International Journal of Foundations of Computer Science

EMPTINESS OF ORDERED MULTI-PUSHDOWN AUTOMATA IS 2ETIME-COMPLETE

MOHAMED FAOUZI ATIG
Uppsala University
mohamed_faouzi.atig@it.uu.se

BENEDIKT BOLLIG
LSV, CNRS & ENS Paris-Saclay, France
bollig@lsv.ens-cachan.fr

PETER HABERMEHL
IRIF, CNRS & University Paris Diderot, France
Peter.Habermehl@irif.fr

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)
Communicated by (xxxxxxxxxx)

We consider *ordered multi-pushdown automata*, a multi-stack extension of pushdown automata that comes with a constraint on stack operations: a pop can only be performed on the first non-empty stack (which implies that we assume a linear ordering on the collection of stacks). We show that the emptiness problem for multi-pushdown automata is 2ETIME-complete. Containment in 2ETIME is shown by translating an automaton into a grammar for which we can check if the generated language is empty. The lower bound is established by simulating the behavior of an alternating Turing machine working in exponential space. We also compare ordered multi-pushdown automata with the model of bounded-phase (visibly) multi-stack pushdown automata, which do not impose an ordering on stacks, but restrict the number of alternations of pop operations on different stacks.

Keywords: Multi-pushdown automata, Emptiness problem

68Q50 Grammars, 68Q68 Automata theory

1. Introduction

Various classes of pushdown automata with multiple stacks have been proposed and studied in the literature. The main goals of these efforts are twofold. First, one may aim at extending the expressive power of pushdown automata, going beyond the class of context-free languages. Second, multi-stack systems may model recursive concurrent programs, in which any sequential process is equipped with a finite-state control and, in addition, can access its own stack to connect procedure calls to their

corresponding returns. In general, however, multi-stack extensions of pushdown automata are Turing powerful and therefore come along with undecidability of basic decision problems. To retain desirable decidability properties of pushdown automata, such as emptiness, one needs to restrict the model accordingly. In [9], Breveglieri et al. define *ordered multi-pushdown automata* (OMPA)[a], 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. Another way to regain decidability in the presence of several stacks is to restrict the domain of input words. In [26], La Torre et al. define *bounded-phase visibly multi-stack pushdown automata* (BVMPA). 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-mentioned cases, the emptiness problem is decidable. The usefulness of such results is demonstrated in [27], where the results of [26] are used to show decidability results for restricted queue systems.

In this paper, we resume the study of OMPA and, in particular, consider their emptiness problem. The decidability of this problem, which is to decide if an automaton admits some accepting run, is fundamental for verification purposes. We show that the emptiness problem for OMPA is 2ETIME-complete. 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. In proving the upper bound, we correct an error in the decidability proof given in [9].[b] We keep their main idea: OMPA are reduced to equivalent *depth-n-grammars*. Deciding emptiness for these grammars then amounts to checking emptiness of an ordinary context-free grammar. For proving 2ETIME-hardness, we borrow an idea from [28], where a 2ETIME lower bound is shown for bounded-phase pushdown-transducer automata. We also show that OMPA with $2m$ stacks are strictly more expressive than BVMPA with m phases providing an alternative proof of decidability of the emptiness problem for BVMPA.

Related work

In their full generality multi-stack pushdown automata are not analyzable as two stacks can simulate the tape of a Turing machine. Decidable subclasses are obtained by placing restrictions on the behaviours of the multi-stack pushdown automata. In [9], Breveglieri et al. propose the class of OMPA which impose a linear ordering on stacks. In this paper, we show that the emptiness problem for OMPA is in 2ETIME. In doing so, we correct an error in the decidability proof given in [9].

Context-bounding has been proposed in [20] as a suitable technique for the analysis of multi-stack pushdown automata. The idea is to consider only runs of the automaton that can be split into a given number of contexts, where in each context pop and push operations are exclusive to one stack. The emptiness problem

[a]OMPA are called multi-pushdown automata in [9].
[b]A similar correction of the proof has been done independently by the authors of [9] themselves [10]. They give a construction for three stack OMPA generalizable to arbitrarily many stacks.

for bounded-context multi-stack pushdown automata is shown to be NP-complete in [18, 20].

in [26], La Torre, Madhusudan and Parlato propose the class of BVMPA as a strict generalization of the notion of a context. They show that runs of such automata can be represented as trees and the emptiness problem can be reduced to the emptiness problem for tree automata. In this paper, we show that OMPA are strictly more expressive than BVMPA (and, thus, than bounded-context multi-stack pushdown automata).

In [2], Atig proposes a uniform framework which can be used to prove the decidability of the emptiness problem for OMPA and BVMPA. This is done by showing that each of their emptiness problem can be reduced to the one for a class of single-stack machines.

In [19], Madhusudan, and Parlato show the benefits of studying the structure of runs of multi-pushdown systems as graphs. Such graphs, with linear edges to describe ordering of the events in the runs and nesting edges relating push events to corresponding pop events, called *multiply nested words*, generalize the nested word representation of runs of single pushdowns [1]. They show that the classes of multiply nested words arising from bounded context, bounded phase and ordered multi-stack pushdown automata have bounded tree-width (*tree-width* is a measure of the complexity of a graph). Using the decidability of the satisfiability problem of monadic second-order logic (MSO) on bounded tree-width graphs [13, 22], the authors show the decidability of the emptiness problem for BVMPA and OMPA.

Uniform approaches that also rely on tree-automata techniques and, moreover, are suitable for showing complementability of OMPA were given in [11, 16].

The techniques used in [2, 11, 16, 19] and the one proposed in this paper to prove the decidability of the emptiness problem for OMPA are quite different since we prove the decidability of OMPA by reducing it to a decidable class of grammars (following [9]).

A more general version of the control state reachability problem, called the global state reachability problem has also been studied. Given a regular set of configurations the aim is to construct a representation (usually again as a regular language) for the set of configurations from where the given set is reachable. Anil Seth [24] describes a 2EXPTIME decision procedure for the global state reachability for BVMPA systems while [3] solves the same problem for OMPA in 2EXPTIME. The techniques used in [3, 24] are also different from the ones used in this paper.

In [28], La Torre, Madhusudan and Parlato prove that the emptiness problem for BVMPA is 2ETIME-hard. Our proof of the 2ETIME-hardness is done by adapting the construction given in [28].

In [17, 29], La Torre and Napoli propose the concept of bounded scope runs, where each symbol in a stack can be popped only if it has been pushed within a given number of context switches. The authors show that the emptiness problem for multi-stack pushdown automata under scope-bounding is decidable (and PSPACE-complete). Moreover, they show that the class of multi-stack pushdown automata

under scope-bounding is incomparable with BVMPA. It is not clear how this decidability result for multi-stack pushdown automata under scope-bounding is related to the one we prove in this paper. Simulating scope-bounded computations using the order restriction on stacks does not seem to be possible. Moreover, the complexity of the emptiness problem for OMPA is clearly higher than for scope-bounded multi-stack pushdown automata.

Recently, it has been shown that multi-stack pushdown automata under scope-bounding have bounded tree-width [14,31]. In [14] a new measure of complexity for multiply nested words called split-width is proposed, which is bounded whenever tree-width is, but seems to yield easier proofs of boundedness.

In the setting of infinite runs, [3] describes a 2EXPTIME decision procedure for the repeated reachability and LTL model-checking problems for OMPA. The procedure described in [3] is based on the decision problem for the global state reachability problem for OMPA.

In [5], the authors investigate the existence of ultimately periodic infinite runs, i.e. runs of the form uv^ω, in multi-stack pushdown systems. They show that if we place a bound on the number of context switches permitted in u and v the resulting problem becomes decidable in EXPTIME.

Recently, [6, 30] showed that LTL model checking for multi-stack pushdown automata under scope-bounded can be solved in EXPTIME.

Finally, it should be noted that a restriction of OMPA has been defined that has single exponential complexity [7]. The idea is that pushs are only possible on the lowest non-empty stack or one of its neighbours.

Outline

The paper is structured as follows: In Section 2, we first introduce multi-stack pushdown automata (MPA) in their full generality. OMPA and BVMPA then arise as proper subclasses of MPA. After some first expressivity results on our basic models, we introduce, also in Section 2, depth-n-grammars. Sections 3 and 4 then establish the 2ETIME upper and, respectively, lower bound of the emptiness problem for OMPA, which constitutes our main result. In Section 5, we compare OMPA with BVMPA. We conclude by identifying some directions for future work.

An extended abstract of this paper appeared as [4].

2. Multi-Pushdown Automata and Depth-n-Grammars

Let us fix some basic notation. We call any non-empty finite set an *alphabet*. For an alphabet Σ, Σ^* is the set of finite words over Σ; the empty word is denoted by ε. The concatenation uv of words $u, v \in \Sigma^*$ is denoted by $u \cdot v$ (though we often simply write uv instead). For two natural numbers $n, m \in \mathbb{N}$ with $1 \leq m \leq n$, we write $[n]$ (resp. $[m..n]$) to denote the set $\{1, \ldots, n\}$ (resp. $\{m, \ldots, n\}$).

2.1. *Multi-Pushdown Automata and Subclasses*

In this section, we define *ordered multi-pushdown automata* and *bounded-phase multi-pushdown automata*. Both appear as a special case of *multi-pushdown automata*: they 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 $t = (q, A_1, \ldots, A_n) \xrightarrow{a} (q', \alpha_1, \ldots, \alpha_n)$. Being in a configuration $\langle p, aw; \gamma_1, \ldots, \gamma_n \rangle$, which is composed of a control state p, the word aw to be read, and a stack content γ_i for any stack i, t can be applied if both $p = q$ and the i-th stack is of the form $\gamma_i = A_i\gamma_i'$ for some γ_i'. Taking the transition and reading the a (which might be the empty word), the system moves to the successor configuration $\langle q', w; \alpha_1\gamma_1', \ldots, \alpha_n\gamma_n' \rangle$. I.e., the new control state is q', and A_i is replaced with α_i on the i-th stack.

Definition 1. *A* **multi-pushdown automaton** *(MPA) is a structure* $\mathcal{M} = (Q, n, \Sigma, \Gamma, \to, q_0, Z_0, F)$ *where*

- Q *is the finite non-empty set of* states*,*
- $n \geq 1$ *is the number of stacks,*
- Σ *is the finite set of* input symbols*,*
- Γ *is the finite set of* stack symbols *(disjoint from* $\Sigma \cup \{\bot\}$*),*
- $\to \subseteq \left[Q \times (\Gamma \cup \{\bot\} \cup \{\varepsilon\})^n\right] \times (\Sigma \cup \{\varepsilon\}) \times \left[Q \times (\Gamma^* \cup \Gamma^*\{\bot\})^n\right]$ *is the finite* transition relation *such that, for all* $((q, A_1, \ldots, A_n), a, (q', \alpha_1, \ldots, \alpha_n)) \in \to$ *and* $i \in [n]$*,* $A_i = \bot$ *iff* $\alpha_i = \alpha\bot$ *for some* $\alpha \in \Gamma^*$*,*
- q_0 *is the* initial state*,*
- $Z_0 \in \Gamma$ *is the* initial memory symbol*, and*
- $F \subseteq Q$ *is the set of* final states*.*

For the sake of readability, we may write $(q, A_1, \ldots, A_n) \xrightarrow{a} (q', \alpha_1, \ldots, \alpha_n)$ if $((q, A_1, \ldots, A_n), a, (q', \alpha_1, \ldots, \alpha_n)) \in \to$. A *stack content* of $\mathcal{M}$ is an element from $\mathit{Stack}_\mathcal{M} = \Gamma^*\{\bot\}$. The symbol $\bot$ marks the bottom of a stack. According to the transition relation, $\bot$ can never be popped. A *configuration* of $\mathcal{M}$ is an $(n+2)$-tuple $\langle q, w; \gamma_1, \ldots, \gamma_n \rangle$ with $q \in Q$, $w \in \Sigma^*$, and $\gamma_1, \ldots, \gamma_n \in \mathit{Stack}_\mathcal{M}$. The set of configurations is denoted by $\mathit{Conf}_\mathcal{M}$. The behavior of $\mathcal{M}$ is described by its *step relation*. For a transition $t = (q, A_1, \ldots, A_n) \xrightarrow{a} (q', \alpha_1, \ldots, \alpha_n)$, we define $\vdash_\mathcal{M}^t \subseteq \mathit{Conf}_\mathcal{M} \times \mathit{Conf}_\mathcal{M}$ to be the least set satisfying the following: if $\gamma_1, \ldots, \gamma_n \in \mathit{Stack}_\mathcal{M} \cup \{\varepsilon\}$ such that $A_i\gamma_i \in \mathit{Stack}_\mathcal{M}$ for all $i \in [n]$, then $\langle q, aw; A_1\gamma_1, \ldots, A_n\gamma_n \rangle \vdash_\mathcal{M}^{(t)} \langle q, w; \alpha_1\gamma_1, \ldots, \alpha_n\gamma_n \rangle$. Let

$$\vdash_\mathcal{M} = \bigcup_{t \in \to} \vdash_\mathcal{M}^{(t)}$$

A configuration $\langle q, w; \gamma_1, \ldots, \gamma_n \rangle$ is *final* if $w = \varepsilon$ and $q \in F$. It is *empty* if $w = \varepsilon$ and $\gamma_1 = \ldots = \gamma_n = \bot$. We may declare a configuration accepting if it is final or

$M = (\{q_0, \ldots, q_3, q_f\}, 2, \{a, b, c\}, \{A, B, Z_0, Z_1\}, \to, q_0, Z_0, \{q_f\})$		
$(q_0, Z_0, \varepsilon) \xrightarrow{\varepsilon} (q_f, \varepsilon, \varepsilon)$	$(q_1, A, \varepsilon) \xrightarrow{a} (q_1, AA, B)$	$(q_3, \bot, Z_1) \xrightarrow{\varepsilon} (q_0, Z_0\bot, \varepsilon)$
$(q_0, Z_0, \varepsilon) \xrightarrow{a} (q_1, AZ_0, BZ_1)$	$(q_2, b, A) \xrightarrow{b} (q_2, \varepsilon, \varepsilon)$	$(q_3, \bot, B) \xrightarrow{c} (q_3, \bot, \varepsilon)$
$(q_1, A, \varepsilon) \xrightarrow{\varepsilon} (q_2, A, \varepsilon)$	$(q_2, Z_0, \varepsilon) \xrightarrow{\varepsilon} (q_3, \varepsilon, \varepsilon)$	

Table 1. A 2-OMPA for $\{a^n b^n c^n \mid n \geq 0\}^*$

empty. For $\mathcal{M}$, we therefore define two possible semantics:

$$L(\mathcal{M}) = \{w \in \Sigma^* \mid \langle q_0, w; Z_0\bot, \bot, \ldots, \bot\rangle \, (\vdash_{\mathcal{M}})^* \, c \text{ for some final configuration } c\}$$
$$L_\bot(\mathcal{M}) = \{w \in \Sigma^* \mid \langle q_0, w; Z_0\bot, \bot, \ldots, \bot\rangle \, (\vdash_{\mathcal{M}})^* \, c \text{ for some empty configuration } c\}$$

We now introduce our main automata model, in which one can pop only from the first non-empty stack (i.e., all preceding stacks equal $\bot$).

Definition 2. *An* **ordered multi-pushdown automaton** *(OMPA) is an MPA* $(Q, n, \Sigma, \Gamma, \to, q_0, Z_0, F)$ *where, for each transition* $(q, A_1, \ldots, A_n) \xrightarrow{a} (q', \alpha_1, \ldots, \alpha_n)$*, there is* $j \in \{1, \ldots, n\}$ *such that* $A_1 = \ldots = A_{j-1} = \bot$*,* $A_j \in \Gamma \cup \{\bot\} \cup \{\varepsilon\}$*, and* $A_{j+1} = \ldots = A_n = \varepsilon$*.*

Table 1 shows an example of a 2-OMPA accepting the language $\{a^n b^n c^n \mid n \geq 0\}^*$. Notice that it accepts the same language by final state and by empty stacks.

Next, let us define visibly multi-pushdown automata [26], another subclass of MPA, where an action is associated with a particular stack operation. An action can be a push, pop, or internal action. In our general setting, these automata can be defined as follows:

Definitions 3–5 are only needed in Section 5. They can be skipped for the moment and may be consulted for later reference.

Definition 3. *A* **visibly multi-pushdown automaton** *(VMPA) is an MPA* $(Q, n, \Sigma, \Gamma, \to, q_0, Z_0, F)$ *such that there is a mapping type* $: \Sigma \to (\{\text{push}, \text{pop}\} \times [n]) \cup \{\text{int}\}$ *satisfying, for all transitions* $(q, A_1, \ldots, A_n) \xrightarrow{a} (q', \alpha_1, \ldots, \alpha_n)$*:*

- $a \neq \varepsilon$*,*
- *if* $type(a) = (\text{push}, i)$ *for some* $i \in [n]$*, then* $A_1 = \ldots = A_n = \varepsilon$*,* $\alpha_i \in \Gamma$*, and* $\alpha_j = \varepsilon$ *for all* $j \in [n] \setminus \{i\}$*,*
- *if* $type(a) = (\text{pop}, i)$ *for some* $i \in [n]$*, then* $A_i \in \Gamma \cup \{\bot\}$*,* $\alpha_i \in \{\bot\} \cup \{\varepsilon\}$*, and* $A_j = \alpha_j = \varepsilon$ *for all* $j \in [n] \setminus \{i\}$*, and*
- *if* $type(a) = \text{int}$*, then* $A_i = \alpha_i = \varepsilon$ *for all* $i \in [n]$*.*

If, in an MPA, we restrict the number of phases, where in one phase pop operations are exclusive to one stack, then we obtain bounded-phase multi-pushdown automata.

