Reachability Problem for Weak Multi-Pushdown Automata

Wojciech Czerwiński, Piotr Hofman, and Sławomir Lasota*

Institute of Informatics, University of Warsaw
{wczerwin,sl}@mimuw.edu.pl

Abstract. This paper is about reachability analysis in a restricted subclass of multi-pushdown automata: we assume that the control states of an automaton are partially ordered, and all transitions of an automaton go downwards with respect to the order. We prove decidability of the reachability problem, and computability of the backward reachability set. As the main contribution, we identify relevant subclasses where the reachability problem becomes NP-complete. This matches the complexity of the same problem for communication-free vector addition systems (known also as commutative context-free graphs), a special case of stateless multi-pushdown automata.

1 Introduction

This paper is about reachability analysis of *multi-pushdown systems*, i.e., systems with a global control state and multiple stacks. The motivation for our work is twofold. On one side, a practical motivation coming from context-bounded analysis of recursive concurrent programs [23, 20, 3]. On the other side, a theoretical motivation coming from partially-commutative context-free grammars, developed recently in [11–13].

Context bounded analysis. Multi-pushdown systems may be used as an abstract model of concurrent programs with recursive procedures. As multi-pushdown systems are a Turing-complete model of computation, they are only applicable for verification under further tractable restrictions. One remarkably successful restriction is imposing a bound on the number of context switches; between consecutive context switches, the system may only perform operations on one stack (local operations). In [23], the context-bounded reachability has been shown decidable, by reduction to reachability of ordinary pushdown systems [5]. This line of research, with applications in formal verification, has been continued successfully, e.g., in [6, 20, 3].

Weak control states. As our starting point we observe that if the number of context switches is bounded, one may safely assume that the control state space is *weak*, in the sense that there is a partial order on control states such that transitions go only downwards with respect to the order. Indeed, the local state space of every thread may be eliminated using a stack, and the global control state essentially enumerates context

* The first author acknowledges a partial support by the Polish MNiSW grant N N206 568640. The other authors acknowledge a partial support by the Polish MNiSW grant N N206 567840.

M. Koutny and I. Ulidowski (Eds.): CONCUR 2012, LNCS 7454, pp. 53–68, 2012.
© Springer-Verlag Berlin Heidelberg 2012

switches. Roughly speaking, the model investigated in this paper extends the above one with respect to operations allowed between two context switches, namely, we do not restrict these operations to one stack only. Thus, if k is the number of stacks, we assume that transitions of a system are of the following form:

$$q,\ \mathrm{X} \xrightarrow{a} q',\ \alpha_1, \ldots, \alpha_k, \tag{1}$$

to mean that in state q, symbol X is popped from one of the stacks, and sequences of symbols $\alpha_1, \ldots, \alpha_k$, respectively, are pushed on stacks. Wlog. one may assume that the symbols of different stacks are different.

Partially-commutative context-free grammars. A special case of the model investigated in this paper is *stateless* multi-pushdown systems. This is still a quite expressible model that subsumes, among the others, context-free graphs (so called Basic Process Algebra [8]) and communication-free Petri nets (so called Basic Parallel Processes [8]). In the stateless case, transitions (1) may be understood as productions of a grammar, with the nonterminal symbols on the right-hand side (stack symbols) subject to a commutativity law. More precisely, for any two symbols X and Y from different stacks, we impose the commutativity law

$$\mathrm{XY} \ = \ \mathrm{YX}.$$

One easily observes that this is a special case of *independence relation* over nonterminal symbols, as defined in trace theory [14][1]. In multi-pushdown systems, the *dependency relation* (complement of independence relation) is always transitive. A general theory of context-free grammars modulo dependency relation that is not necessarily transitive, has been studied recently in [13]; complexity of bisimulation equivalence checking has been investigated in [11, 12]. The present paper complements these results by focusing on reachability analysis.

Contributions. This paper contains two main results. First, we prove decidability of reachability for weak multi-pushdown automata. Our argument is based on a suitable well order on the set of configurations, that strongly depends on the assumption that the control states are weak.

Second, we identify additional restrictions under which the problem is NP-complete; one such restriction is stateless multi-pushdown systems. Our result subsumes (and gives a simpler algorithm for) the case of communication-free Petri nets; reachability thereof is NP-complete as shown in [15]. The last result is similar to NP-completeness of the word problem for partially-commutative context-free grammars [16], where one asks if the given input word is accepted. The reachability question is more difficult to answer, as an input word is not given in advance. In fact the main technical difficulty is to show existence of a polynomial witness for reachability.

As further results, we investigate forward and backward reachability sets, and prove that the backward reachability set of a regular set of configurations is regular and computable, while the forward reachability set needs not be regular in general. Finally,

[1] Note however that the independence is imposed on nonterminal symbols, and not on input letters, as usually in trace theory.

we identify the decidability border for reachability of weak multi-pushdown systems. Roughly speaking, the problem becomes undecidable when one asks about reachability of a given regular set of configurations, instead of a single configuration.

The standard techniques useful for analysis of pushdown systems, such as pumping or the automaton-based approach of [5], do not extend to the multi-pushdown setting. This is why the proofs of our results are based on new insights. The NP-membership proofs are, roughly speaking, based on polynomial witnesses obtained by careful elimination of 'irrelevant' transitions. On the other hand, the decidability results are based on a suitable well order on configurations.

Related research. Multi-pushdown systems are a fundamental model of recursive multi-threaded programs. This is why different instantiations of the multi-pushdown paradigm have been appearing in the literature recently, most often in the context of formal verification. We only mention here a few relevant positions we are aware of, without claiming completeness. All the papers cited below bring some restricted decidability results for reachability or model checking.

