Reachability Analysis of Pushdown Automata: Application to Model-Checking

Ahmed Bouajjani[1] Javier Esparza[2*] Oded Maler[1]

[1] VERIMAG, Centre Equation, 2 av. de Vignate, 38610 Gières, France.
email: Ahmed.Bouajjani@imag.fr, Oded.Maler@imag.fr
[2] Inst. für Informatik, Tech. Univ. München, Arcisstr. 21, 81539 München, Germany.
email: esparza@informatik.tu-muenchen.de

Abstract. We apply the *symbolic* analysis principle to pushdown systems. We represent (possibly infinite) sets of configurations of such systems by means of finite-state automata. In order to reason in a uniform way about analysis problems involving both existential and universal path quantification (such as model-checking for branching-time logics), we consider the more general class of *alternating* pushdown systems and use *alternating* finite-state automata as a representation structure for sets of their configurations. We give a simple and natural procedure to compute sets of predecessors using this representation structure. We incorporate this procedure into the automata-theoretic approach to model-checking to define new model-checking algorithms for pushdown systems against both linear and branching-time properties. From these results we derive upper bounds for several model-checking problems as well as matching lower bounds.

1 Introduction

Systems are commonly modeled by various types of transition systems, including finite automata, pushdown automata, Petri nets, timed or hybrid automata, etc. In this framework, most of the system analysis problems (model-checking, synthesis) reduce to various kinds of "reachability problems" on these models. It is therefore fundamental for system analysis to develop algorithms that compute the set of *all predecessors* of a given set of states S, i.e., the set of states from which it is possible to reach S.

Let $pre(S)$ denote the set of immediate predecessors (via a single transition) of the set S, and let $pre^*(S)$ denote the set of all its predecessors. Clearly, $pre^*(S)$ is the limit of the *infinite* increasing sequence $\{X_i\}_{i\geq 0}$ given by $X_0 = S$ and $X_{i+1} = X_i \cup pre(X_i)$ for every $i \geq 0$.

In the case of finite-state systems, the sets X_i are all finite, and the sequence $\{X_i\}_{i\geq 0}$ is guaranteed to reach a fixpoint, which immediately provides an algorithm to compute $pre^*(S)$. Unfortunately, these properties no longer hold for any non-trivial class of infinite-state systems. For such systems, the first task is then to find a class of *finite* structures that can represent the infinite sets of states we are

* Supported by the University Joseph Fourier (Grenoble I) and by Teilprojekt A3 of the Sonderforschungsbereich 342.

interested in. Since boolean combinations of sets of states are usually interesting, the class should be closed under boolean operations. Moreover, since we wish to check if a given state (for instance the initial state) belongs to an infinite set, the membership problem of the class should be decidable. Once such a class has been found, it remains to show that it is (effectively) closed under the pre^* function.

Several instances of systems and their corresponding representation structures have been considered in the literature. For example, in the case of timed automata, special kinds of polyhedra (regions) are used to represent infinite sets of states (vectors of reals corresponding to clock valuations) [3]. Polyhedra are also used for linear hybrid systems. However, in this case, there is no algorithm for computing a finite representation of the *exact* set of predecessors (the reachability problem is undecidable), but upper approximations of this set can be calculated [2]. In [5], representation structures called QDD's are introduced for FIFO-channel systems. These structures are finite-state automata representing sets of queue contents. As in the case of linear hybrid systems, the procedure for calculating the set of predecessors for these structures is not guaranteed to terminate. Finally, notice that symbolic representations (e.g. BDD's [10]) are also used in the finite-state case in order to overcome the state-explosion problem [17].

In this paper we consider pushdown systems,as well as the more general class of *alternating pushdown systems*, i.e., pushdown systems with both existential and universal nondeterminism (see [20] for a survey on alternating automata). This general setting allows to reason in a uniform way about *analysis* problems where existential and universal path quantification must be considered, like model-checking for branching-time temporal logics (see Section 5) and also about *synthesis* problems, such as finding winning strategies for 2-player games (see [4]).