Definition 4. *A* **bounded-phase multi-pushdown automaton** *(BMPA) is a pair* $\mathcal{B} = (\mathcal{M}, k)$ *where* $k \geq 1$ *and* $\mathcal{M}$ *is an MPA* $(Q, n, \Sigma, \Gamma, \to, q_0, F)$ *such that, for each transition* $(q, A_1, \ldots, A_n) \xrightarrow{a} (q', \alpha_1, \ldots, \alpha_n)$*, we have* $A_1 \cdot \ldots \cdot A_n \in \Gamma \cup \{\bot\} \cup \{\varepsilon\}$.

For defining the language of $\mathcal{B}$, we require the notion of a phase. To this aim, we identify those transitions that correspond to a read/pop operation on a given stack. A transition $(q, A_1, \ldots, A_n) \xrightarrow{a} (q', \alpha_1, \ldots, \alpha_n)$ is a *pop transition on stack* $i \in [n]$ if $A_i \in \Gamma \cup \{\bot\}$. We collect in $Ph(i)$ the transitions $t \in \to$ such that, for all $j \in [n]$, the following holds: if t is a pop transition on stack j, then $j = i$. We define

$$\vdash_{\mathcal{B}}^{(i)} = \bigcup_{t \in Ph(i)} \left(\vdash_{\mathcal{M}}^{(t)}\right)^* \text{ and } \vdash_{\mathcal{B}} = \bigcup_{i \in [n]} \vdash_{\mathcal{B}}^{(i)}$$

Finally, the language of $\mathcal{B}$ is

$$L(\mathcal{B}) = \{w \in \Sigma^* \mid \langle q_0, w; Z_0\bot, \bot, \ldots, \bot\rangle \, (\vdash_{\mathcal{B}})^k \, c \text{ for some final configuration } c\}$$

Combining VMPA and BMPA, we obtain bounded-phase visibly MPA:

Definition 5. *A* bounded-phase visibly multi-pushdown automaton *(BVMPA) is a BMPA* $(\mathcal{M}, k)$ *such that* $\mathcal{M}$ *is a VMPA.*

We introduce some abbreviations: For a natural number $n \geq 1$, we call an MPA/OMPA/VMPA/BMPA/BVMPA an n-MPA/n-OMPA/n-VMPA/n-BMPA/n-BVMPA, respectively, if its number of stacks is n. A k-phase (n-)BMPA is a BMPA of the form $(\mathcal{M}, k)$ (with $\mathcal{M}$ an n-MPA). An MPA *over* Σ is an MPA with input alphabet Σ.

2.2. *First results*

Lemma 6 ([9]) *Let* Σ *be an alphabet,* $n \geq 1$*, and* $L \subseteq \Sigma^*$*. The following statements are equivalent:*

- *There is an* n*-MPA* $\mathcal{M}$ *over* Σ *such that* $L(\mathcal{M}) = L$*.*
- *There is an* n*-MPA* $\mathcal{M}'$ *over* Σ *such that* $L_\bot(\mathcal{M}') = L$*.*

We need the following normal form of n-OMPA for the proof of our main theorem. The normal form restricts the operation on stacks 2 to n: pushing one symbol on these stacks is only allowed while popping a symbol from the first stack, and popping a symbol from them pushes a symbol onto the first stack. Furthermore, the number of symbols pushed on the first stack is limited to two.

Definition 7. *An* n*-OMPA* $(Q, n, \Sigma, \Gamma, \to, q_0, Z_0, F)$ *with* $n \geq 2$ *is in normal form if* $\to$ *contains only the following types of transitions:*

- $(q, A, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q', BC, \varepsilon, \ldots, \varepsilon)$ *for some* $A, B, C \in \Gamma$ *and* $a \in \Sigma \cup \{\varepsilon\}$
- $(q, A, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q', \varepsilon, \ldots, \varepsilon, B, \varepsilon, \ldots, \varepsilon)$ *for some* $A, B \in \Gamma$ *and* $a \in \Sigma \cup \{\varepsilon\}$ *(*B *is pushed on one of stacks* 2 *to* n*)*

- $(q, \bot, \ldots, \bot, A, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q', B\bot, \bot, \ldots, \bot, \varepsilon, \varepsilon, \ldots, \varepsilon)$ *for some* $A, B \in \Gamma$ *and* $a \in \Sigma \cup \{\varepsilon\}$ *(*A *is popped from one of the stacks* 2 *to* n*)*
- $(q, A, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q', \varepsilon, \ldots, \varepsilon)$ *for some* $A \in \Gamma$ *and* $a \in \Sigma \cup \{\varepsilon\}$

Lemma 8. *An* n*-OMPA* M *can be transformed into an* n*-OMPA* M' *in normal form with linear blowup in its size such that* $L(M) = L(M')$*.*

Proof. An easy generalization of the proof for the Chomsky normal form for context-free grammars. □

Next, we recall some properties of the class of languages recognized by n-OMPA and n-BVMPA. We start by defining a renaming operation: A *renaming* of Σ to Σ' is a function $f : \Sigma \to \Sigma'$. It is extended to strings and languages in the natural way: $f(a_1 \ldots a_k) = f(a_1) \cdot \ldots \cdot f(a_k)$ and $f(L) = \bigcup_{w \in L} f(w)$.

Lemma 9 ([9]) *(Closure Properties of OMPA) The class of languages recognized by* n*-OMPA is closed under union, renaming and Kleene-star.*

Lemma 10 ([26]) *(Closure Properties of BVMPA) The class of languages recognized by* n*-BVMPA is closed under union and renaming; but not under Kleene-star.*

2.3. *Depth-n-grammars*

We now define the notion of a depth-n-grammar. Let V_N and V_T be finite disjoint alphabets and let "(" and "$)_i$" for $i \in \{1, \ldots, n\}$ be $n+1$ characters not in $V_N \cup V_T$. An *n-list* is a finite string of the form $\overline{\alpha} = w(\alpha_1)_1(\alpha_2)_2 \ldots (\alpha_n)_n$ where $w \in V_T^*$ and $\alpha_i \in V_N^*$ for all i with $1 \leq i \leq n$.

Definition 11. *A depth-*n*-grammar (*D^n*-grammar) is a tuple* $G = (V_N, V_T, P, S)$ *where* V_N *and* V_T *are the finite disjoint sets of* non-terminal *and* terminal symbols*, respectively,* $S \in V_N$ *is the axiom, and* P *is a finite set of* productions *of the form* $A \to \overline{\alpha}$ *with* $A \in V_N$ *and* $\overline{\alpha}$ *an* n*-list.*

For clarity, we may drop empty components of n-lists in the productions as follows: $A \to w(\varepsilon)_1 \ldots (\varepsilon)_n$ is written as $A \to w$, $A \to (\varepsilon)_1 \ldots (\varepsilon)_n$ is written as $A \to \varepsilon$, and $A \to w(\varepsilon)_1 \ldots (\varepsilon)_{i-1}(\alpha_i)_i(\varepsilon)_{i+1} \ldots (\varepsilon)_n$ is written as $A \to w(\alpha_i)_i$.

We define the *derivation relation* on n-lists as follows. Let $i \in \{1, \ldots, n\}$ and let $\overline{\beta} = (\varepsilon)_1 \ldots (\varepsilon)_{i-1}(A\beta_i)_i(\beta_{i+1})_{i+1} \ldots (\beta_n)_n$ be an n-list, where $\beta_j \in V_N^*$ for all $j \in \{i, \ldots, n\}$. Then,

$$x\overline{\beta} \Rightarrow xw(\alpha_1)_1(\alpha_2)_2 \ldots (\alpha_{i-1})_{i-1}(\alpha_i\beta_i)_i(\alpha_{i+1})_{i+1} \ldots (\alpha_n\beta_n)_n$$

if $A \to w(\alpha_1)_1(\alpha_2)_2 \ldots (\alpha_n)_n$ is a production and $x \in V_T^*$. Notice that only leftmost derivations are defined. As usual we denote by $\Rightarrow^*$ the reflexive and transitive closure of $\Rightarrow$. A terminal string $x \in V_T^*$ is *derivable* from S if $(S)_1(\varepsilon)_2 \ldots (\varepsilon)_n \Rightarrow^* x(\varepsilon)_1 \ldots (\varepsilon)_n$. This will be also denoted by $S \Rightarrow^* x$. The language generated by a D^n-grammar G is $L(G) = \{x \in V_T^* \mid S \Rightarrow^* x\}$.

For example the language $\{a^n b^n c^n \mid n \geq 0\}$ is generated by the depth-2 grammar

$$G^2 = (\{S, B, C\}, \{a, b, c\}, \{S \to \varepsilon, S \to a(SB)_1(C)_2, B \to b, C \to c\}, S)$$

Definition 12. *Let $G = (V_N, V_T, P, S)$ be a D^n-grammar. Then, the underlying context-free grammar is $G_{cf} = (V_N, V_T, P_{cf}, S)$ with $P_{cf} = \{A \to w\alpha_1 \dots \alpha_n \mid A \to w(\alpha_1)_1 \dots (\alpha_n)_n \in P\}$.*

For example, for G^2 we have

$$G^2_{cf} = (\{S, B, C\}, \{a, b, c\}, \{S \to \varepsilon, S \to aSBC, B \to b, C \to c\}, S)$$

Notice that the language generated by G^2_{cf} is $\{a^n(bc)^n \mid n \geq 0\}$.

The following lemma from [9] is obtained by observing that the language generated by a D^n-grammar is empty iff the language generated by its underlying context-free grammar G_{cf} is empty. Furthermore, it is well-known that, on Turing machines, emptiness of context-free grammars can be decided in time linear in its size.

Lemma 13. *The emptiness problem of D^n-grammars is decidable in linear time.*

3. Emptiness of OMPA is in 2ETIME

Here, we show that the emptiness problem of OMPA is in 2ETIME. First, we show that n-OMPA correspond to D^n-grammars with a double exponential number of non-terminal symbols by correcting the construction of [9]. Then, emptiness of D^n-grammars is decidable using the underlying context-free grammar (Lemma 13).

Theorem 14. *A language L is accepted by an n-OMPA iff it is generated by a D^n-grammar.*

The rest of the section is devoted to the proof of this theorem. The "if"-direction is obvious, as a grammar is just an automaton with one state. For the other direction, let L be a language accepted by empty stacks by an n-OMPA $M = (Q, n, \Sigma, \Gamma, \to, q_0, Z_0, F)$. We assume without loss of generality (Lemma 8), that M is in normal form. We construct a D^n-grammar $G_M = (V_N, \Sigma, P, S)$ such that $L(G_M) = L_\perp(M) = L$.

Intuitively, we generalize the proof for the case of 2-OMPA [21]. In [9], an incorrect proof was given for the case of n-OMPA. Please note that the authors of [9] independently gave a generalizable proof for 3-OMPA, which is similar to ours [10]. The general proof idea is the same as for the corresponding proof for pushdown automata. To eliminate states, one has to guess the sequence of states through which the automaton goes by adding pairs of state symbols to the non-terminal symbols of the corresponding grammar. We do this for the first stack. However, when the first stack gets empty, the other stacks may be not empty and one has to know the state in which the automaton is in this situation. For this, we have to guess for all

the other non-empty stacks and each of their stack symbols the state in which the automaton will be when reading these symbols.[c]

To do this for the n-th stack, a pair of state symbols is enough. For the $(n-1)$-th stack, in addition to guessing the state, we also have to know the current state on top of the n-th stack to be able to push correctly symbols onto the n-th stack. Therefore, a pair of pairs of states (4 in total) is needed. For the $(n-2)$-th stack, we need to remember the current state and the states on top of the $(n-1)$-th stack and on top of the n-th stack (in total 8 states) and so on. Therefore, there will be 2^n state symbols to be guessed in the first stack. Furthermore we have special state symbols (denoted q_i^e) to indicate that the i-th stack is empty (contains $\perp$). In Fig. 1 we give an intuitive example illustrating the construction. In the following we will use the term *stack* for the stacks in the OMPA and for the lists in the corresponding grammar.

Now we define the grammar $G_M = (V_N, \Sigma, P, S)$ formally. To define V_N, we first provide symbols of level i denoted by V_i. For i with $2 \leq i \leq n$, let q_i^e be states pairwise different and different from any state of Q (these symbols indicate that the corresponding stack is empty). States of level i are denoted by Q_i and defined as : $Q_n = Q \cup \{q_n^e\}$ and for all i such that $2 \leq i < n$, $Q_i = (Q \times Q_{i+1} \times \cdots \times Q_n) \cup \{q_i^e\}$, and $Q_1 = Q \times Q_2 \times \cdots \times Q_n$. We denote by $\vec{q}_i$ states of Q_i. Then, $V_i = Q_i \times \Gamma \times Q_i$ and $V_N = \{S\} \cup \bigcup_{i=1}^{n} V_i$. Notice that a state in Q_i different from q_i^e has up to 2^{n-i} components. Therefore $|V_N| \leq (|Q|+1)^{2^{n+1}}|\Gamma|$. The set P contains exactly the following productions partitioned into five types ($a \in \Sigma \cup \{\varepsilon\}$):

T1 $S \to ([(q_0, q_2^e, \ldots, q_n^e), Z_0, (q^1, \vec{q}_2^{\,1}, \ldots, \vec{q}_n^{\,1})])_1$
if there is k with $2 \leq k \leq n+1$ such that
- for all i with $2 \leq i < k$ we have $\vec{q}_i^{\,1} = q_i^e$
- if $k \leq n$, then $\vec{q}_k^{\,1} = (q^1, \vec{q}_{k+1}^{\,1}, \ldots, \vec{q}_n^{\,1})$

T2 $[(q^1, \vec{q}_2^{\,1}, \ldots, \vec{q}_n^{\,1}), A, \vec{q}_1^{\,2}] \to a([(q^4, \vec{q}_2^{\,1}, \ldots, \vec{q}_n^{\,1}), B, \vec{q}_1^{\,3}][\vec{q}_1^{\,3}, C, \vec{q}_1^{\,2}])_1$
if $(q^1, A, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q^4, BC, \varepsilon, \ldots, \varepsilon)$

T3 $[(q^1, \vec{q}_2^{\,1}, \ldots, \vec{q}_{j-1}^{\,1}, \vec{q}_j^{\,1}, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), A, (q^2, \vec{q}_2^{\,1}, \ldots, \vec{q}_{j-1}^{\,1}, \vec{q}_j^{\,2}, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1})]$
$\to a([\vec{q}_j^{\,2}, B, \vec{q}_j^{\,1}])_j$ if $\vec{q}_j^{\,2} \neq q_j^e$ and $(q^1, A, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q^2, \varepsilon, \ldots, \varepsilon, B, \varepsilon, \ldots, \varepsilon)$
and B is pushed on stack j for $2 \leq j \leq n$.

T4 $[(q^1, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), A, \vec{q}_j^{\,1}]$
$\to a([(q^4, q_2^e, \ldots, q_{j-1}^e, \vec{q}_j^{\,1}, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), B, (q^2, \vec{q}_2^{\,2}, \ldots, \vec{q}_n^{\,2})])_1$
if $(q^1, \perp, \ldots, \perp, A, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q^4, B\perp, \perp, \ldots, \perp, \varepsilon, \varepsilon, \ldots, \varepsilon)$, A is popped from stack j with $2 \leq j \leq n$ and there is k with $2 \leq k \leq n+1$ such that
- for all i with $2 \leq i < min(k, j)$ we have $\vec{q}_i^{\,2} = q_i^e$
- for all i with $min(k, j) \leq i < k$ we have $\vec{q}_i^{\,1} = \vec{q}_i^{\,2} = q_i^e$
- if $k \leq n$, then $\vec{q}_k^{\,2} = (q^2, \vec{q}_{k+1}^{\,2}, \ldots, \vec{q}_n^{\,2})$

[c]The proof in [9] incorrectly assumes that this state is the same for each stack when the first stack gets empty.