Most often, a model has global control states, subject to some restriction. For instance, the author of [1] assumes that the stacks are ordered, and pop operation can only be performed on the first nonempty stack. Another example is the model introduced in [7] and then further investigated e.g. in [6, 2, 4], that allows for unbounded creation of new stacks; on the other hand, operations on each stack are local, thus no communication between threads is allowed.

Another possible approach is to replace global state space with some communication mechanism between threads. Some successful results on analysis of multi-threaded programs communicating via locks, in a restricted way, has been reported in [18, 17, 9].

In [21] the algorithm for reachability over PA [8] graphs has been provided. The PA class is a generalization of both BPA and BPP that allows, similarly like multi-pushdown systems, both for sequential and interleaved behavior. Finally, in [19] the reachability problem has been shown decidable for Process Rewrite Systems [22] extended with weak control states.

Outline. In the following Section 2 we define the model we work with. Then in Section 3 we state all our results. In the remaining sections we provide proofs of some of the results. The other proofs are omitted due to space limitation.

2 Multi-pushdown Automata

A multi-pushdown automaton (MPDA) is like a single-pushdown one. In a single step one symbol is popped from one of stacks[2], and a number of symbols are pushed on the stacks. Assume there is k stacks. A transition of an automaton is thus of the form:

$$q,\ \mathrm{X} \xrightarrow{a} q',\ \alpha_1, \ldots, \alpha_k, \tag{2}$$

[2] If we allowed for popping from more than one stack at a time, the model would clearly become Turing-complete, even with 1 state only.

to mean that when an automaton reads a in state q, it pops X from one of the stacks, pushes the sequence of symbols α_i on the ith stack, for $i = 1 \dots k$, and goes to state q'. We allow for silent transitions with $a = \varepsilon$. Observe that wlog. one may assume that stack alphabets are disjoint.

Formally, the ingredients of a MPDA are: a finite set of states Q, the number of stacks k, pairwise-disjoint finite stack alphabets $\mathsf{S_1} \dots \mathsf{S_k}$, an input alphabet A, and a finite set of transition rules:

$$\longrightarrow \subseteq \mathsf{Q} \times (\bigcup_{i \leq k} \mathsf{S_i}) \times (\mathsf{A} \cup \{\varepsilon\}) \times \mathsf{Q} \times \mathsf{S_1}^* \times \ldots \times \mathsf{S_k}^* \tag{3}$$

written as in (2). A configuration of a MPDA is a tuple $\langle q, \beta_1, \dots, \beta_k \rangle \in \mathsf{Q} \times \mathsf{S_1}^* \times \ldots \times \mathsf{S_k}^*$. The transition rules (2) induce the transition relation over all configurations in a standard way:

$$\frac{q,\ \mathrm{X} \xrightarrow{a} q',\ \alpha_1, \dots, \alpha_k \qquad \mathrm{X} \in \mathsf{S_i} \qquad \beta_i = \mathrm{X}\beta}{\langle q, \beta_1, \dots \beta_i \dots, \beta_k \rangle \xrightarrow{a} \langle q', \alpha_1\beta_1, \dots \alpha_i\beta \dots, \alpha_k\beta_k \rangle}$$

thus defining the configuration graph of a MPDA. For a configuration $\langle q, \alpha_1, \dots, \alpha_k \rangle$, its *size* is defined as the sum of lengths of the words α_i, $i \leq k$. The same applies to a right-hand side of any transition rule $q\,\mathrm{X} \xrightarrow{a} q'\,\alpha_1 \dots \alpha_k$.

An MPDA is *stateless* if there is just one state (or equivalently no states). Transition rules of an automaton are then of the form:

$$\mathrm{X} \xrightarrow{a} \alpha_1, \dots, \alpha_k \tag{4}$$

and configurations are of the form $\langle \beta_1, \dots, \beta_k \rangle$.

A less severe restriction on control states is the following one. We say that an automaton is *weak* if there is a partial order $\leq$ on its states such that every transition (2) satisfies $q' \leq q$. Clearly, every stateless automaton is weak.

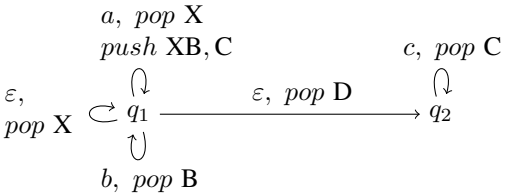
Remark 1. Note that stateless one-stack automata are essentially context-free grammars in Greibach normal form. Thus the configuration graphs are precisely context-free graphs, called also BPA graphs [22, 8]. Another special case is many stacks with singleton alphabets. The stacks are thus essentially counters without zero tests. In this subclass, stateless automata corresponds to communication-free Petri nets [15], called also BPP [10], or commutative context-free graphs [12]. The BPA and BPP classes are members of the Process Rewrite Systems hierarchy of [22] that contains, among the others, pushdown systems and unrestricted Petri nets.

