset$ can be solved in time $O(|\mathcal{M}|^{2^{dn}})$. □

7 Bounded-Phase Visibly Multi-Pushdown Automata

In this section, we recall the definition of visibly multi-pushdown automata [16], another subclass of MPA, where an action is associated with a particular stack operation. An action can be a push, pop, or internal action. Formally, a visibly multi-pushdown automaton (VMPA for short) is a tuple $\mathcal{V} = (\mathcal{M}, type)$ where $\mathcal{M} = (n, Q, \Sigma, \Gamma, \Delta, q_0, \gamma_0, F)$ is a MPA and $type : \Sigma \to (\{Push, Pop\} \times [1, n]) \cup \{Int\}$ is a function satisfying, for all transitions $\langle q, \gamma_1, \ldots, \gamma_n\rangle \xrightarrow{a}_{\mathcal{M}} \langle q', \alpha_1, \ldots, \alpha_n\rangle$:

- $a \neq \epsilon$,
- if $type(a) = (Push, i)$ for some $i \in [1, n]$, then $\gamma_1 = \ldots = \gamma_n = \epsilon$, $\alpha_i \in (\Gamma \setminus \{\bot\})$, and $\alpha_j = \epsilon$ for all $j \in [1, n] \setminus \{i\}$,
- if $type(a) = (Pop, i)$ for some $i \in [1, n]$, then $\gamma_i \in \Gamma$, $\alpha_i \in \{\bot\} \cup \{\epsilon\}$, and $\gamma_j = \alpha_j = \epsilon$ for all $j \in [1, n] \setminus \{i\}$, and
- if $type(a) = Int$, then $\gamma_i = \alpha_i = \epsilon$ for all $i \in [1, n]$.

If, in a VMPA, we restrict the number of phases, where in one phase pop operations are exclusive to one stack, then we obtain bounded-phase visibly multi-pushdown automata. A bounded-phase visibly multi-pushdown automaton (BVMPA for short) is a triple $\mathcal{B} = (\mathcal{M}, type, \tau)$ where $\mathcal{M} = (n, Q, \Sigma, \Gamma, \Delta, q_0, \gamma_0, F)$ is a MPA, τ is a word over $[1, n]$, and $(\mathcal{M}, type)$ is a VMPA. For every $i \in [1, n]$, let $\Sigma_i = \{a \in \Sigma \mid \nexists j \neq i, type(a) = (Pop, j)\}$. The language accepted by $\mathcal{B}$ is defined as follows: $L(\mathcal{B}) = L(\mathcal{M}) \cap (\Sigma^*_{\tau(1)}\Sigma^*_{\tau(2)} \cdots \Sigma^*_{\tau(|\tau|)})$

In the rest of this paper, we use a τ-phase n-BVMPA over Σ to denote a BVMPA of the form $(\mathcal{M}, type, \tau)$ with $\mathcal{M}$ is a n-MPA over Σ.

Next, we show that the class of languages accepted bounded-phase visibly multi-pushdown automata is closed under intersection with a regular languages.

Lemma 7. *Let Σ be an alphabet. Given a τ-phase n-BVMPA $\mathcal{B} = (\mathcal{M}, type, \tau)$ over Σ and a FSA $\mathcal{A}$ over Σ, it is possible to construct a τ-phase n-BVMPA $\mathcal{B}' = (\mathcal{M}', type, \tau)$ such that $L(\mathcal{B}') = L(\mathcal{B}) \cap L(\mathcal{A})$ and $|\mathcal{M}'| = O(|\mathcal{M}||\mathcal{A}|)$.*

Proof. Similar to the proof of Lemma 3. □

8 The Emptiness Problem for a τ-Phase n-BVMPA Is in 2ETIME

In this section, we show that the emptiness problem for bounded-phase visibly multi-pushdown automata is in 2ETIME with respect to the length of τ. To this aim, we proceed as follows: First, we prove that the emptiness for a τ-phase n-BVMPA is reducible to the emptiness problem for a generalized pushdown automaton with transition languages defined by τ'-phase n-BVMPAs with $\tau = \tau' \cdot \tau(|\tau|)$. Then, we use the saturation procedure, given in Section 4 to prove, by induction on $|\tau|$, that the emptiness problem for a τ-phase n-BVMPA is in 2ETIME.

