Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Ordered multi-stack visibly pushdown automata ☆

Dario Carotenuto, Aniello Murano *, Adriano Peron

Università degli Studi di Napoli "Federico II", Italy

ARTICLE INFO

Article history:
Received 31 March 2014
Received in revised form 4 August 2016
Accepted 16 August 2016
Available online xxxx
Communicated by P. Aziz Abdulla

Keywords:
Visibly automata
Formal verification
Model checking
Pushdown automata
Automata-theoretic approach to system verification
Formal languages

ABSTRACT

Visibly Pushdown Automata (*VPA*) are a special case of pushdown machines where the stack operations are driven by the input. In this paper, we consider *VPA with multiple stacks*, namely *n-VPA*, with $n > 1$. These automata introduce a useful model to effectively describe concurrent pushdown systems using a simple communication mechanism between stacks. We show that *n*-VPA are strictly more expressive than VPA. Indeed, *n*-VPA accept some context-sensitive languages that are not context-free and some context-free languages that are not accepted by any VPA. Nevertheless, we show that the class of languages accepted by *n*-VPA is closed under union and intersection. On the contrary, this class turns out to be neither closed under complementation nor determinizable. Moreover, we show that the emptiness problem for *n*-VPA is undecidable. By adding an ordering constraint on stacks (*n*-OVPA), decidability of emptiness can be recovered, as well as the closure under complementation. Using these properties along with the automata-theoretic approach, one can prove that the model checking problem over *n*-OVPA models against *n*-OVPA specifications is decidable.

1. Introduction

In the area of formal design verification, one of the most significant developments is the discovery of the *model checking* technique, which automatically allows to verify on-going behaviors of reactive systems ([13,29,34]). In model checking (for a survey see [14]), one can check the correctness of a system with respect to a desired behavior by checking whether a mathematical model of the former satisfies a formal specification of the latter.

Traditionally, model checking is applied to finite-state systems, typically modeled by labeled-state transition graphs. Recently, model checking has been extended to infinite-state sequential systems (e.g., see [2,8–10,16,17,28,32,35]). These are systems in which each state carries a finite but unbounded amount of information, e.g., a pushdown store. *Pushdown automata* (PDA) naturally model the control flow of sequential programs with nested and recursive procedure calls, resulting in a very useful machinery to tackle with the formal verification of such systems. While many analysis problems, such as identifying dead code and accesses to uninitialized variables, can be captured as regular requirements, many others require inspection of the stack or matching of calls and returns, and these are non-regular context-free. More examples of non-regular properties are given in [30], where the specification of unbounded message buffers is considered.

Since checking context-free properties on PDA is proved in general to be undecidable [20], weaker models have been proposed in the last decade to decide different kinds of non-regular properties. One of the most promising approaches

☆ A preliminary version of this paper appears in the proceedings of the 11-th International Conference on Development in Language Theory, 2007 [12].

* Corresponding author.
E-mail addresses: aniello.murano@unina.it (A. Murano), adrperon@unina.it (A. Peron).

http://dx.doi.org/10.1016/j.tcs.2016.08.012
0304-3975/

Languages	Closure properties			Decision problems
	∪	∩	Complement	Emptiness
Regular	Yes	Yes	Yes	NLOGSPACE-C
CFL	Yes	No	No	PTIME-C
VPL	Yes	Yes	Yes	PTIME-C
n-OPL	Yes	No	No	2-EXPTIME-C
n-VPL	**Yes**	**Yes**	**No**	**UNDECIDABLE**
n-OVPL	**Yes**	**Yes**	**Yes**	**2-EXPTIME**
(k, n)-MVPLs	Yes	Yes	Yes	2-EXPTIME-C

Fig. 1. A comparison between closure properties and decision problems.