Example 1. Assuming a distinguished initial state and acceptance by all stacks empty, weak MPDAs can recognize non-context-free languages. For instance, the language

$$\{a^n b^n c^n : n \geq 0\} \tag{5}$$

is recognized by an automaton described below. The automaton has two states q_1, q_2 and two stacks. The alphabets of the stacks are $\{\mathrm{X}, \mathrm{B}, \mathrm{D}\}$ and $\{\mathrm{C}\}$, respectively. The

starting configuration is $(q_1, \mathrm{XD}, \varepsilon)$. Besides the transition rules, we also present the automaton in a diagram, using *push* and *pop* operations with natural meaning.



$$
\begin{aligned}
q_1,\ \mathrm{X} &\xrightarrow{a} q_1,\ \mathrm{XB},\ \mathrm{C}\\
q_1,\ \mathrm{X} &\xrightarrow{\varepsilon} q_1,\ \varepsilon,\ \varepsilon\\
q_1,\ \mathrm{B} &\xrightarrow{b} q_1,\ \varepsilon,\ \varepsilon\\
q_1,\ \mathrm{D} &\xrightarrow{\varepsilon} q_2,\ \varepsilon,\ \varepsilon\\
q_2,\ \mathrm{C} &\xrightarrow{c} q_2,\ \varepsilon,\ \varepsilon
\end{aligned}
$$

The automaton is weak and uses ε-transitions, which may be however easily eliminated. Acceptance by empty stacks may be easily simulated using acceptance by states. The language (5) is not recognized by a stateless automaton, as shown in [13].

Example 2. Non-context-free languages are recognized even by stateless MPDAs with singleton stack alphabets. The class of languages recognized by this subclass is called *commutative context-free* languages [16], see also [13]. One example is the commutative closure of the language of the previous example: the set of all words with the same number of occurrences of a, b and c.

In the sequel we do not care about initial states nor about acceptance condition, as we will focus on the configuration graph of an automaton. Furthermore, as we only consider reachability problem, the labeling of transitions with input alphabet letters will be irrelevant, thus we write $\longrightarrow$ instead of $\xrightarrow{a}$ from now on.

Using a standard terminology, we say that a MPDA is *normed* if for any state q and any configuration $\langle q, \alpha_1, \ldots, \alpha_k\rangle$, there is a path to the empty configuration

$$\langle q, \alpha_1, \ldots, \alpha_k\rangle \longrightarrow \ldots \longrightarrow \langle p, \varepsilon, \ldots, \varepsilon\rangle$$

for whatever state p. In general, whenever a MPDA is not assumed to be normed we call it *unnormed* for clarity. Note that in all examples above the automata were normed. In fact normedness is not a restriction as far as languages are considered. In the sequel we will however analyze the configuration graphs, and then normedness will play a role.

Further, we say that a MPDA is *strongly normed* if for any state q and any configuration $\langle q, \alpha_1, \ldots, \alpha_k\rangle$, there is a path to the empty configuration

$$\langle q, \alpha_1, \ldots, \alpha_k\rangle \longrightarrow \ldots \longrightarrow \langle q, \varepsilon, \ldots, \varepsilon\rangle$$

containing only transitions that do not change state. Intuitively, whatever is the state q we start in, any top-most symbol X in any stack may „disappear". For stateless automata, strong normedness is the same as normedness.

3 Reachability

Regular Sets. We will consider various reachability problems in the configuration graph of a given MPDA. Therefore, we need a finite way of describing infinite sets

of configurations. A standard approach is to consider *regular* sets. Below we adapt this approach to the multi-stack scenario we deal with.

Consider the configurations of a stateless MPDA, $\mathsf{S} = \mathsf{S_1}^* \times \ldots \times \mathsf{S_k}^*$. There is a natural monoid structure in S, with pointwise identity $\langle \varepsilon, \ldots, \varepsilon \rangle$ and multiplication

$$\langle \alpha_1, \ldots, \alpha_k \rangle \cdot \langle \beta_1, \ldots, \beta_k \rangle \;=\; \langle \alpha_1\, \beta_1, \ldots, \alpha_k\, \beta_k \rangle.$$

Call a subset $L \subseteq \mathsf{S}$ *regular* if there is a finite monoid M and a monoid morphism $\gamma : \mathsf{S} \to M$ that *recognizes* L, which means that $L = \gamma^{-1}(N)$ for some subset $N \subseteq M$. Without loss of generality one may assume that the monoid M is a product of finite monoids $M = M_1 \times \ldots \times M_k$, and that

$$\gamma = \gamma_1 \times \ldots \times \gamma_k \quad \text{where} \quad \gamma_i : \mathsf{S_i}^* \to M_i \;\text{ for } i = 1 \ldots k.$$

Thus we may use an equivalent but more compact representation of regular sets, based on automata: a regular set L is given by a tuple of (nondeterministic) finite automata $\mathcal{B}_1 \ldots \mathcal{B}_k$ over alphabets $\mathsf{S_1} \ldots \mathsf{S_k}$, respectively, together with a set

$$F \subseteq Q_1 \times \ldots \times Q_k$$

of accepting tuples of states, where Q_i denotes the state space of automaton $\mathcal{B}_i$.

Unless stated otherwise, in the sequel we always use such representations of regular sets of configurations. If there are more than one state, we assume a representation for every state. In particular, when saying "polynomial wrt. L", for a regular language L, we mean polynomial wrt. the sum of sizes of automata representing L.

Remark 2. Clearly, the cardinality of the set F of accepting tuples may be exponential wrt. the cardinalities of state spaces of automata $\mathcal{B}_i$. However, complexities we derive in the sequel will never depend on cardinality of F.

Example 3. Assume that there are two stacks. An example of properties we can define is: „odd number of elements on the first stack and symbol A on the top of the second stack, or an even number of the elements on the first stack and the odd number of elements on the second stack". On the other hand, "all stacks have equal size" is not a regular property according to our definition.

Remark 3. We have deliberately chosen a notion of regularity of languages of *tuples* of words. Another possible approach could be to consider regular languages of words, over the product alphabet $(\mathsf{S_1} \cup \bot) \times \ldots \times (\mathsf{S_k} \cup \bot)$, where the additional symbol $\bot$ is necessary for padding. This would yield a larger class, for instance the last language from Example 3 would be regular. The price to pay would be however undecidability of the reachability problems. The undecidability will be discussed below.