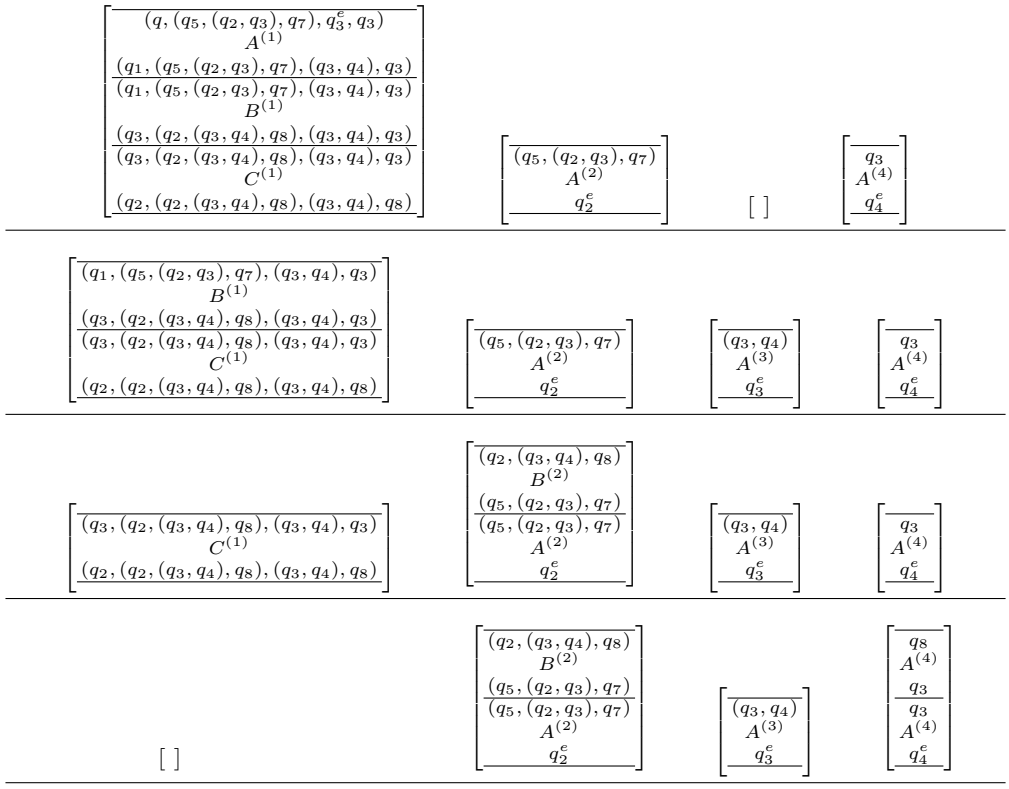

Fig. 1. A sketch of a partial derivation (from top to bottom) of a depth-4-grammar which corresponds to the partial run of a 4-OMPA M of the form $\langle q, \varepsilon; A^{(1)}B^{[1]}C^{(1)}, A^{(2)}, \varepsilon, A^{(4)}\rangle \vdash_M \langle q_1, \varepsilon; B^{(1)}C^{(1)}, A^{(2)}, A^{(3)}, A^{(4)}\rangle \vdash_M \langle q_3, \varepsilon; C^{(1)}, B^{(2)}A^{(2)}, A^{(3)}, A^{(4)}\rangle \vdash_M \langle q_2, \varepsilon; \varepsilon, B^{(2)}A^{(2)}, A^{(3)}, A^{(4)}A^{(4)}\rangle \vdash_M^* \langle q_3, \varepsilon; \varepsilon, A^{(3)}, A^{(4)}A^{(4)}A^{(4)}\rangle \vdash_M^* \langle q_4, \varepsilon; \varepsilon, \varepsilon, A^{(4)}A^{(4)}A^{(4)}\rangle \vdash_M^* \langle q_8, \varepsilon; \varepsilon, \varepsilon, A^{(4)}A^{(4)}\rangle \vdash_M^* \langle q_3, \varepsilon; \varepsilon, \varepsilon, A^{(4)}\rangle \vdash_M^* \langle q_4, \varepsilon; \varepsilon, \varepsilon, \varepsilon\rangle$, where in the first three steps three symbols are popped from the first stack while three symbols are pushed onto the other stacks. In each configuration, if the first stack is non-empty, then the state symbols on top of the other stacks can be found on top of the first stack as well. In the last configuration, the top symbols of the other stacks can be found on top of the second stack. Here, the meaning of $(q_2, (q_3, q_4), q_8)$ is that the automaton is currently in state q_2 and that on top of stack 3 there is (q_3, q_4) and on top of stack 4 there is q_8. (q_3, q_4) means that once the automaton gets to this point (the corresponding symbol is on top of the stack and the first two stacks are empty) it is in state q_3 and on top of stack 4 there is q_4 which in turn means that q_4 is the state of the automaton once it arrives at this point (the first three stacks are empty). q_8 is the state of the automaton once it reaches the corresponding point. As in the classical simulation of pushdown-automata by context-free grammars, two adjacent state symbols are always equal.

$$\text{T5}\ \ [(q^1, \vec{q}^1_2, \ldots, \vec{q}^1_n), A, (q^2, \vec{q}^1_2, \ldots, \vec{q}^1_n)] \to a \text{ if } (q^1, A, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q^2, \varepsilon, \ldots, \varepsilon)$$

The intuition behind the productions is the following. T1 corresponds to the initial productions of the grammar. Here k corresponds to the (guessed) first non-empty stack reached after the first stack is emptied. As explained before and formalised in Lemma 16 it is important that the (guessed) top-most state of level k

appearing on stack k correponds to the (guessed) states on top of the stacks $> k$. T2 corresponds to the productions simulating the transitions of the n-OMPA M changing an A to BC in the first stack. In this case, an intermediate state of level 1, namely $\vec{q}_1^3$, is guessed. T3 corresponds to the productions simulating transitions of M pushing B into stack j while popping A from the first stack. This is only possible, if the guessed states match. T4 corresponds to the productions simulating transitions of M popping an A from the first non-empty stack and pushing a B on top of the first stack. Here, k corresponds to the (guessed) first non empty stack after B will be popped. T5 corresponds to the productions simulating transitions of M where an A is being popped from the first stack. This is only possible if the guessed states match.

Example 15. *Here we give the corresponding grammar for the 2-OMPA of Table 1. Notice that the language accepted by empty stacks is the same as the one accepted by final state. First we bring M into normal form:*

$M = (\{q_0, \ldots, q_3, q_f\}, n, \{a, b, c\}, \{A, B, C, \ldots, I, J, Z_0, Z_1\}, \rightarrow, q_0, Z_0, \{q_f\})$

$(q_0, Z_0, \varepsilon) \xrightarrow{\varepsilon} (q_f, \varepsilon, \varepsilon)$	$(q_1, A, \varepsilon) \xrightarrow{\varepsilon} (q_2, GA, \varepsilon)$	$(q_2, A, \varepsilon) \xrightarrow{b} (q_2, \varepsilon, \varepsilon)$
$(q_0, Z_0, \varepsilon) \xrightarrow{a} (q_1, CZ_0, \varepsilon)$	$(q_2, G, \varepsilon) \xrightarrow{\varepsilon} (q_2, \varepsilon, \varepsilon)$	$(q_2, Z_0, \varepsilon) \xrightarrow{\varepsilon} (q_3, \varepsilon, \varepsilon)$
$(q_1, C, \varepsilon) \xrightarrow{\varepsilon} (q_1, DA, \varepsilon)$	$(q_1, A, \varepsilon) \xrightarrow{a} (q_1, HA, \varepsilon)$	$(q_3, \perp, Z_1) \xrightarrow{\varepsilon} (q_0, Z_0\perp, \varepsilon)$
$(q_1, D, \varepsilon) \xrightarrow{\varepsilon} (q_1, EF, \varepsilon)$	$(q_1, H, \varepsilon) \xrightarrow{\varepsilon} (q_1, IA, \varepsilon)$	$(q_3, \perp, B) \xrightarrow{\varepsilon} (q_3, J\perp, \varepsilon)$
$(q_1, E, \varepsilon) \xrightarrow{\varepsilon} (q_1, \varepsilon, Z_1)$	$(q_1, I, \varepsilon) \xrightarrow{\varepsilon} (q_1, \varepsilon, B)$	$(q_3, J, \varepsilon) \xrightarrow{c} (q_3, \varepsilon, \varepsilon)$
$(q_1, F, \varepsilon) \xrightarrow{\varepsilon} (q_1, \varepsilon, B)$		

The corresponding grammar G_M is given in Table 2. We only give the productions containing productive symbols.

The following key lemma formalizes the intuition about derivations of the grammar G_M by giving invariants satisfied by them (illustrated in Fig. 1). It is the basic ingredient of the proof of Theorem 14. Intuitively, condition 1 says that the first element of the first stack contains the state symbols on top of the other stacks. Condition 3 says that if the first stack is empty, then the top of the first non-empty stack contains the same state symbols as the top of the other stacks. Condition 2 says that the last state symbols in the first stack are of the form allowing condition 3 to be true when the corresponding symbol is popped. Conditions 4 and 5 say that the state symbols guessed form a chain through the stacks. In the following, to avoid confusion we write $A^{(j)}$ to indicate that the non-terminal symbol appears in the j-th list (stack).

Lemma 16. *Let $w(\gamma_1)(\gamma_2)\ldots(\gamma_n)$ be an n-list different from $(\varepsilon)_1\ldots(\varepsilon)_n$ appearing in a derivation of the grammar G_M.*

(1) If $\gamma_1 = [(q^1, \vec{q}_2^1, \ldots, \vec{q}_n^1), A^{(1)}, (q^2, \vec{q}_2^2, \ldots, \vec{q}_n^2)]\gamma_1'$ with $\gamma_1' \in V_1^*$, then for all i with $2 \leq i \leq n$, if γ_i is empty, then $\vec{q}_i^1 = q_i^e$, else $\gamma_i = [\vec{q}_i^1, B^{(i)}, \vec{q}_i^3]\gamma_i'$ with

S	$\rightarrow$	$([(q_0, q_2^e), Z_0, (q_f, q_2^e)])_1 \ + \ ([(q_0, q_2^e), Z_0, (q_3, q_3)])_1$
$[(q_0, q_2^e), Z_0, (q_f, q_2^e)]$	$\rightarrow$	ε
$[(q_0, q_2^e), Z_0, (q_3, q_3)]$	$\rightarrow$	$a([(q_1, q_2^e), C, (q_2, q_3)][(q_2, q_3), Z_0, (q_3, q_3)])_1$
$[(q_1, q_2^e), C, (q_2, q_3)]$	$\rightarrow$	$([(q_1, q_2^e), D, (q_1, q_3)][(q_1, q_3), A, (q_2, q_3)])_1$
$[(q_1, q_2^e), D, (q_1, q_3)]$	$\rightarrow$	$([(q_1, q_2^e), E, (q_1, q_3)][(q_1, q_3), F, (q_1, q_3)])_1$
$[(q_1, q_2^e), E, (q_1, q_3)]$	$\rightarrow$	$([q_3, Z_1, q_2^e])_2$
$[(q_1, q_3), F, (q_1, q_3)]$	$\rightarrow$	$([q_3, B, q_3])_2$
$[(q_1, q_3), A, (q_2, q_3)]$	$\rightarrow$	$([(q_2, q_3), G, (q_2, q_3)][(q_2, q_3), A, (q_2, q_3)])_1$
$[(q_2, q_3), G, (q_2, q_3)]$	$\rightarrow$	ε
$[(q_1, q_3), A, (q_2, q_3)]$	$\rightarrow$	$a([(q_1, q_3), H, (q_2, q_3)][(q_2, q_3), A, (q_2, q_3)])_1$
$[(q_1, q_3), H, (q_2, q_3)]$	$\rightarrow$	$([(q_1, q_3), I, (q_1, q_3)][(q_1, q_3), A, (q_2, q_3)])_1$
$[(q_1, q_3), I, (q_1, q_3)]$	$\rightarrow$	$([q_3, B, q_3])_2$
$[(q_2, q_3), A, (q_2, q_3)]$	$\rightarrow$	b
$[(q_2, q_3), Z_0, (q_3, q_3)]$	$\rightarrow$	ε
$[q_3, Z_1, q_2^e]$	$\rightarrow$	$([(q_0, q_2^e), Z_0, (q_f, q_2^e)])_1 \ + \ ([(q_0, q_2^e), Z_0, (q_3, q_3)])_1$
$[q_3, B, q_3]$	$\rightarrow$	$([(q_3, q_3), J, (q_3, q_3)])_1$
$[(q_3, q_3), J, (q_3, q_3)]$	$\rightarrow$	c

Table 2. The grammar G_M

$\gamma_i' \in V_i^*$.

(2) *If* $\gamma_1 = \gamma_1'[(q^1, \vec{q}_2^1, \ldots, \vec{q}_n^1), A^{(1)}, (q^3, \vec{q}_2^3, \ldots, \vec{q}_n^3)]$ *with* $\gamma_1' \in V_1^*$, *then there exists* k *with* $2 \leq k \leq n+1$ *such that we have both for all* i *with* $2 \leq i < k$, $\vec{q}_i^3 = q_i^e$ *and* $k \leq n$ *implies* $\vec{q}_k^3 = (q^3, \vec{q}_{k+1}^3, \ldots, \vec{q}_n^3)$.

(3) *Suppose that* $\gamma_1 = \varepsilon$. *Let* i *be the smallest* k *such that* γ_k *is not empty and let* $\gamma_i = [(q^1, \vec{q}_{i+1}^1, \ldots, \vec{q}_n^1), A^{(i)}, \vec{q}_i^2]\gamma_i'$ *with* $\gamma_i' \in V_i^*$. *Then, for all* $j > i$, *we have: if* γ_j *is empty, then* $\vec{q}_j^1 = q_j^e$, *else* $\gamma_j = [\vec{q}_j^1, A^{(j)}, \vec{q}_j^3]\gamma_j'$ *with* $\gamma_j' \in V_j^*$.

(4) *For all* i *with* $2 \leq i \leq n$, *if* γ_i *is not empty then for some* $j \geq 1$, $\gamma_i = [\vec{q}_i^1, A_1^{(i)}, \vec{q}_i^2][\vec{q}_i^2, A_2^{(i)}, \vec{q}_i^3] \ldots [\vec{q}_i^{j-1}, A_{j-1}^{(i)}, \vec{q}_i^j][\vec{q}_i^j, A_j^{(i)}, q_i^e]$ *and for all* l *with* $1 \leq l \leq j$, $\vec{q}_i^l \neq q_i^e$.

(5) *If* γ_1 *is not empty, then for some* $j \geq 1$, $\gamma_1 = [\vec{q}_1^1, A_1^{(1)}, \vec{q}_1^2][\vec{q}_1^2, A_2^{(1)}, \vec{q}_1^3] \ldots [\vec{q}_1^{j-1}, A_{j-1}^{(1)}, \vec{q}_1^j][\vec{q}_1^j, A_j^{(1)}, \vec{q}_1^{j+1}]$.

Proof. We prove the five conditions simultaneously by induction.
Base case: Starting from S the type 1 rules produce n-lists of the form $([(q_0, q_2^e, \ldots, q_n^e), Z_0, (q^1, \vec{q}_2^1, \ldots, \vec{q}_n^1)])_1(\varepsilon)_2 \ldots (\varepsilon)_n$ for a k with $2 \leq k \leq n+1$ and

- for all i with $2 \leq i < k$ we have $\vec{q}_i^1 = q_i^e$
- if $k \leq n$, then $\vec{q}_k^1 = (q^1, \vec{q}_{k+1}^1, \ldots, \vec{q}_n^1)$

The n-lists satisfy the 5 conditions (the third and forth condition are trivially satisfied, the others are satisfied by construction).
Inductive step:

Let $\overline{\gamma} = (\gamma_1)_1(\gamma_2)_2 \dots (\gamma_n)_n$ be an n-list satisfying the five conditions. We show that any n-list $\overline{\gamma}' = (\gamma'_1)_1(\gamma'_2)_2 \dots (\gamma'_n)_n$ such that $(\gamma_1)_1(\gamma_2)_2 \dots (\gamma_n)_n \Rightarrow w(\gamma'_1)_1(\gamma'_2)_2 \dots (\gamma'_n)_n$ satisfies the five conditions. We use a case split on the type of rule applied.

- A rule of type 2 is applied. It is of the form $[(q^1, \vec{q}^1_2, \dots, \vec{q}^1_n), A^{(1)}, \vec{q}^2_1] \to a([(q^4, \vec{q}^1_2, \dots, \vec{q}^1_n), B^{(1)}, \vec{q}^3_1][\vec{q}^3_1, C^{(1)}, \vec{q}^2_1])_1$. Then, γ_1 must be of the form $[(q^1, \vec{q}^1_2, \dots, \vec{q}^1_n), A^{(1)}, \vec{q}^2_1]\gamma''_1$. Then $\gamma'_1 = [(q^4, \vec{q}^1_2, \dots, \vec{q}^1_n), B^{(1)}, \vec{q}^3_1][\vec{q}^3_1, C^{(1)}, \vec{q}^2_1]\gamma''_1$.

  For condition 1 we have to show that for all i with $2 \leq i \leq n$, if γ'_i is empty, then $\vec{q}^1_i = q^e_i$, else $\gamma'_i = [\vec{q}^1_i, B^{(i)}, \vec{q}^3_i]\gamma''_i$ with $\gamma''_i \in V_i^*$. If γ'_i is empty, then γ_i is empty as well and by induction hypothesis $\vec{q}^1_i = q^e_i$. If γ'_i is not empty, then by induction hypothesis we know that $\gamma_i = [\vec{q}^1_i, B^{(i)}, \vec{q}^3_i]\gamma''_i$ with $\gamma''_i \in V_i^*$. Since rules of type 2 do not modify γ_i we have $\gamma'_i = [\vec{q}^1_i, B^{(i)}, \vec{q}^3_i]\gamma''_i$ and therefore condition 1 is satisfied for $\overline{\gamma}'$.

  Condition 2 concerns the rightmost element of γ'_1. There are two cases: $|\gamma_1| = 1$ or $|\gamma_1| > 1$. In the latter case the rightmost element of γ'_1 does not change and condition 2 remains true. In the former case, we have $\gamma_1 = [(q^1, \vec{q}^1_2, \dots, \vec{q}^1_n), A^{(1)}, \vec{q}^2_1]$ and $\gamma'_1 = [(q^4, \vec{q}^1_2, \dots, \vec{q}^1_n), B^{(1)}, \vec{q}^3_1]\,[\vec{q}^3_1, C^{(1)}, \vec{q}^2_1]$. Then, clearly if γ_1 satisfies condition 2, then γ'_1 as well (due to $\vec{q}^2_1$ not changing).