A state (we use rather the word "configuration") of a pushdown system is a pair $\langle p, w \rangle$ where p is control location and w is a sequence of stack symbols (the stack contents). As a representation structure for sets of configurations, we propose the *alternating multi-automaton* (AMA), an alternating finite-state automaton with one initial state for each control location. The automaton recognizes the configuration $\langle p, w \rangle$ if it accepts the word w from the initial state corresponding to p. It is important to remember that an AMA is just a tool to represent a set of configurations, and not to confuse its "behaviour" with that of the pushdown system.

It is easy to show that AMA's are closed under boolean operations, and that its membership problem is decidable. Our main result is a simple and natural algorithm for computing the pre^* function. As an application, we construct elegant model-checking algorithms for pushdown systems w.r.t. both linear and branching-time temporal logics. More precisely, we show how to construct AMA's accepting the set of *all* configurations satisfying ω-regular properties of linear-time temporal logics (including all properties expressible in LTL [18] or the linear-time μ-calculus [19]), or properties expressed as formulas of the alternation-free modal μ-calculus. A first version of these results appeared in [8] (where the logic CTL [13] is considered instead of the more expressive alternation-free modal μ-calculus).

Moreover, our approach allows us to obtain a number of complexity results: we show that the model-checking problems mentioned above are in DEXPTIME, and that the model-checking problem for pushdown systems and a subset of CTL can be solved in PSPACE. Using a technique due to Walukiewicz [22], we complement these results with matching lower bounds, i.e., we show that all these problems are complete for their corresponding complexity classes.

The paper is structured as follows. In Section 2, we give an algorithm which computes the pre^* function for pushdown systems. In this case, the representation structure is a simple nondeterministic multi-automaton (i.e., without alternation). We apply this algorithm in Section 3 to the model-checking problem for linear-time logics. In Section 4, we generalize the algorithm given in Section 2 to alternating pushdown systems. In Section 5, we apply the new algorithm to the model-checking problem for branching-time logics. Proofs of the theorems can be found in the full paper [7].

2 Reachability in pushdown systems

2.1 Pushdown Systems

A pushdown system (PDS for short) is a triplet $\mathcal{P} = (P, \Gamma, \Delta)$ where P is a finite set of *control locations*, Γ is a finite *stack alphabet*, and $\Delta \subseteq (P \times \Gamma) \times (P \times \Gamma^*)$ is a finite set of *transition rules*. If $((q,\gamma),(q',w)) \in \Delta$ then we write $(q,\gamma) \hookrightarrow (q',w)$ (we reserve $\rightarrow$ to denote the transition relations of finite automata).

Notice that PDS's have no input alphabet. We do not use them as language acceptors but are rather interested in the behaviours they generate.

A *configuration* of $\mathcal{P}$ is a pair $\langle p, w\rangle$ where $p \in P$ is a control location and $w \in \Gamma^*$ is a *stack content*.

If $(q,\gamma) \hookrightarrow (q',w)$, then for every $w' \in \Gamma^*$ the configuration $\langle q, \gamma w'\rangle$ is an *immediate predecessor* of $\langle q', ww'\rangle$, and $\langle q', ww'\rangle$ is an *immediate successor* of $\langle q, \gamma w'\rangle$. The *reachability relation* $\Rightarrow$ is the reflexive and transitive closure of the immediate successor relation. A *run* of $\mathcal{P}$ is a maximal sequence of configurations such that for each two consecutive configurations c_i and c_{i+1}, c_{i+1} is an immediate successor of c_i. The set of all runs of $\mathcal{P}$ is denoted by $Runs_{\mathcal{P}}$.