Reachability. In this paper we consider the following reachability problem:

INPUT: a MPDA $\mathcal{A}$ and two regular sets of configurations $L, K \subseteq \mathsf{S}$.
QUESTION: is there a path in the configuration graph from L to K?

We will write $L \rightsquigarrow_{\mathcal{A}} K$ if a path from L to K exists in the automaton $\mathcal{A}$. The sets L and K we call *source* and *target* sets, respectively. We will distinguish special cases, when either L or K or both the sets are singletons, thus obtaining four different variants of reachability altogether. For brevity we will use symbol '*1*' for a singleton, and symbol 'REG' for a regular set, and speak of *1*$\rightsquigarrow$REG *reachability* (when L is a singleton), REG$\rightsquigarrow$REG *reachability* (the unrestricted case), and likewise for REG$\rightsquigarrow$*1* and *1*$\rightsquigarrow$*1*.

Before stating the results, we note that all the problems we consider here are NP-hard:

Lemma 1. *The 1$\rightsquigarrow$1 reachability is NP-hard for strongly normed stateless MPDAs, even if all stack alphabets are singletons.*

The above fact follows immediately from NP-completeness of the reachability problem for communication-free Petri nets, see [15] for details.

Results. In presence of states, the 1$\rightsquigarrow$1 reachability problem is obviously undecidable, because the model is Turing powerful. Undecidability holds even for normed MPDAs. We will thus consider only stateless or weak MPDAs from now on.

We start by observing that out of four combinations of the reachability problem, it is sufficient to consider only two, namely the REG$\rightsquigarrow$1 and REG$\rightsquigarrow$REG cases. Indeed, as far as complexity is concerned, we observe the following collapse:

$$1\rightsquigarrow 1 \;=\; \text{REG}\rightsquigarrow 1 \qquad\qquad 1\rightsquigarrow \text{REG} \;=\; \text{REG}\rightsquigarrow\text{REG} \tag{6}$$

independently of a restriction on automata. The first equality follows from our first result:

Lemma 2. *Suppose $\mathcal{A}$ is a weak MPDA. Let L be a regular set of configurations of $\mathcal{A}$ and let t be a configuration of $\mathcal{A}$. Then*

$$L \rightsquigarrow_{\mathcal{A}} t \implies s \rightsquigarrow_{\mathcal{A}} t \text{ for some } s \in L \text{ of size polynomial wrt. } \mathcal{A}, L \text{ and } t.$$

Indeed, the reduction from REG$\rightsquigarrow$1 to 1$\rightsquigarrow$1 is by nondeterministic guessing a source configuration of polynomial size.

The second equality (6) will follow from our results listed below.

Before stating the remaining results, we summarize all of them in the following table. We distinguish cases, corresponding to strongly normed/normed/unnormed case and stateless/weak case. Each entry of the table contains the complexity of REG$\rightsquigarrow$REG reachability problem. Additionally, the complexity of REG$\rightsquigarrow$1 reachability problem is given in cases it is different from the complexity of REG$\rightsquigarrow$REG reachability.

For clarity, we do not distinguish stateless strongly normed case from stateless normed one, as these two cases obviously coincide.

[REG$\rightsquigarrow$1] REG$\rightsquigarrow$REG	strongly normed	normed	unnormed
stateless	NP-compl. (Thm. 2)		[NP-compl. (Thm. 3)] undecidable (Thm. 1)
weak	NP-compl. (Thm. 2)	[decidable] undecidable (Thm. 1)	[decidable (Thm. 4)] undecidable

Now we discuss the results in detail. We first observe an apparent decidability frontier witnessed by stateless unnormed MPDAs and weak normed MPDAs:

Theorem 1. *The $1 \rightsquigarrow \mathrm{REG}$ reachability is undecidable for stateless unnormed MPDAs and for weak normed MPDAs.*

The proof is by reduction of the nonemptiness of intersection of context-free languages and uses three stacks. The case of two stacks remains open.

Thus lack of strong normedness combined with a regular target set yields undecidability in case of stateless automata. Surprisingly, restricting additionally:

- either the automaton to be strongly normed,
- or the target set to a singleton,

makes a dramatical difference for complexity of the problem, as summarized in Theorems 2, 3 and 4 below. In the first theorem we only assume strong normedness:

Theorem 2. *The $\mathrm{REG} \rightsquigarrow \mathrm{REG}$ reachability is NP-complete for strongly normed weak MPDAs.*

Theorem 2 is the main result of this paper. It is proved by showing that reachability is always witnessed by a polynomial witness, obtained by careful elimination of 'irrelevant' transitions.

In the following two theorems we do not assume strong normedness, thus according to Theorem 1 we have to restrict target set to singleton. Under such a restriction, we are able to prove NP-completeness only in the class of stateless MPDA, while for all weak MPDA we merely state decidability:

Theorem 3. *The $\mathrm{REG} \rightsquigarrow 1$ reachability is NP-complete for stateless unnormed MPDAs.*

Theorem 4. *The $\mathrm{REG} \rightsquigarrow 1$ reachability is decidable for weak unnormed MPDAs.*

Theorem 3 is shown similarly to Theorem 2, while the proof of Theorem 4 is based on a well order, the point-wise extension of a variant of Higman ordering.

Open Questions. Except for two entries in the summarizing table above, we know the exact complexity of the reachability problem. The important open question that remains is the actual complexity of $1 \rightsquigarrow 1$ reachability for (normed and unnormed) weak MPDAs. Another interesting question is whether undecidability carries over to automata with two stacks only.