  Condition 3 is trivially satisfied and condition 4 is clearly true for γ'_i, if it is true for γ_i, since all γ_i with i such that $2 \leq i \leq n$ are not modified by rules of type 2. Condition 5 is true, since if γ_1 is of the required form, then so is γ'_1 due to the construction of rules of type 2.
- A rule of type 3 is applied. It is of the form $[(q^1, \vec{q}^1_2, \dots, \vec{q}^1_{j-1}, \vec{q}^1_j, \vec{q}^1_{j+1}, \dots, \vec{q}^1_n), A^{(1)}, (q^2, \vec{q}^1_2, \dots, \vec{q}^1_{j-1}, \vec{q}^2_j, \vec{q}^1_{j+1}, \dots, \vec{q}^1_n)]$ $\to a([\vec{q}^2_j, B^{(j)}, \vec{q}^1_j])_j$ where $\vec{q}^2_j \neq q^e_j$. We consider two cases: $|\gamma_1| = 1$ and $|\gamma_1| > 1$.

  Case $|\gamma_1| = 1$. Then, conditions 1 and 2 are trivially satisfied, since $\gamma'_1 = \varepsilon$. We have $\gamma_1 = [(q^1, \vec{q}^1_2, \dots, \vec{q}^1_{j-1}, \vec{q}^1_j, \vec{q}^1_{j+1}, \dots, \vec{q}^1_n), A^{(1)}, (q^2, \vec{q}^1_2, \dots, \vec{q}^1_{j-1}, \vec{q}^2_j, \vec{q}^1_{j+1}, \dots, \vec{q}^1_n)]$. Applying the induction hypothesis (condition 1), we know that (1) if γ_i with $2 \leq i \leq n$ is empty, then $\vec{q}^1_i = q^e_i$, else $\gamma_i = [\vec{q}^1_i, B^{(i)}, \vec{q}^3_i]\gamma''_i$ with $\gamma''_i \in V_i^*$. Furthermore, there exists some k such that $(q^2, \vec{q}^1_2, \dots, \vec{q}^1_{j-1}, \vec{q}^2_j, \vec{q}^1_{j+1}, \dots, \vec{q}^1_n)$ satisfies the property of condition 2. Since $\vec{q}^2_j \neq q^e_j$, there are two cases: $k < j$ or $k = j$. If $k < j$ we have (2) for all i with $2 \leq i < k$, $\vec{q}^1_i = q^e_i$ and $\vec{q}^1_k = (q^2, \vec{q}^1_{k+1}, \dots, \vec{q}^1_{j-1}, \vec{q}^2_j, \vec{q}^1_{j+1}, \dots, \vec{q}^1_n)$.

  If $k = j$ we have (3) for all i with $2 \leq i < k$, $\vec{q}^1_i = q^e_i$ and $\vec{q}^2_k = (q^2, \vec{q}^1_{j+1}, \dots, \vec{q}^1_n))$.

  Using this, we show that condition 3 is satisfied on $\overline{\gamma}'$. Let i be the smallest k such that γ'_k is not empty. Notice that since γ'_j is not empty, there are two cases: $i = j$ or $i < j$.

  In the case $i = j$, we have $\gamma'_j = [\vec{q}^2_j, B^{(j)}, \vec{q}^1_j]\gamma_j$ and with (1) we have $\vec{q}^1_l = q^e_l$ for all l with $2 \leq l < j$. Therefore case (3) must apply and $\vec{q}^2_j = (q^2, \vec{q}^1_{j+1}, \dots, \vec{q}^1_n)$. Therefore, if for $l > j$, γ'_l is empty, then γ_l is empty and

with (1) we have $\vec{q}_l^{\,1} = q_l^e$, else with (1) we have $\gamma'_l = \gamma_l = [\vec{q}_l^{\,1}, B^{(l)}, \vec{q}_l^{\,3}]\gamma''_l$ showing condition 3.

In the case $i < j$, we have $\gamma'_i = \gamma_i = [(q_i^1, \vec{q}_{i+1}^{\,1}, \ldots, \vec{q}_{j-1}^{\,1}, \vec{q}_j^{\,2}, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), B^{(i)}, \vec{q}_i^{\,3}])\ \gamma''_i$ with $\gamma''_i \in V_i^*$ because of (1) and (2). Let $l > i$ and $l \neq j$. Then, if γ'_l is empty, then γ_l is empty as well and $\vec{q}_l^{\,1} = q_l^e$ with (1). If γ'_l is not empty, then with (1) we have $\gamma'_l = \gamma_l = [\vec{q}_l^{\,1}, B^{(l)}, \vec{q}_l^{\,3}]\gamma''_l$. For $l = j$ we have $\gamma'_l = \gamma'_j = [\vec{q}_j^{\,2}, B^{(j)}, \vec{q}_j^{\,1}]\gamma_j$ and condition 3 is satisfied.

Conditions 4 and 5 are clearly satisfied for $\overline{\gamma}'$, if they are satisfied for $\overline{\gamma}$.

Case $|\gamma_1| > 1$**.** Then, γ_1 is of the form $[(q^1, \vec{q}_2^{\,1}, \ldots, \vec{q}_{j-1}^{\,1}, \vec{q}_j^{\,1}, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), A^{(1)}, (q^2, \vec{q}_2^{\,1}, \ldots, \vec{q}_{j-1}^{\,1}, \vec{q}_j^{\,2}, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1})][(q^2, \vec{q}_2^{\,1}, \ldots, \vec{q}_{j-1}^{\,1}, \vec{q}_j^{\,2}, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), B^{(1)}, \vec{q}_1^{\,3}]\gamma''_1$ with $\gamma''_1 \in V_1^*$. Then $\gamma'_1 = [(q^2, \vec{q}_2^{\,1}, \ldots, \vec{q}_{j-1}^{\,1}, \vec{q}_j^{\,2}, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), B^{(1)}, \vec{q}_1^{\,3}]\gamma''_1$. For all i with $2 \leq i \leq n$ we have that if γ'_i is empty, then γ_i is empty as well and by induction hypothesis condition 1 is satisfied. If γ'_i is not empty, then for $i \neq j$ we have $\gamma'_i = \gamma_i$ and by induction hypothesis condition 1 is true. For $i = j$ we have $\gamma'_j = [\vec{q}_j^{\,2}, B^{(j)}, \vec{q}_j^{\,1}]\gamma_j$ and so condition 1 is true. Condition 2 is true for γ'_1 since by induction hypothesis it is true for γ_1. Condition 3 is trivially true. Conditions 4 and 5 are clearly satisfied for $\overline{\gamma}'$ if they are satisfied for $\overline{\gamma}$.

- A rule of type 4 is applied: $[(q^1, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), A^{(j)}, \vec{q}_j^{\,1}]$
  $\rightarrow a([(q^4, q_2^e, \ldots, q_{j-1}^e, \vec{q}_j^{\,1}, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), B^{(1)}, (q^2, \vec{q}_2^{\,2}, \ldots, \vec{q}_n^{\,2})])_1$
  for some k with $2 \leq k \leq n+1$ and
  - for all i with $2 \leq i < min(k, j)$ we have $\vec{q}_i^{\,2} = q_i^e$
  - for all i with $min(k, j) \leq i < k$ we have $\vec{q}_i^{\,1} = \vec{q}_i^{\,2} = q_i^e$
  - if $k > 0$, then $\vec{q}_k^{\,2} = (q^2, \vec{q}_{k+1}^{\,2}, \ldots, \vec{q}_n^{\,2})$

  We show that condition 1 is satisfied.
  $\gamma'_1 = [(q^4, \vec{q}_2^{\,1}, \ldots, \vec{q}_{j-1}^{\,1}, \vec{q}_j^{\,1}, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), B^{(1)}, (q^2, \vec{q}_2^{\,2}, \ldots, \vec{q}_n^{\,2})]$ with for all i with $2 \leq i < j$, $\vec{q}_i^{\,1} = q_i^e$.

  Take an i with $2 \leq i \leq n$ such that γ'_i is empty. Then, there are three cases: $i < j$, $i = j$ or $i > j$. If $i < j$, we have $\vec{q}_i^{\,1} = q_i^e$ and condition 1 is satisfied. If $i = j$, we have due to the induction hypothesis (condition 4) that $\gamma_i = \gamma_j = [(q^1, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), A^{(j)}, q_j^e]$. Therefore $\vec{q}_j^{\,1} = q_i^e$. In the case $i > j$ we have $\gamma'_i = \gamma_i$ and with induction hypothesis (condition 3) we have $\vec{q}_i^{\,1} = q_i^e$.

  Take an i with $2 \leq i \leq n$ such that γ'_i is not empty. Due to the definition of a derivation j is the smallest k such that γ_k is not empty. Therefore, we have $i \geq j$. For $i = j$ we have $\gamma_j = [(q^1, \vec{q}_{j+1}^{\,1}, \ldots, \vec{q}_n^{\,1}), A^{(j)}, \vec{q}_j^{\,1}]\gamma'_j$. Due to condition 4 applied inductively on γ_j we have $\gamma'_j = [\vec{q}_j^{\,1}, B^{(j)}, \vec{q}_j^{\,5}]\gamma''_i$ for some $\gamma''_j \in S_j^*$. This shows condition 1 for the case $i = j$. For the case $i > j$ we have with induction hypothesis (condition 3) that $\gamma_i = [\vec{q}_i^{\,1}, A^{(i)}, \vec{q}_i^{\,4}]\gamma''_i$. Since $\gamma'_i = \gamma_i$ condition 1 is satisfied. Clearly, condition 2 is satisfied by construction. Condition 3 is trivially satisfied. Conditions 4 and 5 are clearly satisfied.
- A rule of type 5 is applied: This case is very similar to the case of rules of type

3. We have to distinguish two cases: $|\gamma_1| = 1$ and $|\gamma_1| > 1$. Then, the reasoning is the same as for rules of type 3.

This concludes the proof of Lemma 16. □

Now, we continue the formal proof of Theorem 14 by showing that the grammar G_M constructed from the automaton M accepts the same language.

3.1. *From a derivation in G_M to a run of the automaton M*

We prove that for any $x \in \Sigma^*$, $S \Rightarrow^* x$ implies $\langle q_0, x; Z_0\bot, \bot, \dots \bot\rangle \vdash_M^* \langle q, \varepsilon; \bot, \dots, \bot\rangle$ for some $q \in Q$. Let us fix $S \Rightarrow^* x$ (which is a shorthand for $(S)_1(\varepsilon)_2 \dots (\varepsilon)_n \Rightarrow^* x(\varepsilon)_1 \dots (\varepsilon)_n$).

Let $\overline{\gamma} = (\gamma_1)_1(\gamma_2)_2 \dots (\gamma_n)_n$ be an n-list such that $(S)_1(\varepsilon)_2 \dots (\varepsilon)_n \Rightarrow^+ w\overline{\gamma} \Rightarrow^+ x(\varepsilon)_1 \dots (\varepsilon)_n$. Notice that at least one of γ_i is not empty. The configuration $c_{w\overline{\gamma}}$ of M corresponding to $w\overline{\gamma}$ is defined as follows : $c_{w\overline{\gamma}} = \langle q, w'; \alpha_1, \dots, \alpha_n\rangle$ where

- $w' \in \Sigma^*$ such that $x = ww'$
- for all $i \in \{1, \dots, n\}$ we have if $\gamma_i = \varepsilon$, then $\alpha_i = \bot$ and if $\gamma_i = [\vec{q}_i^{\,1}, A_1^{(i)}, \vec{q}_i^{\,2}][\vec{q}_i^{\,2}, A_2^{(i)}, \vec{q}_i^{\,3}] \dots [\vec{q}_i^{\,j-1}, A_{j-1}^{(i)}, \vec{q}_i^{\,j}][\vec{q}_i^{\,j}, A_j^{(i)}, \vec{q}_i^{\,j+1}]$ for some $\vec{q}_i^{\,1}, \dots, \vec{q}_i^{\,j+1} \in Q_i$, then $\alpha_i = A_1^{(i)}A_2^{(i)} \dots A_j^{(i)}\bot$.
- Let i be the smallest k such that $\gamma_k \neq \varepsilon$. If $\gamma_i = [(q_i^1, \vec{q}_{i-1}^{\,1}, \dots, \vec{q}_1^{\,1}), A_1^{(i)}, \vec{q}_i^{\,2}]\gamma_i'$ with $\gamma_i' \in V_i^*$, then $q = q_i^1$.

Now it is enough to show that (4) if $(S)_1(\varepsilon)_2 \dots (\varepsilon)_n \Rightarrow \overline{\gamma}$, then $c_{\overline{\gamma}} = \langle q_0, x; Z_0\bot, \bot, \dots, \bot\rangle$, (5) if $w\overline{\gamma} \Rightarrow x(\varepsilon)_1 \dots (\varepsilon)_n$ then there exists $q \in Q$ with $c_{w\overline{\gamma}} \vdash_M \langle q, \varepsilon; \bot, \dots, \bot\rangle$ and (6) for each step $(S)_1(\varepsilon)_2 \dots (\varepsilon)_n \Rightarrow^+ w\overline{\gamma} \Rightarrow wa\overline{\gamma}' \Rightarrow^+ x$ with $a \in \Sigma \cup \{\varepsilon\}$ in the derivation we have $c_{w\overline{\gamma}} \vdash_M c_{wa\overline{\gamma}'}$.

(4) is true by construction of the rules of type 1. To prove (5) let us consider $\overline{\gamma}$ such that $w\overline{\gamma} \Rightarrow x(\varepsilon)_1 \dots (\varepsilon)_n$. This derivation is only possible using a rule of type 5 being of the form $[(q^1, \vec{q}_2^{\,1}, \dots, \vec{q}_n^{\,1}), A^{(1)}, (q^2, \vec{q}_2^{\,1}, \dots, \vec{q}_n^{\,1})] \to a$ such that $(q^1, A^{(1)}, \varepsilon, \dots, \varepsilon) \xrightarrow{a} (q^2, \varepsilon, \dots, \varepsilon)$. Therefore using Lemma 16 (condition 1) $\overline{\gamma}$ is of the form $([(q^1, q_2^e, \dots, q_n^e), A^{(1)}, (q^2, q_2^e, \dots, q_n^e)])_1(\varepsilon)_2 \dots (\varepsilon)_n$. Hence $c_{w\overline{\gamma}} = \langle q^1, a; A^{(1)}\bot, \bot, \dots, \bot\rangle$ and $c_{w\overline{\gamma}} \vdash_M \langle q^2, \varepsilon; \bot, \dots, \bot\rangle$ using the transition.

We prove (6) by considering the rules of type 2 to type 5 (rules of type 1 can only be applied once in the beginning).

- Rules of type 2 have the form $[(q^1, \vec{q}_2^{\,1}, \dots, \vec{q}_n^{\,1}), A^{(1)}, \vec{q}_1^{\,2}] \to a([(q^4, \vec{q}_2^{\,1}, \dots, \vec{q}_n^{\,1}), B^{(1)}, \vec{q}_1^{\,3}][\vec{q}_1^{\,3}, C^{(1)}, \vec{q}_1^{\,2}])_1$ and there is a transition in M such that $(q^1, A^{(1)}, \varepsilon, \dots, \varepsilon) \xrightarrow{a} (q^4, B^{(1)}C^{(1)}, \varepsilon, \dots, \varepsilon)$.

  Then, $\overline{\gamma}$ is of the form $([(q^1, \vec{q}_2^{\,1}, \dots, \vec{q}_n^{\,1}), A^{(1)}, \vec{q}_1^{\,2}]\gamma_1')_1(\gamma_2)_2 \dots (\gamma_n)_n$ and $\overline{\gamma}'$ is of the form $([(q^4, \vec{q}_2^{\,1}, \dots, \vec{q}_n^{\,1}), B^{(1)}, \vec{q}_1^{\,3}][\vec{q}_1^{\,3}, C^{(1)}, \vec{q}_1^{\,2}]\gamma_1')_1\ (\gamma_2)_2\ \dots (\gamma_n)_n$. Furthermore $c_{w\overline{\gamma}}$ is of the form $\langle q^1, w'; A^{(1)}\alpha_1, \alpha_2, \dots, \alpha_n\rangle$ and $c_{wa\overline{\gamma}'}$ is of the form $\langle q^4, w''; B^{(1)}C^{(1)}\alpha_1, \alpha_2, \dots, \alpha_n\rangle$. Then, clearly $c_{w\overline{\gamma}} \vdash_M c_{wa\overline{\gamma}'}$ using the corresponding transition.

- Rules of type 3 have the form:
$[(q^1, \vec{q}^1_2, \ldots, \vec{q}^1_{j-1}, \vec{q}^1_j, \vec{q}^1_{j+1}, \ldots, \vec{q}^1_n), A^{(1)}, (q^2, \vec{q}^1_2, \ldots, \vec{q}^1_{j-1}, \vec{q}^2_j, \vec{q}^1_{j+1}, \ldots, \vec{q}^1_n)]$ $\rightarrow a([\vec{q}^2_j, B^{(j)}, \vec{q}^1_j])_j$ with $\vec{q}^2_j \neq q^e_j$ and there is a transition $(q^1, A^{(1)}, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q^2, \varepsilon, \ldots, \varepsilon, B^{(j)}, \varepsilon, \ldots, \varepsilon)$ in M. We distinguish two cases: $|\gamma_1| > 1$ or $|\gamma_1| = 1$. In the former case (there are at least 2 non terminals in the first list), because of Lemma 16 (condition 5), $\overline{\gamma}$ is of the form $([(q^1, \vec{q}^1_2, \ldots, \vec{q}^1_n), A^{(1)}, \vec{q}^2_1]\ [\vec{q}^2_1, B^{(1)}, \vec{q}^3_1]\gamma'_1)_1(\gamma_2)_2 \ldots (\gamma_n)_n$. Furthermore $\vec{q}^2_1 = (q^2, \vec{q}^1_2, \ldots, \vec{q}^1_{j-1}, \vec{q}^2_j, \vec{q}^1_{j+1}, \ldots, \vec{q}^1_n)$ as a rule of type 3 is applied and therefore $\overline{\gamma}'$ is of the form $([\vec{q}^2_1, B^{(1)}, \vec{q}^3_1]\gamma'_1)_1\ (\gamma_2)_2 \ldots (\gamma_{j-1})_{j-1}([\vec{q}^2_j, B^{(j)}, \vec{q}^1_j]\gamma_j)_j$ $(\gamma_{j+1})_{j+1} \ldots (\gamma_n)_n$.