is that of *Visibly Pushdown Automata* (VPA) [1]. These are PDA where the push or pop actions on the stack are controlled externally by the input alphabet. Such a restriction on the use of the stack allows to enjoy all desirable closure properties and tractable decision problems, though retaining an expressiveness adequate to formulate program analysis questions (as summarized in Fig. 1). Therefore, checking pushdown properties of pushdown models is feasible as long as the calls and returns are made visible. This visibility condition seems quite natural to write requirements about pre/post conditions as well as for inter-procedural flow properties. In particular, requirements that can be verified in this manner include all regular properties, and non-regular properties such as: partial correctness (if P holds when a procedure is invoked, then, if the procedure returns, P' holds upon return), total correctness (if P holds when a procedure is invoked, then the procedure must return and P' must hold at the return state), local properties (the computation within a procedure by skipping over calls to other procedures satisfies a regular property, for instance, every request is followed by a response), access control (a procedure A can be invoked only if another procedure B is in the current stack), and stack limits (whenever the stack size is bounded by a given constant, a property A holds). Unfortunately, some natural context-free properties like "the number of calls to procedures A and B is the same" cannot be captured by any VPA [1]. Moreover, VPA cannot explicitly represent concurrency: for instance, properties of two threads running in parallel, each one exploiting its own pushdown store.

In this paper, we extend VPA in order to enrich their expressiveness though maintaining some desirable closure properties and decidability results. We first consider VPA with $n > 1$ input-driven pushdown stores and call the proposed model *n-Visibly Pushdown Automata* (n-VPA). As in the VPA case, n-VPA input symbols are partitioned in subclasses, each one triggering a transition belonging to a specific class, i.e., push/pop/local transition. The input symbols also determine the operating stack (one or more stacks working simultaneously). Moreover, visibility in n-VPA affects the transfer of information from one stack to other stacks (a pop from a stack can be paired with a simultaneous pop on a number of stacks). In this way, n-VPA turn out to be strictly more expressive than VPA. In particular, they are able to accept context-sensitive languages that are not context-free. Unfortunately, this extension does not preserve complementation (as it has been proved in [7]) and does not preserve determinizability. Moreover, the proposed extension does not preserve decidability of the emptiness problem already in the case of 2-VPA, as we prove by a reduction from the halting problem for Minsky Machines.

In the automata-theoretic approach to formal verification, to gain with a decidable model checking procedure, decidability of the emptiness problem is crucial. For this reason, we add to n-VPA a suitable restriction on stack operations, namely we consider n-VPA in which stacks are ordered and pop operations on the i-th stack are allowed only if all the previous $j < i$ stacks are empty. We call such a variant *ordered* n-VPA (n-OVPA). The ordering constraint is inspired from the class of *multi-pushdown automata* (MPDA), as they were defined in [11]. In [11], it has been shown that the class of languages accepted by MPDA is strictly included into context-sensitive languages, it is closed under union, but not under intersection and complementation. Moreover, the emptiness problem for MPDA is decidable and 2-EXPTIME-complete, as reported in [5]. Actually, this result corrects an error in the decidability proof given in [11]. As reported in [5], the error has been also pointed out and corrected independently by the authors of [11] (see also [4]).

From an expressive point of view, n-OVPA are a proper subclass of MPDA with n stacks (*n-OPA*). Differently from n-OPA, exploiting visibility allows to recover in n-OVPA closure under intersection and complementation. This is a crucial point as it allows to face the model checking problem following an automata-theoretic approach. Using such a methodology, to verify whether a system, modeled as an n-OVPA S, satisfies a correctness requirement, expressed by an n-OVPA P, we check for emptiness the intersection between the language accepted by S and the complement of the language accepted by P (i.e., $\mathcal{L}(S) \cap \overline{\mathcal{L}(P)} = \emptyset$). Since we prove for n-OVPA that intersection, emptiness, and complementation are all decidable, we get that model checking an n-OVPA model against an n-OVPA specification is decidable as well. This is notable since checking context-free properties over PDA is proved to be undecidable [20], as it is also the case for model checking multi-pushdown properties over MPDA.

The extension we propose for VPA does not only affect expressiveness, but also gives us a way to naturally describe distributed pushdown systems executions. In fact, we show that n-OVPA capture the behavior of systems built on pairs of VPA running in a suitable synchronous way according to a distributed computing paradigm. To this purpose, we introduce a composition operator on VPA parameterized on a communication interface. Given a tuple of VPA, this operator allows to build a *Synchronized System* of VPA (S-VPA), which behaves synchronously and in parallel. A communication between two synchronous VPA consists in a transfer of information from the top of the stack of one VPA to the top of stack of the other. If we interpret each one of the involved VPA as a process with its own pushdown store (containing activation records of procedure calls, for instance), the enforced communication form can be seen as a *Remote Procedure Call* [31], widely exploited

Please cite this article in press as: D. Carotenuto et al., Ordered multi-stack visibly pushdown automata, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2016.08.012