The predecessor function $pre_{\mathcal{P}} : 2^{P\times\Gamma^*} \rightarrow 2^{P\times\Gamma^*}$ is defined as follows: c belongs to $pre_{\mathcal{P}}(C)$ if some immediate successor of c belongs to C. The reflexive and transitive closure of $pre_{\mathcal{P}}$ is denoted by $pre^*_{\mathcal{P}}$. Clearly, $pre^*_{\mathcal{P}}(C) = \{c \in P \times \Gamma^* \mid \exists c' \in C.\ c \Rightarrow c'\}$. We denote by $pre^+_{\mathcal{P}}$ the function $pre_{\mathcal{P}} \circ pre^*_{\mathcal{P}}$. We will omit the subscript $\mathcal{P}$ and write simply pre, pre^*, and pre^+ when it is clear from the context which system is under consideration.

2.2 Multi-automata

Let $\mathcal{P} = (P, \Gamma, \Delta)$ be a pushdown system where $P = \{p^1, \ldots p^m\}$. A *$\mathcal{P}$-multi-automaton* ($\mathcal{P}$-MA for short, or just MA when $\mathcal{P}$ is clear from the context) is a tuple $\mathcal{A} = (\Gamma, Q, \delta, I, F)$ where Q is a finite set of *states*, $\delta \subseteq Q \times \Gamma \times Q$ is a set of *transitions*, $I = \{s^1, \ldots s^m\} \subseteq Q$ is a set of *initial states* and $F \subseteq Q$ is a set of *final states*.

We define the *transition relation* $\longrightarrow \subseteq Q \times \Gamma^* \times Q$ as the smallest relation satisfying:

– if $(q,\gamma,q') \in \delta$ then $q \xrightarrow{\gamma} q'$,

- $q \xrightarrow{\varepsilon} q$ for every $q \in Q$, and
- if $q \xrightarrow{w} q''$ and $q'' \xrightarrow{\gamma} q'$ then $q \xrightarrow{w\gamma} q'$.

$\mathcal{A}$ *accepts* or *recognizes* a configuration $\langle p^i, w\rangle$ if $s^i \xrightarrow{w} q$ for some $q \in F$. The set of configurations recognized by $\mathcal{A}$ is denoted by $Conf(\mathcal{A})$. A set of configurations is *regular* if it is recognized by some MA.

A *w-run* of $\mathcal{A}$, where $w = \gamma_1 \dots \gamma_n \in \Gamma^*$, is a sequence $s^i \xrightarrow{\gamma_1} q_1 \dots \xrightarrow{\gamma_n} q_n$.

2.3 Calculating pre^*

Fix a pushdown system $\mathcal{P} = (P, \Gamma, \Delta)$ where $P = \{p^1, \dots, p^m\}$. We show in this section that given a regular set of configurations C of $\mathcal{P}$ recognized by a MA $\mathcal{A}$, we can construct another MA $\mathcal{A}_{pre^*}$ recognizing $pre^*(C)$.

By definition, $pre^*(C) = \bigcup_{i \geq 0} X_i$ with $X_0 = C$ and $X_{i+1} = X_i \cup pre(X_i)$ for every $i \geq 0$. Therefore, one may try to calculate $pre^*(C)$ by iteratively constructing the increasing sequence $X_0, X_1, \dots$. If $X_{i+1} = X_i$ holds for some $i \geq 0$, then it is clear that $X_i = pre^*(C)$.

However, the existence of such a fixed point is not guaranteed in general, and we may never reach the limit of the X_i sequence. Consider for instance the PDS with one state p, one stack symbol γ, and one transition rule $(p, \gamma) \hookrightarrow (p, \varepsilon)$, and take $C = \{\langle p, \varepsilon\rangle\}$. Clearly, we have $X_i = \{\langle p, \varepsilon\rangle, \langle p, \gamma\rangle, \dots, \langle p, \gamma^i\rangle\}$ and so $X_{i+1} \neq X_i$ for every $i \geq 0$.

To overcome this problem, we calculate $pre^*(C)$ differently, as the limit of another increasing sequence of sets of configurations $Y_0, Y_1, \dots$ for which we can prove the following properties:

P1. $\exists i \geq 0.\ Y_{i+1} = Y_i$,
P2. $\forall i \geq 0.\ X_i \subseteq Y_i$,
P3. $\forall i \geq 0.\ Y_i \subseteq \bigcup_{j \geq 0} X_j \quad = pre^*(C)$.