  Furthermore $c_{w\overline{\gamma}}$ is of the form $\langle q^1, w'; A^{(1)}B^{(1)}\alpha_1, \alpha_2, \ldots, \alpha_n\rangle$ and $c_{wa\overline{\gamma}'}$ is of the form $\langle q^4, w''; B^{(1)}\alpha_1, \alpha_2, \ldots, \alpha_{j-1}, B^{(j)}\alpha_j, \alpha_{j+1}, \ldots, \alpha_n\rangle$. Then clearly $c_{w\overline{\gamma}} \vdash_M c_{wa\overline{\gamma}'}$ using the corresponding transition.

  In the case $|\gamma_n| = 1$, $\overline{\gamma}$ is of the form $([(q^1, \vec{q}^1_2, \ldots, \vec{q}^1_n), A^{(1)}, \vec{q}^2_1])_1\ (\gamma_2)_2$ $\ldots (\gamma_n)_n$ with $\vec{q}^2_1 = (q^2, \vec{q}^1_2, \ldots, \vec{q}^1_{j-1}, \vec{q}^2_j, \vec{q}^1_{j+1}, \ldots, \vec{q}^1_n)$ and $\overline{\gamma}'$ is of the form $(\varepsilon)_1(\gamma_2)_2 \ldots (\gamma_{j-1})_{j-1}([\vec{q}^2_j, B^{(j)}, \vec{q}^1_j]\gamma_j)_j(\gamma_{j+1})_{j+1} \ldots (\gamma_n)_n$.

  Let i be the smallest k such that γ_k is not $\perp$.

  - if $i < j$, then due to Lemma 16 (condition 1) applied on $\overline{\gamma}$ we have $\gamma_i = [\vec{q}^1_i, A^{(i)}, \vec{q}^3_i]\gamma'_i$ and for all l such that $l < i$, $\vec{q}^1_l = q^e_1$. Therefore (7) $\vec{q}^1_i = (q^2, \vec{q}^1_{i+1}, \ldots, \vec{q}^1_{j-1}, \vec{q}^2_j, \vec{q}^1_{j+1}, \ldots, \vec{q}^1_n)$ with Lemma 16 (condition 2).
  - if $i \geq j$, then according to Lemma 16 (conditions 1 and 2) for all l such that $l < i$, $\vec{q}^1_l = q^e_1$ and since $\vec{q}^2_j \neq q^e_j$ we have (8) $\vec{q}^2_j = (q^2, \vec{q}^1_{j+1}, \ldots, \vec{q}^1_n)$.

  Furthermore, $c_{w\overline{\gamma}}$ is of the form $\langle q^1, w'; A^{(1)}\perp, \alpha_2, \ldots, \alpha_n\rangle$ and it is easy to see using (7) and (8) that $c_{wa\overline{\gamma}'}$ is of the form $\langle q^2, w''; \perp, \alpha_2, \ldots, \alpha_{j-1}, B^{(j)}\alpha_j, \alpha_{j+1}, \ldots, \alpha_n\rangle$. Then clearly $c_{w\overline{\gamma}} \vdash_M c_{wa\overline{\gamma}'}$ using the corresponding transition.
- Rules of type 4 have the form $[(q^1, \vec{q}^1_{j+1}, \ldots, \vec{q}^1_n), A^{(j)}, \vec{q}^1_j]$ $\rightarrow a([(q^4, q^e_2, \ldots, q^e_{j-1}, \vec{q}^1_j, \vec{q}^1_{j+1}, \ldots, \vec{q}^1_n), B^{(1)}, (q^2, \vec{q}^2_2, \ldots, \vec{q}^2_n)])_1$ and there is a transition $(q^1, \perp, \ldots, \perp, A^{(j)}, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q^4, B^{(1)}\perp, \perp, \ldots, \perp, \varepsilon, \varepsilon, \ldots, \varepsilon)$ with $2 \leq j \leq n$ in M.

  So, $\overline{\gamma}$ is of the form $(\varepsilon)_1 \ldots (\varepsilon)_{j-1}([(q^1, \vec{q}^1_{j+1}, \ldots, \vec{q}^1_n), A^{(j)}, \vec{q}^1_j]\gamma_j)_j \ldots (\gamma_n)_n$ and $\overline{\gamma}'$ is of the form $([(q^4, q^e_2, \ldots, q^e_{j-1}, \vec{q}^1_j, \vec{q}^1_{j+1}, \ldots, \vec{q}^1_n), B^{(1)}, (q^2, \vec{q}^2_2, \ldots, \vec{q}^2_n)])_1\ (\varepsilon)_2 \ldots (\varepsilon)_{j-1}(\gamma_j)_j \ldots (\gamma_n)_n$.

  Furthermore $c_{w\overline{\gamma}}$ is of the form $\langle q^1, w'; \perp, \ldots, \perp, A^{(j)}\alpha_j, \alpha_{j+1}, \ldots, \alpha_n\rangle$ and $c_{wa\overline{\gamma}'}$ is of the form $\langle q^4, w''; B^{(1)}\perp, \perp, \ldots, \perp, \alpha_j, \ldots, \alpha_n\rangle$. Then clearly $c_{w\overline{\gamma}} \vdash_M c_{wa\overline{\gamma}'}$ using the corresponding transition.
- Rules of type 5: The reasoning is the same as for rules of type 3.

3.2. *From a run of the automaton M to a derivation in G_M*

Here we prove that for any $x \in \Sigma^*$, $\langle q_0, x; Z_0\perp, \perp, \ldots \perp\rangle \vdash^*_M \langle q_1, \varepsilon; \perp, \ldots, \perp\rangle$ for some $q_1 \in Q$ implies $S \Rightarrow^* x$.

Let us fix the sequence $\langle q_0, x; Z_0\perp, \perp, \ldots \perp\rangle \vdash^*_M \langle q_1, \varepsilon; \perp, \ldots, \perp\rangle$. Let $c_0 =$

$\langle q_0, x; Z_0\bot, \bot, \ldots, \bot\rangle$. To each configuration c appearing in the configuration sequence we will give a corresponding n-list $\overline{\alpha^c}$ in the grammar. In the following we will often use α_i, α_i' and α_i'' for $1 \leq i \leq n$ for some string in V_i^* as well as γ_i, γ_i', γ_i'' and γ_i''' for $1 \leq i \leq n$ for some string in $Stack_{\mathcal{M}}$.

Let $c = \langle q, w; \gamma_1, \ldots, \gamma_n\rangle$ be a configuration with $c_0 \vdash_M^* c \vdash_M^* \langle q, \varepsilon; \bot, \ldots, \bot\rangle$. We will define $\overline{\alpha^c} = u(\alpha_1^c)_1 \ldots (\alpha_n^c)_n$ with $x = uw$ inductively. If $c = \langle q, \varepsilon; \bot, \ldots, \bot\rangle$ then $\overline{\alpha^c} = x(\varepsilon)_1 \ldots (\varepsilon)_n$, else there are two cases: $\gamma_1 = \bot$ or not.

In the case $\gamma_1 = \bot$ let i be the smallest k such that γ_k is not $\bot$. Let $\gamma_i = A_1^{(i)} \ldots A_j^{(i)}\bot$. Then, $\alpha_i^c = [\vec{q}_i^1, A_1^{(i)}, \vec{q}_i^2][\vec{q}_i^2, A_2^{(i)}, \vec{q}_i^3] \ldots [\vec{q}_i^j, A_j^{(i)}, q_i^e]$ such that $\vec{q}_i^1 = (q, \vec{q}_{i+1}^{1,i}, \ldots, \vec{q}_n^{1,i})$ where for l with $i < l \leq n$,

- if γ_l is $\bot$, then $\vec{q}_l^{1,i} = q_l^e$, else let $c' = \langle q', x'; \bot, \ldots, \bot, \gamma_l, \gamma_{l+1}' \ldots, \gamma_n'\rangle$ be the configuration later in the run, just before the top symbol of γ_l is read. Inductively we have $\alpha_l^{c'} = [\vec{q}_l^1, B_1^{(l)}, \vec{q}_l^2]\alpha_l'$. Then, $\vec{q}_l^{1,i} = \vec{q}_l^1$.

The $\vec{q}_i^2, \ldots, \vec{q}_i^j$ are defined inductively by considering the configuration later in the sequence where the corresponding symbols $A_2^{(i)} \ldots A_j^{(i)}$ are read. The other α_i^c are then also defined inductively.

In the case $\gamma_1 \neq \bot$, let $\gamma_1 = A_1^{(1)} \ldots A_j^{(1)}\bot$. Then, $\alpha_1^c = [\vec{q}_1^1, A_1^{(1)}, \vec{q}_1^2]\ [\vec{q}_1^2, A_2^{(1)}, \vec{q}_1^3] \ldots [\vec{q}_1^j, A_j^{(1)}, \vec{q}_1^{j+1}]$, such that $\vec{q}_1^1 = (q, \vec{q}_2^{1,i}, \ldots, \vec{q}_n^{1,i})$ where for l with $2 \leq l \leq n$,

- if γ_l is $\bot$, then $\vec{q}_l^{1,i} = q_l^e$, else let $c' = \langle q', x'; \bot, \ldots, \bot, \gamma_l, \gamma_{l+1}' \ldots, \gamma_n'\rangle$ be the configuration later in the run, just before the top symbol of γ_l is read. Inductively we have $\alpha_l^{c'} = [\vec{q}_l^1, B_1^{(l)}, \vec{q}_l^2]\alpha_l'$. Then, $\vec{q}_l^{1,i} = \vec{q}_l^1$.

The $\vec{q}_1^2, \ldots, \vec{q}_1^j$ are defined inductively by considering the configuration later in the sequence just before the corresponding symbols $A_2^{(i)} \ldots A_j^{(i)}$ are read. Finally, $\vec{q}_1^{j+1}$ is defined as follows. Let $c' = \langle q', x'; \gamma_1', \ldots, \gamma_n'\rangle$ be the first configuration later in the run where γ_1' is $\bot$. Then $\vec{q}_1^{j+1} = (q', \vec{q}_2^{1,j+1}, \ldots, \vec{q}_n^{1,j+1})$ where for l with $2 \leq l \leq n$, if γ_l is $\bot$, then $\vec{q}_l^{1,j+1} = q_l^e$, else inductively we have $\alpha_l^{c'} = [\vec{q}_l^1, B_1^{(l)}, \vec{q}_l^2]\alpha_l'$ and $\vec{q}_l^{1,j+1} = \vec{q}_l^1$. The other α_i^c are then also defined inductively.

Now, it is sufficient to prove that (9) $S \Rightarrow \overline{\alpha^{c_0}}$ and (10) for each step $c_0 \vdash_M^* c \vdash_M c' \vdash_M^* \langle q, \varepsilon; \bot, \ldots, \bot\rangle$ we have $\overline{\alpha^c} \Rightarrow \overline{\alpha^{c'}}$.

(9) is true by construction of the rules of type 1. We prove (10) by considering all types of transitions of the automaton applied to go from c to c'. Let $c = \langle q, w; \gamma_1, \ldots, \gamma_n\rangle$ and $c' = \langle q', w'; \gamma_1', \ldots, \gamma_n'\rangle$ with $aw' = w$ for some $a \in \Sigma \cup \{\varepsilon\}$.

- A transition of the form $(q, A^{(1)}, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q', B^{(1)}C^{(1)}, \varepsilon, \ldots, \varepsilon)$ is applied. Then, γ_1 is of the form $A^{(1)}\gamma_1''$ and γ_1' is of the form $B^{(1)}C^{(1)}\gamma_1''$. Therefore, $\overline{\alpha^c}$ is of the form $u([(q, \vec{q}_2^1, \ldots, \vec{q}_n^1), A^{(1)}, \vec{q}_1^2]\alpha_1)_1 \ldots (\alpha_n)_n$ and $\overline{\alpha^{c'}}$ is of the form $ua([(q', \vec{q}_2^1, \ldots,\ \vec{q}_n^1)\ , B^{(1)}, \vec{q}_1^3][\vec{q}_1^3, C^{(1)}, \vec{q}_1^2]\alpha_1)_1 \ldots (\alpha_n)_n$ since the contents of the other stacks is not changed by application of the rule. Then, by construction of G_M there is a rule of type 2 allowing $\overline{\alpha^c} \Rightarrow \overline{\alpha^{c'}}$.

- A transition of the form $(q, A^{(1)}, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q', \varepsilon, \ldots, \varepsilon, B^{(j)}, \varepsilon, \ldots, \varepsilon)$ is applied. Then c is of the form $\langle q, w; A^{(1)}\gamma_1'', \ldots, \gamma_{j-1}, \gamma_j, \gamma_{j+1}, \ldots, \gamma_n\rangle$ and c' is of the form $\langle q', w'; \gamma_1'', \ldots, \gamma_{j-1}, B^{(j)}\gamma_j, \gamma_{j+1}, \ldots, \gamma_n\rangle$. There are two cases: γ_1'' is $\perp$ or not.

  Case $\gamma_1'' = \perp$. Then, let i be the smallest k such that γ_k is not $\perp$. There are three cases: $i < j$, $i = j$ or $i > j$.

  In the case $i < j$ we have that $\overline{\alpha^c}$ is of the form $u([(q, q_2^e, \ldots, q_{i-1}^e, \vec{q}_i^1, \ldots, \vec{q}_j^1, \ldots, \vec{q}_n^1), A^{(1)}, (q', q_2^e, \ldots, q_{i-1}^e, \vec{q}_i^1, \ldots, \vec{q}_{j-1}^1, \vec{q}_j^2, \vec{q}_{j+1}^1, \ldots, \vec{q}_n^1)])_1\ (\varepsilon)_2 \ldots (\varepsilon)_{i-1}([\vec{q}_i^1, C^{(i)}, \vec{q}_i^3]\alpha_i)_i\ (\alpha_{i+1})_{i+1} \ldots (\alpha_n)_n$. with $\vec{q}_i^1 = (q', \vec{q}_{i+1}^1, \ldots, \vec{q}_{j-1}^1, \vec{q}_j^2, \vec{q}_{j+1}^1, \ldots, \vec{q}_n^1)$. Furthermore, $\overline{\alpha^{c'}}$ must then be of the form $ua(\varepsilon)_1 \ldots (\varepsilon)_{i-1}([\vec{q}_i^1, C^{(i)}, \vec{q}_i^3]\alpha_i)_i\ (\alpha_{i+1})_{i+1} \ldots (\alpha_{j-1})_{j-1}\ (\alpha_j')_j(\alpha_{j+1})_{j+1} \ldots (\alpha_n)_n$. α_j' consists of one symbol of V_j followed by α_j. By construction of G_M there is a rule of type 3 allowing $\overline{\alpha^c} \Rightarrow \overline{\alpha^{c'}}$.

  In the case $i = j$ we have that $\overline{\alpha^c}$ is of the form $u([(q, q_2^e, \ldots, q_{j-1}^e, \vec{q}_j^1, \ldots, \vec{q}_n^1), A^{(1)}, (q', q_2^e, \ldots, q_{j-1}^e, \vec{q}_j^2, \vec{q}_{j+1}^1, \ldots, \vec{q}_n^1)])_1\ (\varepsilon)_2 \ldots (\varepsilon)_{j-1}([\vec{q}_j^1, C^{(i)}, \vec{q}_j^3]\alpha_j)_j\ (\alpha_{j+1})_{j+1} \ldots (\alpha_n)_n$ with $\vec{q}_j^2 = (q', \vec{q}_{j+1}^1, \ldots, \vec{q}_n^1)$. Then, $\overline{\alpha^{c'}}$ must be of the form $ua(\varepsilon)_1 \ldots (\varepsilon)_{j-1}([\vec{q}_j^2, B^{(j)}, \vec{q}_j^1][\vec{q}_j^1, C^{(j)}, \vec{q}_j^3]\alpha_j)_j\ (\alpha_{j+1})_{j+1} \ldots (\alpha_n)_n$ and by construction of G_M there is a rule of type 3 allowing $\overline{\alpha^c} \Rightarrow \overline{\alpha^{c'}}$.

  In the case $i > j$ we have that $\overline{\alpha^c}$ is of the form $u([(q, q_2^e, \ldots, q_{i-1}^e, \vec{q}_i^1, \ldots, \vec{q}_1^1), A^{(1)}, (q', q_2^e, \ldots, q_{j-1}^e, \vec{q}_j^2, q_{j+1}^e, \ldots, q_{i-1}^e, \vec{q}_i^1, \ldots, \vec{q}_n^1)])_1\ (\varepsilon)_2 \ldots (\varepsilon)_{j-1}\ (\varepsilon)_j(\varepsilon)_{j-1} \ldots (\alpha_i)_i \ldots (\alpha_n)_n$ with $\vec{q}_j^2 = (q', q_{j+1}^e, \ldots, q_{i-1}^e, \vec{q}_i^1, \ldots, \vec{q}_n^1)$. Furthermore, $\overline{\alpha^{c'}}$ must then be of the form $ua(\varepsilon)_1 \ldots (\varepsilon)_{j-1}([\vec{q}_j^2, B^{(i)}, q_j^e])_i\ (\varepsilon)_{j+1} \ldots (\varepsilon)_{i-1}\ (\alpha_i)_i \ \ldots\ (\alpha_n)_n$ and by construction of G_M there is a rule of type 3 allowing $\overline{\alpha^c} \Rightarrow \overline{\alpha^{c'}}$.