For the rest of this section, let us fix a BVMPA $\mathcal{B} = (\mathcal{M}, type, \tau)$ where $\mathcal{M} = (n, Q, \Sigma, \Gamma, \Delta, q_0, \gamma_0, F)$ is a MPA. We assume w.l.o.g that $\Sigma \cap \Gamma = \emptyset$. For every $i \in [1, n]$, let $\Sigma_i = \{a \in \Sigma \mid \nexists j \neq i, type(a) = (Pop, j)\}$. Moreover, let us assume that $k = \tau(|\tau|)$ and $\tau = \tau' k$.

8.1 Simulation of a τ-Phase n-BVMPA by a GPA

In the following, we prove that it is possible to construct a GPA $\mathcal{P}$ such that the emptiness problem for $\mathcal{B}$ is reducible to the emptiness problem for $\mathcal{P}$.

Theorem 5. *Assume that $k > 1$. Then, it is possible to construct a GPA $\mathcal{P} = (P, \Sigma, \Gamma, \delta, p_0, \bot, \{p_f\})$ such that $P = Q \cup \{p_0, p_f\}$ and we have:*

- *$L(\mathcal{B}) \neq \emptyset$ if and only if $L(\mathcal{P}) \neq \emptyset$, and*
- *For every $p, p' \in P$, $a \in \Sigma_\epsilon$, and $\gamma \in \Gamma$, there is a τ'-phase n-BVMPA $\mathcal{B}_{(p,\gamma,a,p')} = (\mathcal{M}_{(p,\gamma,a,p')}, type', \tau')$ over Σ' such that $\Gamma \subseteq \Sigma'$, $|\mathcal{M}_{(p,\gamma,a,p')}| = O(|\mathcal{M}|^2)$, and $\delta(p, \gamma, a, p') = \big((L(\mathcal{B}_{(p,\gamma,a,p')})|_\Gamma)\big)^R\{\bot\}$.*

Proof. By definition, $L(\mathcal{B}) \neq \emptyset$ if and only if there are $q \in F$ and $\sigma_j \in \Sigma^*_{\tau(j)}$ for all $j \in [1, |\tau|]$ such that $\rho = (q_0, \gamma_0\bot, \bot, \ldots, \bot) \xrightarrow{\sigma_1 \cdots \sigma_{|\tau|}}{}^*_{\mathcal{T}(\mathcal{M})} (q, \bot, \ldots, \bot)$. Thus, the emptiness problem for $\mathcal{B}$ can be reduced to the problem of checking whether there are $q' \in Q$, $w_1, \ldots, w_n \in Stack(\mathcal{M})$, $q \in F$, and $\sigma_j \in \Sigma^*_{\tau(j)}$ for all $j \in [1, |\tau|]$ such that: (1) $w_l = \bot$ for all $l \in [1, n]$ and $l \neq k$, (2) $\rho_1 = (q_0, \gamma_0\bot, \bot, \ldots, \bot) \xrightarrow{\sigma_1 \cdots \sigma_{(|\tau|-1)}}{}^*_{\mathcal{T}(\mathcal{M})} (q', w_1, \ldots, w_n)$, and (3) $\rho_2 = (q', w_1, \ldots, w_n) \xrightarrow{\sigma_{(|\tau|)}}{}^*_{\mathcal{T}(\mathcal{M})} (q, \bot, \ldots, \bot)$. Observe that at the configuration $(q', w_1, \ldots, w_n)$ only the content of the k-stack of $\mathcal{M}$ can be different from $\bot$. Moreover, during the run ρ_2, pop and push operations are exclusive to the k-th stack of $\mathcal{M}$. So, in order to prove Theorem 5, it is sufficient to show that all possible contents w_k of the k-stack of $\mathcal{M}$ reached by the run ρ_1 can be characterized by a language accepted