Reachability Set. Now we consider the problem of computing the whole reachability set. For a given automaton $\mathcal{A}$, and a set L of configurations, we consider forward and backward reachability sets of L, defined as:

$$\{s : L \rightsquigarrow_{\mathcal{A}} s\} \qquad \text{and} \qquad \{s : s \rightsquigarrow_{\mathcal{A}} L\},$$

respectively. It turns out that the backward reachability set may be computed under the strong normedness assumption.

Theorem 5. *For weak strongly normed MPDAs, the backward reachability set of a regular set is an effectively computable regular set.*

Roughly speaking, we show that the backward reachability set is upward closed with respect to the point-wise extension of a suitable variant of Higman ordering.

On the other hand, the forward reachability set needs not be regular, even in the case of strongly normed stateless automata, as shown in the following example.

Example 4. Consider a stateless automaton with two stacks, over alphabets $\{\mathrm{A}, \mathrm{X}\}$ and $\{\mathrm{B}\}$, and the following transition rules:

$$\mathrm{X} \longrightarrow \mathrm{XA},\ \mathrm{B} \qquad \mathrm{X} \to \varepsilon,\ \varepsilon \qquad \mathrm{A} \to \varepsilon,\ \varepsilon \qquad \mathrm{B} \to \varepsilon,\ \varepsilon.$$

The set of configurations reachable from the configuration $(\mathrm{X}, \varepsilon)$ is not regular:

$$\{(\mathrm{A}^i, \mathrm{B}^j) : i, j \in \mathbb{N}\} \ \cup\ \{(\mathrm{XA}^k, \mathrm{B}^l) : k \geq l\}.$$

Relaxed Regularity. The relaxed definition of regularity, as discusses in Remark 3, makes the reachability problem intractable in all cases. The following theorem is shown by reduction from the Post Correspondence Problem:

Theorem 6. *The $1{\rightsquigarrow}\mathrm{REG}$ reachability is undecidable for stateless strongly normed MPDAs, under the relaxed notion of regularity.*

Furthermore, the backward reachability set of a relaxed regular set is not necessarily regular, even in stateless strongly normed MPDAs. The illustrating example is omitted due to space limitation.

4 Proof of Lemma 2

Consider a MPDA $\mathcal{A}$ and a regular set L of configurations of $\mathcal{A}$. Let $s \in L$ be source configuration and let t be an arbitrary target configuration. Suppose $s \rightsquigarrow_{\mathcal{A}} t$. We will show that the size of s may be reduced, while preserving membership in L. The crucial but simple idea of the proof will rely on an analysis of *relevance* of symbol occurrences, to be defined below.

Symbol occurrences. Suppose that there is a path π from s to t, consisting of consecutive transitions $s \longrightarrow s_1 \longrightarrow s_2 \ldots \longrightarrow s_n = t$. We will consider all individual occurrences of symbols that appear in the configurations. For instance, in the following exemplary sequence of two-stack configurations

$$\langle q, \mathrm{AA}, \mathrm{C}\rangle \longrightarrow \langle q, \mathrm{BBA}, \mathrm{DC}\rangle \longrightarrow \langle q, \mathrm{ABBA}, \mathrm{DC}\rangle \tag{7}$$

there are altogether 14 *symbol occurrences*: 3 in the first configuration, 5 in the second one and 6 in the third one.

Recall that every transition $s_i \longrightarrow s_{i+1}$ is induced by some transition rule $\mathrm{X} \longrightarrow \alpha$ of the automaton. Then there is a distinguished occurrence of symbol X in s_i that is

involved in the transition. In the sequel we use the term *symbol occurrence involved in a transition*.

Precisely one occurrence of symbol in s_i is involved in the transition $s_i \longrightarrow s_{i+1}$; for every other occurrence of a symbol in s_i there is a *corresponding* occurrence of the same symbol in s_{i+1}. (Note that we always make a difference between corresponding symbol occurrences from different configurations.) All remaining occurrences of symbols in s_{i+1} are created by the transition; we call these occurrences *fresh*.

We define the *descendant* relation as follows. All fresh symbol occurrences in s_{i+1} are descendants of the symbol occurrence in s_i involved in the transition $s_i \longrightarrow s_{i+1}$. Moreover, a symbol occurrence in s_{i+1} corresponding to a symbol occurrence in s_i is its descendant too. We will use term *descendant* for the reflexive-transitive closure of the relation defined above and the term *ancestor* for its inverse relation. In particular, every symbol occurrence in t is descendant of a unique symbol occurrence in s. The descendant relation is a forest, i.e., a disjoint union of trees.

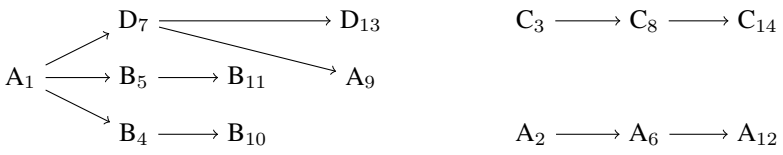
Example 5. As an example, consider again the sequence of transitions (7), with symbol occurrences identified by subscripts $1 \ldots 14$:

$$\langle q, \mathrm{A}_1\mathrm{A}_2, \mathrm{C}_3\rangle \longrightarrow \langle q, \mathrm{B}_4\mathrm{B}_5\mathrm{A}_6, \mathrm{D}_7\mathrm{C}_8\rangle \longrightarrow \langle q, \mathrm{A}_9\mathrm{B}_{10}\mathrm{B}_{11}\mathrm{A}_{12}, \mathrm{D}_{13}\mathrm{C}_{14}\rangle \quad (8)$$

Say the transitions are induced by the following two transition rules:

$$q,\ \mathrm{A} \longrightarrow q,\ \mathrm{BB},\ \mathrm{D} \qquad\qquad q,\ \mathrm{D} \longrightarrow q,\ \mathrm{A},\ \mathrm{D}$$

The descendant relation can be presented as the following forest:



The symbol occurrences involved in the two transitions (8) are A_1 in the first configuration and D_7 in the second one. The fresh symbol occurrences are B_4, B_5 and D_7 in the second configuration, and A_9 and D_{13} in the third one.