  Case $\gamma_1'' \neq \perp$**.** Then, γ_1'' is of the form $C^{(1)}\gamma_1'''$. Let $\overline{\alpha^{c'}}$ be of the form $ua([(q', \vec{q}_2^1, \ldots, \vec{q}_{j-1}^1, \vec{q}_j^2, \vec{q}_{j+1}^1, \ldots, \vec{q}_n^1), C^{(1)}, \vec{q}_1^3]\ \alpha_1'')_1)(\alpha_2)_2 \ \ldots\ (\alpha_n)_n$. Since the only stack which has changed by applying the rule is stack j, we have that $\overline{\alpha^c}$ is of the form $u([(q, \vec{q}_2^1, \ldots, \vec{q}_n^1), A^{(1)}, (q', \vec{q}_2^1, \ldots, \vec{q}_{j-1}^1, \vec{q}_j^2, \vec{q}_{j+1}^1, \ldots, \vec{q}_n^1)][(q', \vec{q}_2^1, \ldots, \vec{q}_{j-1}^1, \vec{q}_j^2, \vec{q}_{j+1}^1, \ldots, \vec{q}_n^1), C^{(1)}, \vec{q}_1^3]\ \alpha_1'')_1)\ (\alpha_2)_2 \ldots (\alpha_{j-1})_{j-1}\ (\alpha_j')_j\ (\alpha_{j+1})_{j+1} \ldots (\alpha_n)_n$.

  Then, by construction of G_M there is a rule of type 3 allowing $\overline{\alpha^c} \Rightarrow \overline{\alpha^{c'}}$.
- A transition of the form $(q, \perp, \ldots, \perp, A^{(j)}, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q', B^{(1)}\perp, \perp, \ldots, \perp, \varepsilon, \varepsilon, \ldots, \varepsilon)$ is applied. Then c is of the form $\langle q, w; \perp, \ldots, \perp, A^{(j)}\gamma_j'', \gamma_{j+1}, \ldots, \gamma_n\rangle$ and c' is of the form $\langle q', w'; B^{(1)}, \perp, \ldots, \perp, \gamma_j'', \gamma_{j+1}, \ldots, \gamma_n\rangle$. Then, $\overline{\alpha^c}$ is of the form $u(\varepsilon)_1 \ldots (\varepsilon)_{j-1}\ ([(q, \vec{q}_{j+1}^1, \ldots, \vec{q}_n^1), A^{(j)}, \vec{q}_j^1]\alpha_j'')_j\ (\alpha_{j+1})_{j+1} \ldots (\alpha_n)_n$ and $\overline{\alpha^{c'}}$ is of the form $ua([(q', q_2^e, \ldots, q_{j-1}^e, \vec{q}_j^1, \ldots, \vec{q}_n^1), B^{(1)}, (q^2, \vec{q}_2^2, \ldots, \vec{q}_n^2)])_1 \ldots (\varepsilon)_{j-1}\ (\alpha_j'')_j\ (\alpha_{j+1})_{j+1} \ldots (\alpha_n)_n$. It can be easily verified using the definition of $\overline{\alpha^{c'}}$ that there exists k with $2 \leq k \leq n+1$
  - for all i with $2 \leq i < min(k, j)$ we have $\vec{q}_i^2 = q_i^e$
  - for all i with $min(k, j) \leq i < k$ we have $\vec{q}_i^1 = \vec{q}_i^2 = q_i^e$
  - if $k > 0$, then $\vec{q}_k^2 = (q^2, \vec{q}_{k+1}^2, \ldots, \vec{q}_n^2)$

Then, by construction of G_M there is a rule of type 4 allowing $\overline{\alpha^c} \Rightarrow \overline{\alpha^{c'}}$.

- A transition of the form $(q, A^{(1)}, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q', \varepsilon, \ldots, \varepsilon)$ where $a \in \Sigma \cup \{\varepsilon\}$ is applied. In this case, the reasoning is the same as for transitions of the form $(q, A^{(1)}, \varepsilon, \ldots, \varepsilon) \xrightarrow{a} (q', \varepsilon, \ldots, \varepsilon, B^{(j)}, \varepsilon, \ldots, \varepsilon)$

This concludes the proof. By observing that the size of the grammar G_M corresponding to an OMPA M is double exponential in the number of stacks and using Lemma 13 we obtain the following corollary.

Corollary 17. *The emptiness problem of OMPA is in 2ETIME.*

It is worth mentioning that the emptiness problem of OMPA is in PTIME when we fix the number of stacks.

4. Emptiness of OMPA is 2ETIME-hard

In this section, we prove that the double exponential upper bound established in Section 3 is tight.

In [28], it is shown that the complexity class 2ETIME is captured by infinite-state automata whose transition rules are defined by a form of multi-stack rewriting that is different from our ordered stack policy. To show that the emptiness problem of OMPA is 2ETIME-hard, we adapt one of the constructions from [28].[d]

Theorem 18. *The emptiness problem for OMPA is 2ETIME-hard under log-lin reductions.*[e]

Proof. It is well-known that the class of problems solvable by alternating Turing machines in space bounded by 2^{dn} for some d (call it AESPACE) equals 2ETIME [12]. Thus, it is sufficient to show that any problem in AESPACE can be reduced, under log-lin reductions, to the emptiness problem for OMPA.

So let T be an alternating Turing machine working in space bounded by 2^{dn} and let w be an input for T of length n. We construct an OMPA M with $2dn + 4$ stacks such that the language of M is non-empty iff w is accepted by T. The simulation of T proceeds in two phases: (1) M guesses a possible accepting run of T on w; (2) M verifies if the guess is indeed a run.

Without loss of generality, we assume that a transition of T is of the form $(q, a) \to op_1 \wedge op_2$ (there may be several transitions sharing the same left-hand side). The meaning of the transition is as follows: If the current state is q and the current symbol is a, then T may branch, simultaneously, into the configurations obtained when applying operations op_1 and op_2 (which may overwrite a and change the position of the head). This allows us to represent a run of T as a finite binary

[d]Note that this does not imply that OMPA capture the class 2ETIME.
[e]Recall that a problem A is called *log-lin reducible* to some problem B if A can be reduced to B by a logspace bounded deterministic Turing machine producing an output tape of linear size [25].

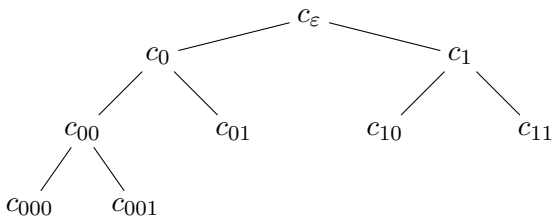

Fig. 2. A run of an alternating Turing machine

tree, as shown in Fig. 2, whose nodes are labeled with configurations. The run is accepting if all leaf configurations are accepting.

The main idea is to encode the (tree) run of a Turing machine as a string so that it can be stored and be manipulated on stacks. Following [28], we write the labeled tree from Fig. 2 as the following string (let c^r denote the reverse of c):

$$\Big(c_\varepsilon \, \big(c_0 \, (\, c_{00} \, (\, c_{000} \, c_{000}^r \,)(\, c_{001} \, c_{001}^r \,) \, c_{00}^r \,)(\, c_{01} \, c_{01}^r \,) \, c_0^r \, \big) \big(c_1 \, (\, c_{10} \, c_{10}^r \,)(\, c_{11} \, c_{11}^r \,) \, c_1^r \, \big) \, c_\varepsilon^r \, \Big)$$

This string corresponds to a depth-first-left-to-right traversal of the tree. Note that, in turn, the tree can be uniquely recovered from the well bracketing of the string. The encoding allows us to access, locally, those pairs of configurations that are related by an edge in the tree and thus need to agree with a transition. The trick is to represent each configuration twice, once written from left to right, and once from right to left. Then, every two configurations that are connected by an edge in the tree run can be written side by side in the string encoding. For example, the edge c_ε — c_0 can be retrieved at the beginning of the string, while c_ε — c_1 is situated at its end (in reverse order).

A configuration $c = a_1 \dots (q, a_i) \dots a_{2^{dn}}$ (with a_i being the i-th symbol on the tape of T) is actually encoded as a string

$$\begin{array}{l} (-, a_1, a_2, e)(a_1, a_2, a_3, e) \dots (a_{i-3}, a_{i-2}, a_{i-1}, e) \\ (a_{i-2}, a_{i-1}, (q, a_i), e)(a_{i-1}, (q, a_i), a_{i+1}, e)((q, a_i), a_{i+1}, a_{i+2}, e) \\ (a_{i+1}, a_{i+2}, a_{i+3}, e) \dots (a_{2^{dn}-1}, a_{2^{dn}}, -, e) \end{array}$$

where each letter is a quadruple. The second component of the i-th letter/quadruple indicates the symbol a_i at the i-th position of T in configuration c. Exactly one such position is also equipped with the current state q. Moreover, the first and third component of a letter maintain, for technical reasons, the symbol of the predecessor and, respectively, successor position on the tape. It is crucial that the encoding of configuration c also contains, at each position, the transition e chosen at c that determines the two successor configurations (if any).

Now, the string encodings of possible run trees of T are generated by the (sketched) context-free grammar

$$\begin{array}{lcl} A & \to & (\, C \,) \\ C & \to & \alpha C \alpha \;\mid\; \alpha A A \alpha \;\mid\; \alpha\, \alpha \end{array}$$

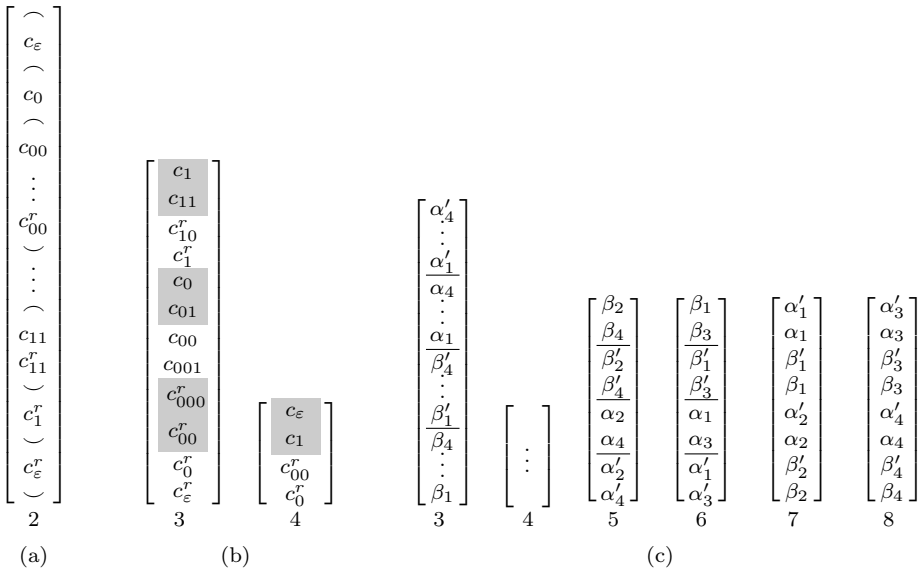

Fig. 3. Guessing and verifying a run of an alternating Turing machine

where α ranges over the quadruples described above, i.e., the atomic building blocks of an encoding of a configuration of T. Note that the grammar can be modified so as to make sure that it produces only valid encodings of configurations, all leaves are accepting configurations, and the initial configuration corresponds to the input w. Using two stacks, we can generate a string encoding of a (possible) run of T and write it onto the second stack, with c_ε at the top, while leaving the first stack empty behind us (cf. Fig. 3(a)).

The OMPA M now checks if the word written onto stack 2 stems from a run of T that corresponds to the transitions. To this aim, we first extract, from stack 2, any pair of configurations that need to be compared wrt. the transition relation of T, namely configurations that are connected by an edge in the tree run. Since configurations at inner tree nodes have to be compared with several other configurations, they need to be duplicated, so that we have to use two further stacks. Let us explain, by means of Fig. 3(b), how we transfer the contents of stack 2 onto new stacks 3 and 4 (and, in doing so, reverse its order), so that all neighboring configurations to be checked are written side by side. The bottom of stack 3 will consist of $c_0^r\, c_\varepsilon^r$ (bottom meant to be at the end of the word). Now, configuration c_0, in the tree run, is connected with both c_ε and c_{00}. Therefore, we shall have two copies of c_0^r. To do so, we take c_0 from stack 2 and simultaneously write c_0^r both onto stack 3 and stack 4. We continue like this and write every further configuration that we see on stack 2, both onto stack 3 and stack 4, until we discover two consecutive letters

of the form $(\ldots, -, e)(-, \ldots, e)$, which corresponds to reaching a leaf.[f]

At the end of this procedure, there are an even number of configurations on both 3 and 4. Two consecutive configurations (which, in Fig. 3(b), share a white or gray region) have to be compared with each other and shall differ only locally, since T modifies its tape locally. Moreover, it still remains to verify that c_ε and c_0, c_0 and c_{01}, etc. correspond to a transition of T. The encoding of one single configuration will now allow us to compare two configurations letter by letter. Recall that the encoding includes a component for a transition e, which has been selected to be executed next and which has been guessed in the above grammar. We would like to compare the k-th letter of one with the k-th letter of another configuration. To access corresponding letters simultaneously, we divide the configurations on stacks 3 and 4 into two, using two further stacks, 5 and 6. We continue this until corresponding letters are arranged one below the other. This procedure, which requires $2dn$ additional stacks, is illustrated in Fig. 3(c) where each of the $\alpha_i, \alpha_i', \beta_i, \beta_i'$ stands for a quadruple plus some information on how it has to be compared with its neighbor. This is discussed below.

Note that, in some cases, we encounter pairs of the form (c, c') whereas, in some other cases, we face pairs of the form $(c^r, (c')^r)$. Whether we deal with the reverse of a configuration or not can be recognized on the basis of its border symbols (i.e., $(-, a_1, a_2, e)$ or $(a_{2^{dn}-1}, a_{2^{dn}}, -, e)$). Consider, for example, stacks 3 and 4 in Fig. 3(b). We want to compare c_ε and c_1 (written on stack 4) where c_ε is of the form $(-, a_1, a_2, e)\ldots$, i.e., it is read from left to right. Suppose the chosen transition e is such that c_ε branches into c and c'. Then, locally comparing c_ε and c_1, we can check whether $c' = c_1$. If, at the bottom of stack 3, we compare $c_\varepsilon^r = (a_{2^{dn}-1}, a_{2^{dn}}, -, e)\ldots$ with c_0^r, then we need to check whether $c = c_0$. In other words, the order in which a configuration is read indicates if we follow the right or left successor in the run tree. This information has to be added to the quadruples. □

From Corollary 17 and Theorem 18, we deduce our main result:

Theorem 19. *The emptiness problem of OMPA is 2ETIME-complete under log-lin reductions.*[g]

5. Comparison to bounded-phase multi-stack pushdown automata

In this section, we consider bounded-phase multi-stack (visibly) pushdown automata (BMPA/BVMPA). We refer to Definitions 3–5. We will show that, given a BVMPA $\mathcal{B}$, it is possible to construct an OMPA $\mathcal{O}$ such that $\mathcal{B}$ and $\mathcal{O}$ accept the same

[f] Leaf configurations need only be written once. The OMPA will simply guess when we reach such a configuration and not write, for example, c_{000}^r onto stack 4.

[g] The emptiness problem of OMPA is 2EXPTIME-complete, too. Hereby, 2EXPTIME denotes the class of all decision problems solvable by a deterministic Turing machine in time $exp(exp(n^d))$ for some constant d ($exp(x)$ denoting 2^x). Note that 2EXPTIME is a robust complexity class. On the other hand, 2ETIME is not robust, as it is not closed under logspace reductions.

language (i.e., $L(\mathcal{O}) = L(\mathcal{B})$). In addition, we prove that OMPA are *strictly* more expressive than BVMPA.

Theorem 20. *OMPA are strictly more expressive than BVMPA.*

Proof. In the following, we show that the class of languages recognized by BVMPA is included in the class of the languages recognized by OMPA. Once this result is proved, it easy to show the *strict* inclusion since the class of languages recognized by OMPA is closed under Kleene-star (Lemma 9), whereas the class of languages recognized by BVMPA is not (Lemma 10).

Let us fix a n-BVMPA $\mathcal{B} = (\mathcal{M}, k)$ where $k \geq 1$ and $\mathcal{M} = (Q, n, \Sigma, \gamma, \rightarrow, q_0, Z_0, F)$ is a VMPA and $type : \Sigma \rightarrow (\{\text{push}, \text{pop}\} \times [n]) \cup \{\text{int}\}$ is a mapping satisfying conditions of the Definition 3. The schema of the proof is as follows: First, we show that the language $L(\mathcal{B})$ can be written as a finite union of languages which are all recognizable by $2k$-OMPA. Then, we use the property that the class of languages recognized by $2k$-OMPA is closed under union (Lemma 9) to build a $2k$-OMPA accepting exactly the language $L(\mathcal{B})$.