Property (P1) ensures termination of the procedure that computes the sequence of Y_i's. Property (P2) ensures that, by calculating the limit of the Y_i's, we capture (at least) the whole set $pre^*(C)$, and property (P3) ensures that only elements of $pre^*(C)$ are captured.

The Y_i's are formally defined as the sets of configurations recognized by a sequence $\mathcal{A}_0, \mathcal{A}_1 \dots$ of MA's satisfying for every $i \geq 0$ the following property: $\mathcal{A}_{i+1}$ has *the same states* as $\mathcal{A}_i$, and possibly a superset of its transitions. Since a MA with n states and m input symbols can have at most $n^2 \cdot m$ transitions, the Y_i's must converge to a fixpoint.[3]

We start with a MA $\mathcal{A}$ recognizing the regular set of configurations C. We assume without loss of generality that $\mathcal{A}$ has no transition leading to an initial state (every MA can be converted to one having this property). We take $\mathcal{A}_0 = \mathcal{A}$. We denote by $\rightarrow_i$ the transition relation of $\mathcal{A}_i$. For every $i \geq 0$, $\mathcal{A}_{i+1}$ is obtained from $\mathcal{A}_i$ by conserving the same states and transitions, and adding for every transition rule

[3] The idea is inspired by the construction given in [6], pages 91-93, of a finite-state automaton recognizing the closure of a regular language under the rewriting relation induced by a *monadic string-rewriting system.*

$(p^j, \gamma) \hookrightarrow (p^k, w)$ and every state q such that $s^k \xrightarrow{w}_i q$ a new transition $s^j \xrightarrow{\gamma}_{i+1} q$. Then, for every $i \geq 0$ we define $Y_i = Conf(\mathcal{A}_i)$. Note that the new transitions added to $\mathcal{A}_i$ in order to construct $\mathcal{A}_{i+1}$ start at initial states.

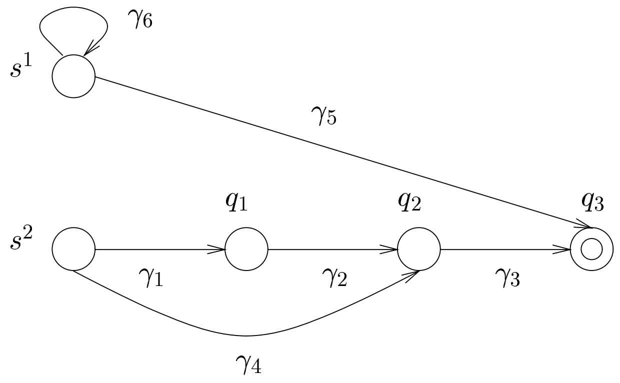
To understand the idea behind this construction, observe that $\langle p^k, \gamma w' \rangle$ is an immediate predecessor of $\langle p^j, ww' \rangle$ by the rule $(p^j, \gamma) \hookrightarrow (p^k, w)$. So, if the word ww' is accepted starting from s^k by $\mathcal{A}_i$ ($s^k \xrightarrow{w}_i q \xrightarrow{w'}_i q' \in F$), then the new transition in $\mathcal{A}_{i+1}$ allows to accept $\gamma w'$ starting from s^j ($s^j \xrightarrow{\gamma}_{i+1} q \xrightarrow{w'}_i q' \in F$). Let us illustrate the construction by means of an example.

Let $\mathcal{P}$ be the PDS such that $P = \{p^1, p^2\}$, $\Gamma = \{\gamma_1, \ldots, \gamma_6\}$, and Δ contains the rules

$$(p^2, \gamma_4) \hookrightarrow (p^2, \gamma_1\gamma_2) \quad (p^1, \gamma_5) \hookrightarrow (p^2, \gamma_4\gamma_3) \quad (p^1, \gamma_6) \hookrightarrow (p^1, \varepsilon)$$

Consider the set of configurations $C = \{\langle p^2, \gamma_1\gamma_2\gamma_3 \rangle\}$. It can be represented by a MA $\mathcal{A}$ such that $Q = \{s^1, s^2, q_1, q_2, q_3\}$, $I = \{s^1, s^2\}$, $F = \{q_3\}$, and δ contains the transitions $s^2 \xrightarrow{\gamma_1} q_1$, $q_1 \xrightarrow{\gamma_2} q_2$, and $q_2 \xrightarrow{\gamma_3} q_3$.