Relevant symbol occurrences. As the automaton $\mathcal{A}$ is weak, the number of transitions in π that change state is bounded by the number of states of $\mathcal{A}$. All remaining transitions in π do not change state.

Consider all the occurrences of all symbols in all configurations along the path π, including configurations s and t themselves. A symbol occurrence is called *relevant* if some of its descendants:

- belongs to the target configuration t; or
- is involved in some transition in π that changes state.

Otherwise, a symbol occurrences is *irrelevant*. In particular, all symbol occurrences in t are relevant. Referring back to our example, all symbol occurrences appearing in (8) are relevant.

Note that if t is not the empty configuration then every configuration in π contains at least one relevant symbol occurrence. On the other side, in every configuration, the number of relevant occurrences is always bounded by the sum of the size of t and the number of states of $\mathcal{A}$.

Small source configuration. So prepared, we are ready to prove that there is a configuration $s' \in L$ of polynomial size with $s' \rightsquigarrow_{\mathcal{A}} t$. We will rely on the following claim:

Lemma 3. *For any configuration s' obtained from s by removing some irrelevant symbol occurrences, it holds $s' \rightsquigarrow_{\mathcal{A}} t$.*

The lemma follows from the following two observations: (1) all the transitions in π involving symbol occurrences remaining in s' and their descendants may be re-done; (2) the resulting configuration will be exactly t, as only irrelevant symbol occurrences have been removed from s.

Recall that the language L is represented by a tuple $\mathcal{B}_1 \dots \mathcal{B}_k$ of deterministic finite automata, one automaton per stack. Consider the content of a fixed ith stack in s, say $w \in \mathsf{A}_i^*$. Let n be the number of states of $\mathcal{B}_i$. The run of the automaton $\mathcal{B}_i$ over w labels every position of w by some state. We will use a standard pumping argument to argue that every block of consecutive irrelevant symbol occurrences in s may be reduced in length to at most n. Indeed, upon every repetition of a state of $\mathcal{B}_i$, the word w may be shortened by removing the induced infix, while preserving membership in L. By repeating the pumping argument for all blocks of consecutive irrelevant symbol occurrences in all stacks in s, one obtains a configuration s', still belonging to L, of quadratic size. By Lemma 3 we know that $s' \rightsquigarrow t$, as required.

5 Proof of Theorem 2

NP-hardness follows from Lemma 1. The proof of membership in NP relies on the following two core lemmas:

Lemma 4. *The 1$\rightsquigarrow$1 reachability problem is in NP for strongly normed weak MPDAs.*

Lemma 5. *Let $\mathcal{A}$ be a strongly normed weak MPDA and let L, K be regular sets of configurations. If $L \rightsquigarrow K$ then $s \rightsquigarrow t$ for some $s \in L$ and $t \in K$ of size polynomial wrt. the sizes of $\mathcal{A}$, L and K.*

The two lemmas easily yield a decision procedure for REG$\rightsquigarrow$REG reachability: guess configurations $s \in L$ and $t \in K$ of size bounded by a polynomial deduced from the proof of Lemma 5, and then apply the procedure of Lemma 4 to check if $s \rightsquigarrow t$.

The rest of this section is devoted to the part of the proof of Lemma 4. The remaining part of the proof, together with the proof of Lemma 5, are omitted.

5.1 Proof of Lemma 4

Consider a MPDA $\mathcal{A}$ and two configurations s and t. We will define a nondeterministic polynomial-time decision procedure for $s \rightsquigarrow_{\mathcal{A}} t$.

Stateless assumption. For simplicity, we assume that both s and t have the same control state. Thus we can treat transitions that lead from s to t as stateless transitions. At the very end of the proof, we will discuss how to generalize it to the general case of strongly normed weak MPDAs.

Polynomial witness. Our aim is to show that if there is a path from s to t then there is a path of polynomial length. So stated, the above claim may not be verbally true, even in the case of context-free graphs, as witnessed by the following simple example.

$$\mathrm{X}_1 \longrightarrow \mathrm{X}_2\mathrm{X}_2 \qquad \mathrm{X}_2 \longrightarrow \mathrm{X}_3\mathrm{X}_3 \qquad \dots \qquad \mathrm{X}_{n-1} \longrightarrow \mathrm{X}_n\mathrm{X}_n \qquad \mathrm{X}_n \longrightarrow \varepsilon \tag{9}$$

The example scales with respect to n, and thus the shortest path from the configuration X_1 to X_n is of exponential length. As a conclusion, one must use some subtle analysis in order to be able to reduce the length of a witness of existence of the path as required. Note that X_1 is relevant and thus can not be simply omitted.

Proof idea. As a first step towards a polynomial bound on the witness of the path from s to t, we will modify the notion of transition. Intuitively speaking, our aim is to consider exclusively relevant symbol occurrences.

By a *subword* we mean any subsequence of a given word. For instance, $aaccbc$ is a subword of $aacabbcbcbc$. Further, by a subtransition of $\mathrm{X} \longrightarrow \alpha_1 \dots \alpha_k$ we mean any $\mathrm{X} \longrightarrow \beta_1 \dots \beta_k$ such that the following conditions hold:

- *subword*: β_i is a subword of α_i, for all $i \in \{1 \dots k\}$; and
- *nonemptiness*: $\beta_1 \dots \beta_k \neq \varepsilon$, i.e., at least one of words β_i is nonempty.