Definition 21. *For every sequence $i_1, \ldots, i_k \in [n]$, let $L_{i_1,\ldots,i_k}(\mathcal{B}) = \{w \in \Sigma^* \mid \langle q_0, w; Z_0\bot, \bot, \ldots, \bot\rangle \vdash_{\mathcal{B}}^{(i_1)} \cdot \vdash_{\mathcal{B}}^{(i_2)} \cdots \vdash_{\mathcal{B}}^{(i_k)} c$ for some final configuration $c\}$.*

Then, we have that $L(\mathcal{B})$ is the union of all these languages $L_{i_1,\ldots,i_k}(\mathcal{B})$.

Lemma 22. $L(\mathcal{B}) = \bigcup_{i_1,\ldots,i_k \in [n]} L_{i_1,\ldots,i_k}(\mathcal{B})$.

Proof. An immediate consequence of the definition of the language $L(\mathcal{B})$ and Definition 21. □

Next, we prove that each language $L_{i_1,\ldots,i_k}(\mathcal{B})$, where $i_1, \ldots, i_k \in [n]$, is recognized by a $2k$-OMPA.

Proposition 23. *For every sequence $i_1, \ldots, i_k \in [n]$, it is possible to construct a $2k$-OMPA $\mathcal{O}_{i_1,\ldots,i_k}$ such that $L(\mathcal{O}_{i_1,\ldots,i_k}) = L_{i_1,\ldots,i_k}(\mathcal{B})$.*

Proof. In the following, we show that it is possible to construct an $2k$-OMPA $\mathcal{O}_{i_1,\ldots,i_k}$ over Σ such that $L(\mathcal{O}_{i_1,\ldots,i_k}) = L_{i_1,\ldots,i_k}(\mathcal{B})$. The constructed $2k$-OMPA $\mathcal{O}_{i_1,\ldots,i_k}$ will only use its first $(2k-1)$ during the simulation and its $2k$-th stack will never be manipulated. In fact, we use $2k$ stacks instead of $(2k-1)$ stacks in the only aim to simplify the notation and avoiding to deal with the $(2k-1)$-th stack as a special case.

First, we observe that any computation accepting a word $w \in L_{i_1,\ldots,i_k}(\mathcal{B})$ can be decomposed into k phases, where in each phase (say j), $\mathcal{M}$ can only pop from the stack i_j (but it can push onto all stacks).

Let $j \in [k]$ be the current phase of $\mathcal{M}$. For every $l \in [k]$, we define $next_{\geq j}(l)$ to be $min\big(\{m \mid j \leq m \leq k \wedge i_m = l\} \cup \{k+1\}\big)$ (i.e., the closest phase to j in $[j..k]$

such that $\mathcal{M}$ is allowed to pop from the l-th stack. Note that $next_{\geq j}(i_j) = j$ and if a such phase does not exist, then $next_{\geq j}(l) = k + 1$.

We give hereafter the details of the construction of the $2k$-OMPA $\mathcal{O}_{i_1,\ldots,i_k}$ over Σ such that $L(\mathcal{O}_{i_1,\ldots,i_k}) = L_{i_1,\ldots,i_k}(\mathcal{B})$. $\mathcal{O}_{i_1,\ldots,i_k}$ is built up from $\mathcal{M}$ such that the following invariant is preserved during the simulation of $\mathcal{M}$ when its current phase is j: the content of the l-th stack of $\mathcal{M}$ is stored in the $(2 \cdot next_{\geq j}(l) - 1)$-th stack of $\mathcal{O}_{i_1,\ldots,i_k}$ if $next_{\geq j}(l) \neq k + 1$ (while for every $r \in [2j - 2]$, the r-th stack of $\mathcal{O}_{i_1,\ldots,i_k}$ is empty). Then, an internal move (labeled by a symbol $a \in \Sigma$ such that $type(a) = int$) of $\mathcal{M}$ is simulated by an internal move (labeled by a) of $\mathcal{O}_{i_1,\ldots,i_k}$; a pop rule (labeled by a symbol $a \in \Sigma$ such that $type(a) = (pop, i_j)$) of $\mathcal{M}$ from the i_j-th stack corresponds to a pop rule (labeled by a) of $\mathcal{O}_{i_1,\ldots,i_k}$ from the $(2j-1)$-th stack; and a push rule (labeled by a symbol $a \in \Sigma$ such that $type(a) = (push, l)$) onto the l-th stack of $\mathcal{M}$ is simulated by a push rule (labeled by a) of $\mathcal{O}_{i_1,\ldots,i_k}$ onto the $(2next_{\geq j}(l) - 1)$-th stack if $next_{\geq j}(l) \neq (k + 1)$, and an internal move (labeled by a) of $\mathcal{O}_{i_1,\ldots,i_k}$ otherwise .

On switching phase from j to $(j + 1)$, $\mathcal{O}_{i_1,\ldots,i_k}$ moves the content of the $(2j - 1)$-th stack onto the $(2next_{\geq j+1}(i_j) - 1)$-th stack using the $(2j)$-th stack as an intermediary one if $next_{\geq j+1}(i_j) \neq k+1$, and empties the $(2j-1)$-th stack otherwise. Observe that all the above described behaviours maintain the stated invariant since $next_{\geq (j+1)}(l) = next_{\geq j}(l)$ for all $l \neq i_j$.

Formally, the $2k$-OMPA $\mathcal{O}_{i_1,\ldots,i_k} = (Q', 2k, \Sigma, \Gamma', \rightarrow', q'_0, Z'_0, F')$ is defined as:

- $Q' = \big(\big(Q \cup \big(Q \times \{\downarrow\}\big)\big) \times [2k]\big) \cup \{q'_0\}$ is a finite set of states with $q'_0 \notin Q$,
- $\Gamma' = \Gamma \cup \{Z'_0\}$ is a finite stack alphabet with $Z'_0 \notin \Gamma$,
- $F' = F \times \{2k - 1\}$ is the set of final states,
- The transition relation $\rightarrow'$ is given as the union of the following three relations $\rightarrow_{\mathsf{init}}$ (for the initialisation phase), $\rightarrow_{\mathsf{sim}}^{(j)}$ (for the simulation of a phase), and $\rightarrow_{\mathsf{sw}}^{(\ell)}$ (for the simulation of a phase switch), with $j \in [k]$ and $\ell \in [k-1]$. These three relations are defined as the smallest relations satisfying the following conditions:

  - **Initialisation phase:** $(q'_0, Z'_0, \varepsilon, \ldots, \varepsilon) \xrightarrow{\varepsilon} ((q_0, 1), \alpha_1, \ldots, \alpha_{2k}) \in \rightarrow_{\mathsf{init}}$ where $\alpha_{2 \cdot next_{\geq 1}(1)-1} = Z_0$ if and only if $next_{\geq 1}(1) \neq k + 1$ and $\alpha_l = \varepsilon$ for all $l \in ([2k] \setminus \{2 \cdot next_{\geq 1}(1) - 1\})$. This corresponds to a move from the initial configuration of $\mathcal{O}_{i_1,\ldots,i_k}$ to a configuration encoding the initial configuration of $\mathcal{M}$ w.r.t. the invariant (stated above).

  - **Simulation of a phase of $\mathcal{M}$:** For every $l \in [n]$, for every $j \in [k]$, and for every transition rule $(q, A_1, \ldots, A_n) \xrightarrow{a} (q', \alpha_1, \ldots, \alpha_n) \in \rightarrow$, we have:

    * **Simulation of an internal move:** If $type(a) = \text{int}$ (i.e., $A_i = \alpha_i = \varepsilon$ for all $i \in [n]$), then $((q, 2j - 1), \varepsilon, \ldots, \varepsilon) \xrightarrow{a} ((q', 2j - 1), \varepsilon, \ldots, \varepsilon) \in \rightarrow_{\mathsf{sim}}^{(j)}$.
    * **Simulation of a push move:** If $type(a) = (\text{push}, l)$ (i.e., $A_1 = \ldots = A_n = \varepsilon$, $\alpha_l \in \Gamma$, and $\alpha_i = \varepsilon$ for all $i \in [n] \setminus \{l\}$), then $((q, 2j -$

$1), \varepsilon, \ldots, \varepsilon) \xrightarrow{a} ((q', 2j-1), \alpha'_1, \ldots, \alpha'_{2k}) \in \rightarrow_{\mathsf{sim}}^{(j)}$ where: (1) $\alpha'_r = \varepsilon$ for all $r \in \big([2k] \setminus (2 \cdot next_{\geq j}(l) - 1)\big)$, and (2) $\alpha'_{(2 \cdot next_{\geq j}(l)-1)} = \alpha_l$ if $next_{\geq j}(l) \neq k+1$.

- ∗ **Simulation of a pop move:** If $type(a) = (\mathrm{pop}, l)$ (i.e., $A_l \in \Gamma \cup \{\bot\}$, $\alpha_l \in \{\bot\} \cup \{\varepsilon\}$, and $A_i = \alpha_i = \varepsilon$ for all $i \in [n] \setminus \{l\}$) and $next_{\geq j}(l) = j$, then $((q, 2j-1), A'_1, \ldots, A'_{2k}) \xrightarrow{a} ((q', 2j-1), \alpha'_1, \ldots, \alpha'_{2k}) \in \rightarrow_{\mathsf{sim}}^{(j)}$ where: (1) $A'_r = \alpha'_r = \bot$ for all $r \leq [2j-2]$, (2) $A'_{2j-1} = A_l$ and $\alpha'_{2j-1} = \alpha_l$, (3) $A'_s = \alpha'_s = \varepsilon$ for all $s \in [2j..2k]$.

– **Simulation of a phase switch of $\mathcal{M}$:** For every $\ell \in [k-1]$ and for every $q \in Q$, we have:

- ∗ **A nondeterministic choice of phase switch:** The transition rule $((q, 2\ell-1), \varepsilon, \ldots, \varepsilon) \xrightarrow{\varepsilon} ((q, \downarrow, 2\ell-1), \varepsilon, \ldots, \varepsilon)$ is in $\rightarrow_{\mathsf{sw}}^{(\ell)}$.
- ∗ **Moving the content of the $(2\ell-1)$-th stack into the 2ℓ-stack (in a reversed order):** For every stack symbol $A \in \Gamma$, the transition rule $((q, \downarrow, 2\ell-1), A_1, \ldots, A_{2k}) \xrightarrow{\varepsilon} ((q, \downarrow, 2\ell-1), \alpha_1, \ldots, \alpha_{2k})$ is in $\rightarrow_{\mathsf{sw}}^{(\ell)}$ where: (1) $A_r = \alpha_r = \bot$ for all $r \in [2\ell-2]$, (2) $A_{2\ell-1} = \alpha_{2\ell} = A$ and $A_{2\ell} = \alpha_{2\ell-1} = \varepsilon$, and (3) $A_s = \alpha_s = \varepsilon$ for all $s \in [2\ell+1..2k]$.
- ∗ **Checking the emptiness of the $(2\ell-1)$-th stack:** The transition rule $((q, \downarrow, 2\ell-1), A_1, \ldots, A_{2k}) \xrightarrow{\varepsilon} ((q, \downarrow, 2\ell), \alpha_1, \ldots, \alpha_{2k})$ is in $\rightarrow_{\mathsf{sw}}^{(\ell)}$ where: (1) $A_r = \alpha_r = \bot$ for all $r \in [2\ell-1]$, and (2) $A_s = \alpha_s = \varepsilon$ for all $s \in [2\ell..2k]$.
- ∗ **Moving the content of the $(2\ell-1)$-th stack into the $\big(2 \cdot next_{\ell+1}(i_\ell) - 1\big)$-stack (in reversed order):** For every stack symbol $A \in \Gamma$, the transition rule $((q, \downarrow, 2\ell), A_1, \ldots, A_{2k}) \xrightarrow{\varepsilon} ((q, \downarrow, 2\ell), \alpha_1, \ldots, \alpha_{2k})$ is in $\rightarrow_{\mathsf{sw}}^{(\ell)}$ where: (1) $A_r = \alpha_r = \bot$ for all $r \in [2\ell-1]$, (2) $A_{2\ell} = A$, (3) $A_s = \varepsilon$ for all $s \in [2\ell+1..2k]$, and (4) $\alpha_m = \varepsilon$ for every $m \in \big([2\ell..2k] \setminus (2 \cdot next_{\geq \ell+1}(i_\ell) - 1)\big)$, and (5) $\alpha_{2 \cdot next_{\geq \ell+1}(i_\ell)-1} = A$ if $next_{\geq \ell+1}(i_\ell) \neq k+1$.
- ∗ **Checking the emptiness of the 2ℓ-th stack:** The transition rule $((q, \downarrow, 2\ell), A_1, \ldots, A_{2k}) \xrightarrow{\varepsilon} ((q, 2\ell+1), \alpha_1, \ldots, \alpha_{2k})$ is in $\rightarrow_{\mathsf{sw}}^{(\ell)}$ where: (1) $A_r = \alpha_r \bot$ for all $r \in [2\ell]$, and (2) $A_s = \alpha_s = \varepsilon$ for all $s \in [2\ell+1..2k]$.

Before going into details of the proof, let us introduce some definitions. For every $j \in [k]$ and $\ell \in [k-1]$, we define the relations $\vdash_{\mathsf{sim}}^{(j)}$ and $\vdash_{\mathsf{sw}}^{(\ell)}$ as follows:

$$\vdash_{\mathsf{sim}}^{(j)} = \Big(\bigcup_{t \in \rightarrow_{\mathsf{sim}}^{(j)}} \vdash_{\mathcal{O}_{i_1,\ldots,i_k}}^{(t)} \Big)^* \qquad \text{and} \qquad \vdash_{\mathsf{sw}}^{(\ell)} = \Big(\bigcup_{t \in \rightarrow_{\mathsf{sw}}^{(\ell)}} \vdash_{\mathcal{O}_{i_1,\ldots,i_k}}^{(t)} \Big)^*$$

Definition 24. *Let State be a mapping from the set of configurations of $\mathcal{O}_{i_1,\ldots,i_k}$ to its set of control states Q' such that $State(\langle q', w; \gamma_1, \ldots, \gamma_{2k}\rangle) = q'$.*

Then, the relation between the set of configuration of $\mathcal{M}$ and $\mathcal{O}_{i_1,\ldots,i_k}$ is given by the following definition:

Definition 25. *Let* $\mu : (Conf_{\mathcal{M}} \times [k]) \to Conf_{\mathcal{O}_{i_1,\ldots,i_k}}$ *be a mapping defined as follows:* $\mu(\langle q, w; \gamma_1, \ldots, \gamma_n\rangle, j) = \langle (q, 2j-1), w; \gamma'_1, \ldots, \gamma'_{2k}\rangle$ *where for all* $i \in [2k]$, $\gamma'_i = \gamma_l$ *if* $i = 2 \cdot next_{\geq j}(l) - 1$ *for some* $l \in [n]$, *and* $\gamma'_i = \bot$ *otherwise.*

Proposition 23 is an immediate consequence of Lemma 26 and Lemma 27 (since c is final configuration of $\mathcal{M}$ iff $\mu(c,k)$ is a final configuration of $\mathcal{O}_{i_1,\ldots,i_k}$):

Lemma 26. *For every word* $w \in \Sigma^*$ *and every configuration* c *of* $\mathcal{M}$, *if* $\langle q_0, w; Z_0\bot, \bot, \ldots, \bot\rangle \vdash_{\mathcal{B}}^{(i_1)} \cdots \vdash_{\mathcal{B}}^{(i_k)} c$, *then*

$$\langle q'_0, w; Z'_0\bot, \bot, \ldots, \bot\rangle \, (\vdash_{\mathcal{O}_{i_1,\ldots,i_k}})^* \, \mu(c,k)\,.$$

Lemma 27. *For every* $j \in [k]$, *every* $w \in \Sigma^*$, *and every configuration* c' *of* $\mathcal{O}_{i_1,\ldots,i_k}$ *such that* $State(c') \in Q \times \{2j-1\}$, *if* $\langle q'_0, w; Z'_0\bot, \bot, \ldots, \bot\rangle \, (\vdash_{\mathcal{O}_{i_1,\ldots,i_k}})^* \, c'$, *then there is a configuration* c *of* $\mathcal{M}$ *such that* $\langle q_0, w; Z_0\bot, \bot, \ldots, \bot\rangle \vdash_{\mathcal{B}}^{(i_1)} \cdots \vdash_{\mathcal{B}}^{(i_j)} c$ *and* $c' = \mu(c,j)$.

5.1. *Proof of Lemma 26*

First, we need to prove the following lemma:

Lemma 28. *For every configuration* c_1 *and* c_2 *of* $\mathcal{M}$ *and every* $j \in [k]$, *if* $c_1 \vdash_{\mathcal{B}}^{(i_j)} c_2$ *then* $\mu(c_1, j) \vdash_{\mathsf{sim}}^{(j)} \mu(c_2, j)$. *Moreover, for every configuration* c' *of* $\mathcal{M}$ *and for* $\ell \in [k-1]$, *we have* $\mu(c', \ell) \vdash_{\mathsf{sw}}^{(\ell)} \mu(c', \ell+1)$.