The picture below shows the automaton $\mathcal{A}_{pre^*}$ obtained at the end of the construction.



In the first step (from $\mathcal{A}_0$ to $\mathcal{A}_1$) we have $s^2 \xrightarrow{\gamma_1\gamma_2}_0 q_2$ and $s^1 \xrightarrow{\epsilon}_0 s^1$, and so we add the transitions $s^2 \xrightarrow{\gamma_4}_1 q_2$ and $s^1 \xrightarrow{\gamma_6}_1 s^1$ corresponding respectively to the first and to the third transition rules of $\mathcal{P}$. No other transitions are added. The new automaton now accepts all immediate predecessors of $\langle p^2, \gamma_1\gamma_2\gamma_3 \rangle$, namely the configuration $\langle p^2, \gamma_4\gamma_3 \rangle$ (note that the set of words accepted from s^1 is empty at this step).

In the second step, we add the transition $s^1 \xrightarrow{\gamma_5}_2 q_3$, corresponding to the second transition rule of $\mathcal{P}$. At this point the construction stops since no further transition must be added. So, we have $\mathcal{A}_{pre^*} = \mathcal{A}_2$, and

$$pre^*(C) = (\{p^1\} \times \gamma_6^*\gamma_5) \cup (\{p^2\} \times \{\gamma_1\gamma_2\gamma_3, \gamma_4\gamma_3\})$$

Observe that in this example we have $X_1 = Y_1$ but $X_2 \subset Y_2$. Indeed, in the second step of the construction, after adding $s^1 \xrightarrow{\gamma_5}_2 q_3$, $\mathcal{A}_2$ accepts all the configurations of the form $\langle p^1, \gamma_6^k\gamma_5 \rangle$ for every $k \geq 0$, whereas only $\langle p^1, \gamma_5 \rangle$ belongs to X_2. However, despite the fact that these configurations are not immediate predecessors of X_1 configurations, they are all in $pre^*(C)$ because $\langle p^1, \gamma_6^k \gamma_5 \rangle \in X_{k+2}$ for every $k \geq 0$.

The proofs of the properties (P1), (P2), and (P3) are given in the full paper. We deduce from these properties the following theorem.

Theorem 2.1 *Given a PDS $\mathcal{P}$ and a regular set of configurations recognized by a $\mathcal{P}$-MA $\mathcal{A}$, we can construct a $\mathcal{P}$-MA $\mathcal{A}_{pre^*}$ recognizing $pre^*(Conf(\mathcal{A}))$.*

We conclude the section with a remark on complexity. In order to construct $\mathcal{A}_{i+1}$ from $\mathcal{A}_i$, we compute for each transition rule $(p, \gamma) \hookrightarrow (p', w)$ of the PDS $\mathcal{P}$ the set of states q such that $s' \xrightarrow{w}_i q$, and then add the transition $s \xrightarrow{\gamma}_{i+1} q$ to $\mathcal{A}_{i+1}$. The computation time of the set is quadratic in the number of states of $\mathcal{A}_i$ (which is equal to the number of states of $\mathcal{A}$) and linear in the length of w ([1], Theorem 9.5). Thus, the construction of $\mathcal{A}_{i+1}$ from $\mathcal{A}_i$ takes time $O(|\mathcal{A}|^2 \cdot |\mathcal{P}|^2)$.

Now, the sequence $\mathcal{A}_0, \mathcal{A}_1, \ldots$ must reach the fixpoint $\mathcal{A}_{pre^*}$ after at most $O(|\mathcal{A}|^2 \cdot |\mathcal{P}|)$ steps, because this is an upper bound on the number of transitions of any $\mathcal{P}$-MA having the same states as $\mathcal{A}$. So the computation of $\mathcal{A}_{pre^*}$ takes $O(|\mathcal{A}|^4 \cdot |\mathcal{P}|^3)$ time.