by a τ'-phase n-BVMPA $\mathcal{B}_{q'} = (\mathcal{M}_{q'}, type', \tau')$ over Σ' such that $\Gamma \subseteq \Sigma'$ (i.e., $w_k \in \big((L(\mathcal{B}_{q'})|_\Gamma)\big)^R\{\bot\}$). Once this is done, we can construct a GPA $\mathcal{P}$ that simulates $\mathcal{B}$. The automaton $\mathcal{P}$ proceeds as follows: First, it pushes in its stack the content w_k of the k-th stack of $\mathcal{M}$ reached by the run ρ_1 using the language of $\mathcal{B}_{q'}$. Then, $\mathcal{P}$ starts to simulate the run ρ_2 by performing the same operations on its state and its stack as the ones performed by $\mathcal{M}$ on its state and its k-th stack. This is possible since along the run ρ_2 pop and push operations of $\mathcal{M}$ are exclusive to k-th stack. Finally, $\mathcal{P}$ checks if the current configurations is final and moves its state to the final state p_f.

Constructing the τ'-phase n-BVMPA $\mathcal{B}_{q'}$: In the following, we show that it is possible to construct, for every $q' \in Q$, a τ'-phase n-BVMPA $\mathcal{B}_{q'}$ such that $w_k \in \big((L(\mathcal{B}_{q'})|_\Gamma)\big)^R\{\bot\}$ if and only if there are $\sigma_j \in \Sigma^*_{\tau(j)}$ for all $j \in [1, |\tau|[$ such that $(q_0, \gamma_0\bot, \bot, \ldots, \bot) \xrightarrow{\sigma_1 \cdots \sigma_{(|\tau|-1)}}{}^*_{\mathcal{T}(\mathcal{M})} (q', w_1, \ldots, w_n)$ where $w_l = \bot$ for all $l \in [1, n]$ and $l \neq k$. For that, let us define the n-MPA $\mathcal{M}'$ that contains all the transitions of $\mathcal{M}$ and that have the ability to push the new fresh symbol $\sharp$ instead of any possible pushed symbol by $\mathcal{M}$ into the k-th stack. Moreover, the symbol $\sharp$ can be popped from the k-th stack at any time by $\mathcal{M}'$ without changing its state. Formally, $\mathcal{M}'$ is defined by the tuple $(n, Q, \Sigma', \Gamma', \Delta', q_0, \gamma_0, F)$ where $\Gamma' = \Gamma \cup \{\sharp\}$ is the stack alphabet, $\Sigma' = \Sigma \cup \Gamma \cup \{\sharp\}$ is the input alphabet, and Δ' is the smallest transition relation satisfying the following conditions:

- $\Delta \subseteq \Delta'$.
- If $\langle q_1, \gamma_1, \ldots, \gamma_n\rangle \xrightarrow{a}_{\mathcal{M}} \langle q_2, \alpha_1, \ldots, \alpha_n\rangle$ and $type(a) = (Push, k)$, then $\langle q_1, \gamma_1, \ldots, \gamma_n\rangle \xrightarrow{\alpha_k}_{\mathcal{M}_{q'}} \langle q_2, \alpha'_1, \ldots, \alpha'_n\rangle$ with $\alpha'_l = \alpha_l$ for all $l \neq k$ and $\alpha'_k = \sharp$.
- For every $q' \in Q$, $\langle q', \gamma_1, \ldots, \gamma_n\rangle \xrightarrow{\sharp}_{\mathcal{M}_{q'}} \langle q', \alpha'_1, \ldots, \alpha'_n\rangle$ if $\gamma_l = \epsilon$ for all $l \neq k$, $\gamma_k = \sharp$, and $\alpha_1 = \ldots = \alpha_n = \epsilon$.

Let $\mathcal{B}' = (\mathcal{M}', type', \tau')$ be a τ'-phase n-BVMPA where the function $type' : \Sigma' \to (\{Push, Pop\} \times [1, n]) \cup \{Int\}$ is defined as follows: $type'(a) = type(a)$ for all $a \in \Sigma$, $type'(\sharp) = (Pop, k)$, and $type'(\gamma) = (Push, k)$ for all $\gamma \in \Gamma$.