Proof. To show that for every two configurations c_1 and c_2 of $\mathcal{M}$ and every $j \in [k]$, we have that if $c_1 \vdash_{\mathcal{B}}^{(i_j)} c_2$ then $\mu(c_1, j) \vdash_{\mathsf{sim}}^{(j)} \mu(c_2, j)$, it is sufficient to show that for every $t \in Ph(i_j)$ and for every two configurations c_1 and c_2 of $\mathcal{M}$, we have that if $c_1 \vdash_{\mathcal{M}}^{(t)} c_2$, then there is a transition rule $t' \in \to_{\mathsf{sim}}^{(j)}$ such that $\mu(c_1, j) \vdash_{\mathcal{O}_{i_1,\ldots,i_k}}^{(t')} \mu(c_2, j)$. This can be easily proved using a case split on the type of rule t applied.

The fact that for every configuration c' of $\mathcal{M}$ and every $\ell \in [k-1]$, we have that $\mu(c', \ell) \vdash_{\mathsf{sw}}^{(\ell)} \mu(c', \ell+1)$ is due to that, during the simulation of a phase switch, $\mathcal{O}_{i_1,\ldots,i_k}$ first moves, in reverse order, the content of the $2\ell-1$ into the 2ℓ-th stack (so, the content of the 2ℓ-th stack is the reverse of the content of the $2\ell-1$-stack). Then, it moves, in reverse order, the content of the 2ℓ-th stack into the $\big(2 \cdot next_{\geq \ell+1}(i_\ell) - 1\big)$-stack if it exists (i.e., $next_{\geq \ell+1}(i_\ell) \neq k+1$). Moreover, we have that $next_{\geq \ell}(r) = next_{\geq \ell+1}(r)$ for all $r \neq i_\ell$ by definition. Thus, we get that the reachable configuration from $\mu(c', \ell)$ after the simulation of a phase switch is exactly $\mu(c', \ell+1)$. □

Now, let $c_0 = \langle q_0, w; Z_0\bot, \bot, \ldots, \bot\rangle$ be the initial configuration of $\mathcal{B}$ and $c_0 \vdash_{\mathcal{B}}^{(i_1)} \cdots \vdash_{\mathcal{B}}^{(i_k)} c$ be a computation of $\mathcal{B}$. Then there are $c_1, \ldots, c_{k-1}$ configurations of $\mathcal{M}$ such that: $c_0 \vdash_{\mathcal{B}}^{(i_1)} c_1 \vdash_{\mathcal{B}}^{(i_2)} c_2 \cdots c_{k-1} \vdash_{\mathcal{B}}^{(i_k)} c$. We use Lemma 28 to construct the following computation: (1) $\langle q'_0, w; Z'_0\bot, \bot, \ldots, \bot\rangle \vdash_{\mathcal{O}_{i_1,\ldots,i_k}}^{(t_0)} \mu(c_0, 1)$ where

$t_0 \in \rightarrow_{init}$, (2) For every $j \in [k-1]$, we have $\mu(c_{j-1}, j) \vdash_{\mathsf{sim}}^{(j)} \mu(c_j, j) \vdash_{\mathsf{sw}}^{(j)} \mu(c_j, j+1)$, and (3) $\mu(c_{k-1}, k) \vdash_{\mathsf{sim}}^{(k)} \mu(c, k)$. Hence, $\langle q'_0, w; Z'_0\bot, \bot, \ldots, \bot\rangle \, (\vdash_{\mathcal{O}_{i_1,\ldots,i_k}})^* \, \mu(c, k)$.

5.2. *Proof of Lemma 27*

The proof of Lemma 27 is done by induction on j and based on the use of Lemma 29 and Lemma 30 which can be proved easily using the formal definition of $\mathcal{O}_{i_1,\ldots,i_k}$.

Lemma 29. *For every $j \in [k]$, every configuration c_1 of $\mathcal{B}$, every configurations c'_1 and c'_2 of $\mathcal{O}_{i_1,\ldots,i_k}$ such that: $\mu(c_1, j) = c'_1$ and $State(c'_1), State(c'_2) \in Q \times \{2j-1\}$, we have that if $c'_1 \vdash_{\mathsf{sim}}^{(j)} c'_2$, then there is a configuration c_2 of $\mathcal{B}$ such that $c_1 \vdash_{\mathcal{B}}^{(i_j)} c_2$.*

Lemma 30. *For every $\ell \in [k-1]$, every configuration c_1 of $\mathcal{B}$, all configurations c'_1 and c'_2 of $\mathcal{O}_{i_1,\ldots,i_k}$ such that: $\mu(c_1, \ell) = c'_1$, $State(c'_1) \in Q \times \{2\ell - 1\}$, and $State(c'_2) \in Q \times \{2\ell + 1\}$, we have that if $c'_1 \vdash_{\mathsf{sw}}^{(\ell)} c'_2$, then $\mu(c_1, \ell + 1) = c'_2$.*

The base case where $j = 1$ is proved by Lemma 29 and by the fact that the configuration c_1 defined as follows: $\langle q'_0, w; Z'_0\bot, \ldots, \bot\rangle \vdash_{\mathcal{O}_{i_1,\ldots,i_k}}^{(t_0)} c_1$ with $t_0 \in \rightarrow_{init}$ satisfies the condition that $\mu(\langle q_0, w; Z_0\bot, \ldots, \bot\rangle, 1) = c_1$. For the induction step, suppose that Lemma 27 holds for j. Now, suppose that there is a computation $\langle q'_0, w; Z'_0\bot, \bot, \ldots, \bot\rangle \, (\vdash_{\mathcal{O}_{i_1,\ldots,i_k}})^* \, c'$ of $\mathcal{O}_{i_1,\ldots,i_k}$ where $State(c') \in Q \times \{2j+1\}$. Then, there exist two configurations c'_2 and c'_3 of $\mathcal{O}_{i_1,\ldots,i_k}$ such that $State(c'_2) \in Q \times \{2j-1\}$, $State(c'_3) \in Q \times \{2j+1\}$, and the computation $\langle q'_0, w; Z'_0\bot, \bot, \ldots, \bot\rangle \, (\vdash_{\mathcal{O}_{i_1,\ldots,i_k}})^* \, c'$ can be decomposed as follows: (1) $\langle q'_0, w; Z'_0\bot, \bot, \ldots, \bot\rangle \, (\vdash_{\mathcal{O}_{i_1,\ldots,i_k}})^* \, c'_2$, (2) $c'_2 \vdash_{\mathsf{sw}}^{(j)} c'_3$, and (3) $c'_3 \vdash_{\mathsf{sim}}^{(j+1)} c'$. By induction hypothesis, there is a configuration c'' of $\mathcal{B}$ such that $\langle q_0, w; Z_0\bot, \bot, \ldots, \bot\rangle \vdash_{\mathcal{B}}^{(i_1)} \cdots \vdash_{\mathcal{B}}^{(i_j)} c''$ and $c'_2 = \mu(c'', j)$. In addition, we can use Lemma 30, to show that $c'_3 = \mu_(c'', j+1)$. Now, we can apply Lemma 29 to prove the existence of a configuration c of $\mathcal{B}$ such that $c'' \vdash_{\mathcal{B}}^{(i_{j+1})} c$ (hence, $\langle q_0, w; Z_0\bot, \bot, \ldots, \bot\rangle \vdash_{\mathcal{B}}^{(i_1)} \cdots \vdash_{\mathcal{B}}^{(i_{j+1})} c$) and $c = \mu(c', j+1)$.

5.3. *OMPA are strictly more expressive than BMPA*

In the following, we extend the previous result to bounded-phase multi-pushdown automata.

Theorem 31. *OMPA are strictly more expressive than BMPA.*

The idea behind proving the strict inclusion is the following: First, we transform a n-BMPA $\mathcal{B} = (\mathcal{M}, k)$ to a n-BVMPA $\mathcal{B}'$ such that $L(\mathcal{B}) = f\big(L(\mathcal{B}')\big)$ for some renaming function f. In fact, $\mathcal{B}'$ extends the labelling of each transition of $\mathcal{B}$ by the redundant information on its type (i.e., on which stack, this transition is perfomered and if it is a push, pop, or internal action). Then, we apply Theorem 20 and construct OMPA $\mathcal{O}'$ such that $L(\mathcal{O}') = L(\mathcal{B}')$. Finally, we use the

closure of OMPAs under renaming (Lemma 9) to construct an OMPA $\mathcal{O}$ such that $L(\mathcal{O}) = f\big(L(\mathcal{O}')\big) = f\big(L(\mathcal{B}')\big) = L(\mathcal{B})$.

Formally, let $\mathcal{B} = (\mathcal{M}, k)$ be a n-BMPA over Σ. Then, it is possible to construct a $\mathcal{B}' = (\mathcal{M}', k)$ over $\Sigma' = \Sigma'_c \cup \Sigma'_r \cup \Sigma'_{int}$ where $\Sigma'_c = \big(\Sigma \cup \{\varepsilon\}\big) \times \{c\} \times [n]$, $\Sigma'_r = \big(\Sigma \cup \{\varepsilon\}\big) \times \{r\} \times [n]$, and $\Sigma'_{int} = \big(\Sigma \cup \{\varepsilon\}\big) \times \{int\}$, such that $\mathcal{B}'$ extends the transition labelling of $\mathcal{B}$ as follows: (1) every push transition of $\mathcal{B}$ on the i-th stack labelled by $a \in \Sigma \cup \{\varepsilon\}$ is now labelled by (a, c, i) in $\mathcal{B}'$, (2) every pop transition of $\mathcal{B}$ from the i-th stack labelled by $a \in \Sigma \cup \{\varepsilon\}$ is now labelled by (a, r, i), and (3) all the remaining transitions of $\mathcal{B}$ labelled by $a \in \Sigma \cup \{\varepsilon\}$ are now labelled by (a, int). Let f be a function that maps each symbol of the form (a, c, i), (a, r, i), and (a, int) to a. Then, $w \in L(\mathcal{B})$ iff there is some $w' \in L(\mathcal{B}')$ such that $w = f(w')$. It follows that $L(\mathcal{B}) = f\big(L(\mathcal{B}')\big)$. Consider now the OMPA $\mathcal{O}'$ over Σ' constructed from $\mathcal{B}'$ such that $L(\mathcal{O}') = L(\mathcal{B}')$, thanks to Theorem 20. Then, it is possible to construct from $\mathcal{O}'$ an OMPA $\mathcal{O}$ over Σ such that $L(\mathcal{O}) = f\big(L(\mathcal{O}')\big)$ (Lemma 9) which implies that $L(\mathcal{B}) = L(\mathcal{O})$.

To prove the *strict* inclusion, it is easy to see that BMPA are not closed under Kleene-star whereas OMPA are (Lemma 9).

6. Conclusion

We have shown that the emptiness problem for multi-pushdown automata (OMPA) is 2ETIME-complete. The study of the emptiness problem is the first step of a comprehensive study of verification problems for OMPA. For standard pushdown automata, a lot of work has been done (see for example [8]) concerning various model-checking problems. It will be interesting to see how these results carry over to OMPA and at which cost. A basic ingredient of model-checking algorithms is typically to characterize the set of successors or predecessors of sets of configurations. For OMPA, this problem remains to be studied. Another class of extended pushdown automata has been studied extensively: the class of higher-order pushdown automata (HPDA, see for example [15]). It is quite easy to see that HPDA of order n can simulate OMPA with n stacks (which allows us to use all verification results for HPDA also for OMPA). However, the converse is wrong, since emptiness of pushdown automata of order n is $(n-1)$-EXPTIME-complete [15]. Therefore, it is interesting to study dedicated algorithms for the verification of OMPA.

An extension towards ω-automata, allowing for infinite executions, would be worthwhile. Adapting our techniques to that setting might lead to a generalization of results by Seth [23], who provides a game-theoretic proof for decidability of emptiness for Büchi BMPA.

Acknowledgments

We are grateful to the reviewers for their careful reading and many pertinent comments.

References

[1] Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3), 2009.

[2] Mohamed Faouzi Atig. From multi to single stack automata. In Paul Gastin and François Laroussinie, editors, *CONCUR*, volume 6269 of *Lecture Notes in Computer Science*, pages 117–131. Springer, 2010.

[3] Mohamed Faouzi Atig. Global model checking of ordered multi-pushdown systems. In Kamal Lodaya and Meena Mahajan, editors, *FSTTCS*, volume 8 of *LIPIcs*, pages 216–227. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[4] Mohamed Faouzi Atig, Benedik Bollig, and Peter Habermehl. Emptiness of multi-pushdown automata is 2ETIME-complete. In *Proceedings of DLT'08*, volume 5257 of *Lecture Notes in Computer Science*, pages 121–133. Springer, 2008.

[5] Mohamed Faouzi Atig, Ahmed Bouajjani, Michael Emmi, and Akash Lal. Detecting fair non-termination in multithreaded programs. In P. Madhusudan and Sanjit A. Seshia, editors, *CAV*, volume 7358 of *Lecture Notes in Computer Science*, pages 210–226. Springer, 2012.

[6] Mohamed Faouzi Atig, Ahmed Bouajjani, K. Narayan Kumar, and Prakash Saivasan. Linear-time model-checking for multithreaded programs under scope-bounding. In Supratik Chakraborty and Madhavan Mukund, editors, *ATVA*, volume 7561 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2012.

[7] Mohamed Faouzi Atig, K. Narayan Kumar, and Prakash Saivasan. Adjacent ordered multi-pushdown systems. *Int. J. Found. Comput. Sci.*, 25(8):1083–1096, 2014.

[8] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proceedings of CONCUR'97*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 1997.

[9] Luca Breveglieri, Alessandra Cherubini, Claudio Citrini, and Stefano Crespi-Reghizzi. Multi-push-down languages and grammars. *International Journal of Foundations of Computer Science*, 7(3):253–292, 1996.

[10] Luca Breveglieri, Alessandra Cherubini, and Stefano Crespo Reghizzi. personal communication.

[11] Dario Carotenuto, Aniello Murano, and Adriano Peron. Ordered multi-stack visibly pushdown automata. *Theor. Comput. Sci.*, 656:1–26, 2016.

[12] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[13] Bruno Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars*, pages 313–400. World Scientific, 1997.

[14] Aiswarya Cyriac, Paul Gastin, and K. Narayan Kumar. MSO decidability of multi-pushdown systems via split-width. In *Proceedings of CONCUR'12*, volume 7454 of *Lecture Notes in Computer Science*, pages 547–561. Springer, 2012.

[15] Joost Engelfriet. Iterated stack automata and complexity classes. *Information and Computation*, 95(1):21–75, 1991.

[16] Salvatore La Torre, Margherita Napoli, and Gennaro Parlato. A unifying approach for multistack pushdown automata. In *Proceedings of MFCS'14, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 377–389. Springer, 2014.

[17] Salvatore La Torre, Margherita Napoli, and Gennaro Parlato. Scope-bounded pushdown languages. *Int. J. Found. Comput. Sci.*, 27(2):215–234, 2016.

[18] Akash Lal, Tayssir Touili, Nicholas Kidd, and Thomas W. Reps. Interprocedural analysis of concurrent programs under a context bound. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS*, volume 4963 of *Lecture Notes in Computer Science*,

pages 282–298. Springer, 2008.
[19] P. Madhusudan and Gennaro Parlato. The tree width of auxiliary storage. In Thomas Ball and Mooly Sagiv, editors, *POPL*, pages 283–294. ACM, 2011.
[20] Shaz Qadeer and Jakob Rehof. Context-bounded model checking of concurrent software. In Nicolas Halbwachs and Lenore D. Zuck, editors, *TACAS*, volume 3440 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2005.
[21] Pierluigi San Pietro. Two-stack automata. Technical Report 92-073, Dipartimento di elettronica e informazione, Politechnico di Milano, 1992.
[22] Detlef Seese. The structure of models of decidable monadic theories of graphs. *Ann. Pure Appl. Logic*, 53(2):169–195, 1991.
[23] Anil Seth. Games on multi-stack pushdown systems. In *Proceedings of LFCS'09*, volume 5407 of *Lecture Notes in Computer Science*, pages 395–408. Springer, 2009.
[24] Anil Seth. Global reachability in bounded phase multi-stack pushdown systems. In Tayssir Touili, Byron Cook, and Paul Jackson, editors, *CAV*, volume 6174 of *Lecture Notes in Computer Science*, pages 615–628. Springer, 2010.
[25] Larry J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, MIT, 1974.
[26] Salvatore La Torre, P. Madhusudan, and Gennaro Parlato. A robust class of context-sensitive languages. In *Proceedings of LICS*, pages 161–170. IEEE, 2007.
[27] Salvatore La Torre, P. Madhusudan, and Gennaro Parlato. Context-bounded analysis of concurrent queue systems. In *Proceedings of TACAS'08*, Lecture Notes in Computer Science, pages 299–314. Springer, 2008.
[28] Salvatore La Torre, P. Madhusudan, and Gennaro Parlato. An infinite automaton characterization of double exponential time. In *Proceedings of CSL'08*, volume 5213 of *Lecture Notes in Computer Science*. Springer, 2008.
[29] Salvatore La Torre and Margherita Napoli. Reachability of multistack pushdown systems with scope-bounded matching relations. In Joost-Pieter Katoen and Barbara König, editors, *CONCUR*, volume 6901 of *Lecture Notes in Computer Science*, pages 203–218. Springer, 2011.
[30] Salvatore La Torre and Margherita Napoli. A temporal logic for multi-threaded programs. In Jos C. M. Baeten, Thomas Ball, and Frank S. de Boer, editors, *IFIP TCS*, volume 7604 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2012.
[31] Salvatore La Torre and Gennaro Parlato. Scope-bounded multistack pushdown systems: Fixed-point, sequentialization, and tree-width. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *FSTTCS*, volume 18 of *LIPIcs*, pages 173–184. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.