3 Model-Checking Linear-Time Temporal Logics

Let $Prop$ be a finite set of atomic propositions, and let $\Sigma = 2^{Prop}$. It is well known that the semantics of properties expressed in linear time temporal logics like LTL or the linear-time μ-calculus are ω-regular sets over the alphabet Σ. Moreover, there exist algorithms which construct Büchi automata to recognize these sets [21, 20]. This is all we need to know about these logics in this paper in order to give model checking algorithms for PDS's.

Let $\mathcal{P} = (P, \Gamma, \Delta)$ be a PDS, and let $\Lambda : P \rightarrow \Sigma$ be a labelling function, which associates a set of true propositions with every control location p. Given a formula φ of such an ω-regular logic we wish to solve the following problem:

> Compute the set of *all* configurations c of $\mathcal{P}$ such that every run starting from c satisfies φ (via the labelling function Λ).

Then, the model checking problem consists in checking whether a given initial configuration belongs to this set of configurations.

We start by constructing a Büchi automaton $\mathcal{B}$ corresponding to the negation of φ. The product of the PDS $\mathcal{P}$ and this Büchi automaton yields a Büchi PDS $\mathcal{BP}$ with a set of *repeating control locations* $G \subseteq P$. Then, the original problem reduces straightforwardly to the following *accepting run problem*:

> Compute the set $\mathcal{C}$ of configurations c of $\mathcal{BP}$ such that $\mathcal{BP}$ has an accepting run starting from c (i.e., a run which visits infinitely often configurations with control locations in G).

(Notice that the emptiness problem of Büchi PDS's - whether the initial configuration has an accepting run - reduces to the accepting run problem via the membership problem of MA).

The following proposition shows that the accepting run problem of Büchi PDS's can be reduced to a reachability problem:

Proposition 3.1 *Let c be a configuration of a Büchi PDS $\mathcal{B}P$. $\mathcal{B}P$ has an accepting run starting from c if and only if there exist configurations $\langle p, \gamma \rangle$, $\langle g, u \rangle$, and $\langle p, \gamma v \rangle$, not all three equal, such that $g \in G$ and:*

(1) $c \Rightarrow \langle p, \gamma w \rangle$ for some $w \in \Gamma^*$, and
(2) $\langle p, \gamma \rangle \Rightarrow \langle g, u \rangle \Rightarrow \langle p, \gamma v \rangle$.

We can reformulate conditions (1) and (2) of Proposition 3.1 as follows:

(1′) $c \in pre^*(\{p\} \times \gamma\Gamma^*)$, and
(2′) $\langle p, \gamma \rangle \in pre^+((G \times \Gamma^*) \cap pre^*(\{p\} \times \gamma\Gamma^*))$.

Since $G \times \Gamma^*$ and $\{p\} \times \gamma\Gamma^*$ are regular sets, we can use Theorem 2.1 to construct MA's recognizing the sets $pre^*(\{p\} \times \gamma\Gamma^*)$ and $pre^+((G \times \Gamma^*) \cap pre^*(\{p\} \times \gamma\Gamma^*))$ (for pre^+ we need to define for a MA $\mathcal{A}$ another MA recognizing $pre(Conf(\mathcal{A}))$, which is a simple exercise). Therefore, by Proposition 3.1, we can construct a MA which recognizes the set of all configurations having an accepting run: First, we determine all the configurations $\langle p, \gamma \rangle$ (there are finitely many of them) for which (2′) holds, and then we construct a MA recognizing the union of the sets $pre^*(\{p\} \times \gamma\Gamma^*)$ for all such pairs.