Let $\Xi_k = \{a \in \Sigma' \mid type'(a) = (Pop, k)\}$. Then, from Lemma 7, it is possible to construct, for every q', a τ'-phase n-BVMPA $\mathcal{B}_{q'} = (\mathcal{M}'_{q'}, type', \tau')$ from $\mathcal{B}'$ such that $L(\mathcal{B}_{q'}) = L((\mathcal{M}'(q_0, \gamma_0, q'), type', \tau')) \cap (\Sigma \cup \Gamma)^*\{\sharp\}^*(\Sigma' \setminus \Xi_k)^*$. The relation between $\mathcal{B}_{q'}$ and $\mathcal{M}$ is given by Lemma 8 which can be proved by induction.

Lemma 8. *For every $q' \in Q$, $w_k \in ((L(\mathcal{B}_{q'})|_\Gamma)^R\{\bot\})$ iff there are $\sigma_j \in \Sigma^*_{\tau(j)}$ for all $j \in [1, |\tau|[$ such that $(q_0, \gamma_0\bot, \bot, \ldots, \bot) \xrightarrow{\sigma_1 \cdots \sigma_{(|\tau|-1)}}{}^*_{\mathcal{T}(\mathcal{M})} (q', w_1, \ldots, w_n)$ where $w_l = \bot$ for all $l \in [1, n]$ and $l \neq k$.*

Constructing the GPA $\mathcal{P}$: Formally, the transition function δ is defined as follows: for every $p, p' \in P$, $\gamma \in \Gamma$, and $a \in \Sigma$, we have:

- Initialization: $\delta(p, \gamma, a, p') = \big(L(\mathcal{B}_{p'})|_\Gamma\big)\{\bot\}$ if $p = q_0$, $p' \in Q$, $\gamma = \bot$, and $a = \epsilon$. This transition pushes in the stack of $\mathcal{P}$ the content of the k-th stack of $\mathcal{M}$ reached by the run ρ_1.

- Simulation of a push operation on the k-th stack of $\mathcal{M}$: $\delta(p,\gamma,a,p') = \{\alpha_k\gamma \mid \langle p,\gamma_1,\ldots,\gamma_n\rangle \xrightarrow{a}_{\mathcal{M}} \langle p',\alpha_1,\ldots,\alpha_n\rangle\}$ if $p,p' \in Q$, and $a \in \Sigma_k$ such that $type(a) = (Push,k)$.
- Simulation of a pop operation on the k-th stack of $\mathcal{M}$: $\delta(p,\gamma,a,p') = \{\alpha_k \mid \langle p,\gamma_1,\ldots,\gamma_n\rangle \xrightarrow{a}_{\mathcal{M}} \langle p',\alpha_1,\ldots,\alpha_n\rangle, \gamma = \gamma_k\}$ if $p,p' \in Q$ and $a \in \Sigma_k$ such that $type(a) = (Pop,k)$.
- Simulation of an internal operation of $\mathcal{M}$: $\delta(p,\gamma,a,p') = \{\gamma\}$ if $p,p' \in Q$ and $a \in \Sigma_k$ such that $type(a) = Int$.
- Final configuration: $\delta(p,\gamma,a,p') = \{\epsilon\}$ if $p \in F$, $p' = p_f$, $\gamma = \perp$, and $a = \epsilon$.
- Otherwise: $\delta(p,\gamma,a,p') = \emptyset$.

Then, it is easy to see that $L(\mathcal{B}) \neq \emptyset$ if and only if $L(\mathcal{P}) \neq \emptyset$. □

As an immediate consequence of Theorem 5, we obtain that the emptiness problem for a τ-phase n-BVMPA is in 2ETIME.

Theorem 6. *The emptiness problem for a BVMPA $\mathcal{B} = (\mathcal{M}, type, \tau)$ can be solved in time $O(|\mathcal{M}|^{2^{d|\tau|}})$ for some constant d.*

The proof of Theorem 6 is similar to the proof of Theorem 4.

9 Conclusion

In this paper, we have shown that the emptiness problem for both OMPA and BVMPA can be reduced to the one for the class of effective generalized pushdown automata. We provide here simple algorithms for checking the emptiness problem for each of these models and proving their 2ETIME upper bounds.