Note that relying on the notion of relevance one easily deduces that whenever there is a sequence of transitions from s to t, then there is also sequence of *subtransitions*. Indeed, it is sufficient to remove irrelevant symbol occurrences in all transitions along the path from s to t.

Clearly, the converse implication is not true in general. For instance, if we add symbols X_0, A and the transition $\mathrm{X}_0 \longrightarrow \mathrm{X}_1\mathrm{A}$ to the example (9), there is a sequence of subtransitions from the configuration X_0 to X_n. Our aim now it to modify the notion of subtransition in such a way that the converse implication does hold as well, i.e., that existence of a sequence of subtransitions implies existence of a sequence of transitions. This requires certain amount of boring book-keeping, as defined in detail below.

Marked subtransitions. We will need an additional copy of every stack alphabet A_i, denoted by $\bar{\mathsf{A}}_i$, for $i = 1 \dots k$. Thus for every $a \in \mathsf{A}_i$ there is a corresponding marked symbol $\bar{a} \in \bar{\mathsf{A}}_i$. Formally, let the ith stack alphabet be $\mathsf{A}_i \cup \bar{\mathsf{A}}_i$.

A *marked subword* of a word $w \in \mathsf{A}_i^*$ is any word in $(\mathsf{A}_i \cup \bar{\mathsf{A}}_i)^*$ that may be obtained from w by the following *marking procedure*:

- color arbitrary occurrences in w (the idea is to color irrelevant symbol occurrences),
- mark every occurrence that is followed by any colored occurrence,
- and finally remove colored occurrences.

For instance, according to the coloring $a\mathbf{ac}a\mathbf{bb}c\mathbf{b}cbc$, a marked subword of $aacabbcbcbc$ is $\bar{a}\bar{a}\bar{c}cbc$.

Recall that a word $w \in \mathsf{A}_i^*$ represents a content of the ith stack, with the left-most symbol being the top-most. Intuitively, the idea behind the notion of marked subword is to keep track of removed occurrences that are covered by other symbols on the stack.

A notion of *marked subtransition* is a natural adaptation of the notion of subtransition. Compared to subtransitions, there are two differences: 'subword' is replaced with 'marked subword'; and whenever the left-side symbol is marked, then it may only put marked symbols on its stack. Formally, a marked subtransition of $\mathrm{X} \longrightarrow \alpha_1 \dots \alpha_k$ is any $\mathrm{X} \longrightarrow \beta_1 \dots \beta_k$ such that the following conditions hold:

- *subword*: β_i is a marked subword of α_i, for all $i \in \{1 \dots k\}$;
- *nonemptiness*: $\beta_1 \dots \beta_k \neq \varepsilon$, i.e., at least one of words β_i is nonempty; and
- *marking inheritance*: if $X \in \bar{\mathsf{A}}_i$ is marked then all symbols in β_i are marked.

Note that there are exponentially many different marked subtransitions, but each one is of polynomial size. Finally, note that every subtransition is obtained from some transition by the marking procedure as above, applied to every stack separately.

By the nonemptiness assumption on marked subtransitions we obtain a simple but crucial observation:

Lemma 6. *Along a sequence of marked subtransitions, the size of configuration can not decrease.*

A *marked subconfiguration* of a configuration $\langle \alpha_1, \dots, \alpha_k \rangle$ is any tuple $\langle \beta_1, \dots, \beta_k \rangle$ such that β_i is a marked subword of α_i for all $i \in \{1 \dots k\}$.

Lemma 7. *For two configurations s and t, the following conditions are equivalent:*

(1) there is a sequence of transitions from s to t,
(2) there is a sequence of marked subtransitions from u to t, for some marked subconfiguration u of s.

Proof. The implication from (1) to (2) follows immediately. The sequence of marked subtransitions is obtained by application of the marking procedure to all transitions. For every transition, color in the marking procedure precisely those symbol occurrences that are irrelevant.

Now we show the implication from (2) to (1). The proof uses strong normedness.

Assume a sequence π of marked subtransitions from u to t, for some marked subconfiguration u of s. Recall that each subtransition in π has its original transition of $\mathcal{A}$. We claim that there is a sequence of transitions from s to t, that contains the original transitions of all the marked subtransitions appearing in π, and *canceling sequences*

$$q\,\mathrm{X} \longrightarrow \dots \longrightarrow \langle q, \varepsilon, \dots, \varepsilon \rangle \tag{10}$$

for some stack symbols X, existing due to strong normedness assumption.

The sequence of transitions from s to t is constructed by reversing the marking procedure. For the ease of presentation, beside letters from A_i, we will also use colored letters.

Start with the configuration s, and choose any coloring of symbol occurrences in s that induces u as the outcome of the marking procedure. Then consecutively apply the following rule:

- If the top-most symbol X on some stack is colored, apply a canceling sequence for X.
- Otherwise, apply the original transition of the next subtransition from π, using again some coloring that could have been used in the marking procedure.

For correctness, we need to show that all colored occurrences of symbols are eventually canceled out, as this guarantees that the final configuration is precisely t.

Let's inspect π. As no symbol in t is marked, every marked symbol occurrence eventually disappears as a result of firing of some subtransition. Recall that marking of a symbol $\bar{\mathrm{X}}$ disappears only if the subtransition pushes nothing on the stack of $\bar{X}$. As a consequence, every colored symbol occurrence will eventually appear on the top of its stack. Thus the canceling sequence for X will be eventually applied. □

Lemma 8. *For two configurations u and v, if there is a sequence of marked subtransitions from u to v, then there is such a sequence of polynomial length wrt. the sizes of u, v and $\mathcal{A}$.*

This is the last lemma needed for NP-membership. Its proof is omitted.