The sizes of the MA's for the sets $G \times \Gamma^*$ and $\{p\} \times \gamma\Gamma^*$ are polynomial in the size of the Büchi PDS. Hence, since the computation of $pre^*_{\mathcal{P}}(Conf(\mathcal{A}))$ for a MA $\mathcal{A}$ takes polynomial time in the size of $\mathcal{P}$ and the number of states of $\mathcal{A}$, we deduce the following result:

Theorem 3.1 *The accepting run problem of Büchi PDS's can be solved in polynomial time.*

Since the membership problem of MA's can be solved in linear time, a consequence of Theorem 3.1 is that the emptiness problem of Büchi PDS's can also be solved in polynomial time.

Theorem 3.2 *The model checking problems for LTL and the linear-time μ-calculus and PDS's are DEXPTIME-complete. The model checking problem for a fixed formula is polynomial in the size of the PDS.*

Proof. Let us first prove membership in DEXPTIME. Let $\mathcal{P}$ be a PDS of size $n_{\mathcal{P}}$ and φ a formula of length n_{φ}. It is well known that it is possible to construct a Büchi automaton $\mathcal{B}$ for the negation of φ having exponential size in n_{φ}, and this construction can be done in exponential time [21, 19]. Hence, the product of $\mathcal{P}$ and $\mathcal{B}$ has polynomial size in $n_{\mathcal{P}}$ and exponential size on n_{φ}. Applying Theorem 3.1 we obtain an exponential time bound. If the formula φ is fixed, then we have an algorithm polynomial in $n_{\mathcal{P}}$.

To prove hardness, we use a reduction from the problem of deciding whether a given linearly bounded alternating Turing machine accepts a given input or not. The details of the reduction are given in the full paper. □

The model-checking problem for LTL or the linear-time μ-calculus and finite-state systems is known to be PSPACE-complete, but polynomial in the size of the system. Since the properties of systems one wishes to check can be usually encoded into short formulas, model-checkers based on linear-time logics, like SPIN [16], have proved to be useful in practice. Theorem 3.2 shows that the complexity of model-checking for PDS's is worse than the complexity for finite-state systems, but not much worse: it remains polynomial in the size of the system.

4 Reachability in Alternating Pushdown Systems

4.1 Alternating Pushdown Systems

We consider now the problem of computing the set of predecessors of a regular set of configurations of an *alternating* pushdown system. We show that this set is also regular, and we give a procedure for constructing its representation by means of *alternating* finite-state multi-automata. To this end, we generalize the technique described in the Section 2. The construction we give is used in the model checking algorithms for branching-time logics given in the next section.

An *alternating pushdown system* (APDS for short) is a triplet $\mathcal{P} = (P, \Gamma, \Delta)$, where P and Γ are as for PDSs, and Δ is a function that assigns to each element of $P \times \Gamma$ a negation-free boolean formula over elements of $P \times \Gamma^*$. We assume that boolean formulae are always in disjunctive normal form, which allows us to equivalently define Δ as a subset of the set $(P \times \Gamma) \times 2^{P \times \Gamma^*}$ of *transition rules*: for example, instead of writing

$$\Delta(p, \gamma) = ((p_1, w_1) \vee (p_2, w_2)) \wedge (p_3, w_3)$$

we write

$$\{\ ((p, \gamma), \{(p_1, w_1), (p_3, w_3)\})\ ,\ \ ((p, \gamma), \{(p_2, w_2), (p_3, w_3)\})\ \} \subseteq \Delta$$

or just

$$(p, \gamma) \hookrightarrow \{(p_1, w_1), (p_3, w_3)\}\ ,\ \ (p, \gamma) \hookrightarrow \{(p_2, w_2), (p_3, w_3)\}$$

If $(p, \gamma) \hookrightarrow \{(p_1, w_1), \ldots, (p_n, w_n)\}$, then for every $w \in \Gamma^*$ the configuration $\langle p, \gamma w\rangle$ is an *immediate predecessor* of the set $\{\langle p_1, w_1 w\rangle, \ldots, \langle p_n, w_n w\rangle\}$, and this set is an *immediate successor* of $\langle p, \gamma w\rangle$. Intuitively, at the configuration $\langle p, \gamma w\rangle$ the APDS selects nondeterministically a transition rule of the form

$$(p, \gamma) \hookrightarrow \{(p_1, w_1), \ldots, (p_n, w_n)\}$$

and forks into n copies in the configurations $\langle p_1, w_1 w\rangle, \ldots, \langle p_n, w_n w\rangle$.