Recently, A. Seth has showed in [15] that the set of predecessors of a regular set of configurations of a BVMPA is a regular set and effectively constructible. We believe that our automata-based saturation procedure for computing the set of predecessors for an effective GPA can be used to show (by induction on the number of stacks) that the set of predecessors of a regular set of configurations of an OMPA is a regular set and effectively constructible (which may answer to a question raised in [15]).

It is quite easy to see that the model-checking problems for omega-regular properties of effective generalized pushdown automata are decidable. This can be done by adapting the construction given in [4]. This result can be used to establish some decidability/complexity results concerning the model-checking problems for omega-regular properties of both OMPA and BVMPA.

References

1. Atig, M.F., Bollig, B., Habermehl, P.: Emptiness of multi-pushdown automata is 2ETIME-complete. In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257, pp. 121–133. Springer, Heidelberg (2008)
2. Atig, M.F., Bouajjani, A., Touili, T.: On the reachability analysis of acyclic networks of pushdown systems. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 356–371. Springer, Heidelberg (2008)

3. Bouajjani, A., Esparza, J.: Rewriting models of boolean programs. In: Pfenning, F. (ed.) RTA 2006. LNCS, vol. 4098, pp. 136–150. Springer, Heidelberg (2006)
4. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 135–150. Springer, Heidelberg (1997)
5. Bouajjani, A., Müller-Olm, M., Touili, T.: Regular symbolic analysis of dynamic networks of pushdown systems. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 473–487. Springer, Heidelberg (2005)
6. Breveglieri, L., Cherubini, A., Citrini, C., Crespi Reghizzi, S.: Multi-push-down languages and grammars. International Journal of Foundations of Computer Science 7(3), 253–292 (1996)
7. Caucal, D.: On infinite transition graphs having a decidable monadic theory. In: Meyer auf der Heide, F., Monien, B. (eds.) ICALP 1996. LNCS, vol. 1099, pp. 194–205. Springer, Heidelberg (1996)
8. Esparza, J., Knoop, J.: An automata-theoretic approach to interprocedural data-flow analysis. In: Thomas, W. (ed.) FOSSACS 1999. LNCS, vol. 1578, pp. 14–30. Springer, Heidelberg (1999)
9. Kahlon, V.: Boundedness vs. unboundedness of lock chains: Characterizing decidability of pairwise cfl-reachability for threads communicating via locks. In: LICS, pp. 27–36. IEEE Computer Society, Los Alamitos (2009)
10. Lal, A., Reps, T.W.: Reducing concurrent analysis under a context bound to sequential analysis. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 37–51. Springer, Heidelberg (2008)
11. Qadeer, S.: The case for context-bounded verification of concurrent programs. In: Havelund, K., Majumdar, R., Palsberg, J. (eds.) SPIN 2008. LNCS, vol. 5156, pp. 3–6. Springer, Heidelberg (2008)
12. Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 93–107. Springer, Heidelberg (2005)
13. Ramalingam, G.: Context-sensitive synchronization-sensitive analysis is undecidable. ACM Trans. Program. Lang. Syst. 22(2), 416–430 (2000)
14. Reps, T.W., Schwoon, S., Jha, S.: Weighted pushdown systems and their application to interprocedural dataflow analysis. In: Cousot, R. (ed.) SAS 2003. LNCS, vol. 2694, pp. 189–213. Springer, Heidelberg (2003)
15. Seth, A.: Global reachability in bounded phase multi-stack pushdown systems. In: CAV 2010. LNCS. Springer, Heidelberg (2010)
16. La Torre, S., Madhusudan, P., Parlato, G.: A robust class of context-sensitive languages. In: Proceedings of LICS, pp. 161–170. IEEE, Los Alamitos (2007)
17. La Torre, S., Madhusudan, P., Parlato, G.: An infinite automation characterization of double exponential time. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 33–48. Springer, Heidelberg (2008)
18. La Torre, S., Madhusudan, P., Parlato, G.: Reducing context-bounded concurrent reachability to sequential reachability. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 477–492. Springer, Heidelberg (2009)