Decision procedure. Now we drop the stateless assumption. Note that the notion of marked subconfiguration and marked subtransition may be easily adapted to transitions that change state. We do not impose however the nonemptiness condition on transitions that change state, which is in accordance with the intuition that irrelevant symbol occurrences are removed in the marking procedure. Using Lemmas 6, 7 and 8 we will define the nondeterministic decision procedure for strongly normed weak MPDAs.

Let the two given configurations s and t have control states q and p, respectively. In the first step, the algorithm guesses a number of marked subconfigurations $t_1 \dots t_{n-1}$, where n is not greater than the number of states of $\mathcal{A}$, and marked subtransitions that change state:

$$t_1 \longrightarrow s_1 \qquad t_2 \longrightarrow s_2 \qquad \dots \qquad t_{n-1} \longrightarrow s_{n-1}$$

such that s_i and t_{i+1} have the same control states for $i \in \{0 \dots n-1\}$. For convenience, we write s_0 instead of s and t_n instead of t. In particular, we assume that the control state of t_1 is q, and the control state of s_{n-1} is p. Relying on Lemma 6, it is sufficient to consider configurations of sizes satisfying the following inequalities:

$$\mathrm{size}(s_i) \leq \mathrm{size}(t_{i+1}) \qquad \text{for } i \in \{1 \dots n-1\}. \tag{11}$$

In the second phase, the algorithm guesses, for $i \in \{0 \dots n-1\}$, a sequence of subtransitions from s_i to t_{i+1} of length bounded by polynomial derived from the proof of Lemma 8; and checks that the respective sequences of subtransitions lead from s_i to t_{i+1}, as required by Lemma 7. □

Acknowledgements. We are grateful to anonymous reviewers for careful reading and many valuable comments.

References

1. Atig, M.F.: From Multi to Single Stack Automata. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 117–131. Springer, Heidelberg (2010)
2. Atig, M.F., Bouajjani, A.: On the Reachability Problem for Dynamic Networks of Concurrent Pushdown Systems. In: Bournez, O., Potapov, I. (eds.) RP 2009. LNCS, vol. 5797, pp. 1–2. Springer, Heidelberg (2009)
3. Atig, M.F., Bouajjani, A., Qadeer, S.: Context-bounded analysis for concurrent programs with dynamic creation of threads. Logical Methods in Computer Science 7(4) (2011)
4. Bouajjani, A., Emmi, M.: Analysis of recursively parallel programs. In: POPL, pp. 203–214 (2012)
5. Bouajjani, A., Esparza, J., Maler, O.: Reachability Analysis of Pushdown Automata: Application to Model-Checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 135–150. Springer, Heidelberg (1997)
6. Bouajjani, A., Esparza, J., Schwoon, S., Strejcek, J.: Reachability analysis of multithreaded software with asynchronous communication. In: Software Verification: Infinite-State Model Checking and Static Program Analysis (2006)
7. Bouajjani, A., Müller-Olm, M., Touili, T.: Regular Symbolic Analysis of Dynamic Networks of Pushdown Systems. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 473–487. Springer, Heidelberg (2005)
8. Burkart, O., Caucal, D., Moller, F., Steffen, B.: Verification of infinite structures. In: Handbook of Process Algebra, pp. 545–623. Elsevier (2001)
9. Chadha, R., Madhusudan, P., Viswanathan, M.: Reachability under Contextual Locking. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 437–450. Springer, Heidelberg (2012)
10. Christensen, S.: Decidability and Decomposition in process algebras. PhD thesis, Dept. of Computer Science. University of Edinburgh, UK (1993)
11. Czerwiński, W., Fröschle, S., Lasota, S.: Partially-Commutative Context-Free Processes. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 259–273. Springer, Heidelberg (2009)
12. Czerwiński, W., Fröschle, S., Lasota, S.: Partially-commutative context-free processes: expressibility and tractability. Information and Computation 209, 782–798 (2011)
13. Czerwiński, W., Lasota, S.: Partially-commutative context-free languages (submitted, 2012)
14. Diekert, V., Rozenberg, G.: The book of traces. World Scientific (1995)
15. Esparza, J.: Petri nets, commutative context-free grammars, and basic parallel processes. Fundam. Inform. 31(1), 13–25 (1997)
16. Huynh, D.T.: Commutative grammars: The complexity of uniform word problems. Information and Control 57(1), 21–39 (1983)
17. Kahlon, V.: Reasoning about Threads with Bounded Lock Chains. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 450–465. Springer, Heidelberg (2011)
18. Kahlon, V., Ivančić, F., Gupta, A.: Reasoning About Threads Communicating via Locks. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 505–518. Springer, Heidelberg (2005)

19. Kretínský, M., Rehák, V., Strejcek, J.: Reachability is decidable for weakly extended process rewrite systems. Inf. Comput. 207(6), 671–680 (2009)
20. Lal, A., Reps, T.W.: Reducing concurrent analysis under a context bound to sequential analysis. Formal Methods in System Design 35(1), 73–97 (2009)
21. Lugiez, D., Schnoebelen, P.: The regular viewpoint on PA-processes. Theor. Comput. Sci. 274(1-2), 89–115 (2002)
22. Mayr, R.: Process rewrite systems. Inf. Comput. 156(1-2), 264–286 (2000)
23. Qadeer, S., Rehof, J.: Context-Bounded Model Checking of Concurrent Software. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 93–107. Springer, Heidelberg (2005)