A *run* of $\mathcal{P}$ for an initial configuration c is a tree of configurations with root c, such that the children of a node c' are the configurations that belong to one of its immediate successors (nodes of the form $\langle p, \varepsilon\rangle$ have no successors).

We define the *reachability relation* $\Rightarrow \subseteq (P \times \Gamma^*) \times 2^{P \times \Gamma^*}$ between configurations and sets of configurations. Informally, $c \Rightarrow C$ if and only if C is a finite frontier (finite maximal set of incomparable nodes) of a run of $\mathcal{P}$ starting from c. Formally, $\Rightarrow$ is the smallest subset of $(P \times \Gamma^*) \times 2^{P \times \Gamma^*}$ such that:

1. $c \Rightarrow \{c\}$ for every $c \in P \times \Gamma^*$,
2. if c is an immediate predecessor of C, then $c \Rightarrow C$,

3. if $c \Rightarrow \{c_1, \ldots, c_n\}$ and $c_i \Rightarrow C_i$ for each $1 \leq i \leq n$, then $c \Rightarrow (C_1 \cup \ldots \cup C_n)$.

The function $pre_{\mathcal{P}}: 2^{P \times \Gamma^*} \rightarrow 2^{P \times \Gamma^*}$ is now defined as follows: c belongs to $pre_{\mathcal{P}}(C)$ if some immediate successor of c is contained in C (observe that the immediate successor of c is now a set). We denote by $pre^*_{\mathcal{P}}$ the transitive closure of $\lambda C.\ (C \cup pre_{\mathcal{P}}(C))$, i.e., given a set of configurations C, $pre^*_{\mathcal{P}}(C) = \bigcup_{i \geq 0} X_i$, where $X_0 = C$ and $X_{i+1} = X_i \cup pre_{\mathcal{P}}(X_i)$, for every $i \geq 0$. As in the case of PDS's, $pre^*_{\mathcal{P}}(C) = \{c \in P \times \Gamma^* \mid \exists C' \subseteq C.\ c \Rightarrow C'\}$.

4.2 Alternating multi-automata

Fix an APDS $\mathcal{P} = (P, \Gamma, \Delta)$. An *alternating $\mathcal{P}$-multi-automaton* ($\mathcal{P}$-AMA for short, or just AMA when $\mathcal{P}$ is clear from the context) is a tuple $\mathcal{A} = (\Gamma, Q, \delta, I, F)$ which differs from an MA only in the nature of δ. δ is now a function that assigns to every pair of $Q \times \Gamma$ a positive boolean formula with Q as set of variables. As in the case of APDSs, we can equivalently represent δ as a set of transitions, which are elements of $(Q \times \Gamma) \times 2^Q$.

The *transition relation* $\rightarrow \subseteq Q \times \Gamma^* \times 2^Q$ is the smallest relation satisfying

– if $(q, \gamma, Q') \in \delta$ then $q \xrightarrow{\gamma} Q'$,
– $q \xrightarrow{\varepsilon} \{q\}$ for every $q \in Q$,
– if $q \xrightarrow{w} \{q_1, \ldots, q_n\}$ and $q_i \xrightarrow{\gamma} Q_i$ for each $1 \leq i \leq n$, then $q \xrightarrow{w\gamma} (Q_1 \cup \ldots \cup Q_n)$.

A configuration $\langle p^i, w \rangle$ is *recognized* by $\mathcal{A}$ if $s^i \xrightarrow{w} Q'$ for some $Q' \subseteq F$. Given a finite sequence $w \in \Gamma^*$ and a state $q \in Q$, a *run* of $\mathcal{A}$ over w starting from q is a finite tree whose nodes are labelled by states in Q and whose edges are labelled by symbols in Γ, such that the root is labelled by q, and the labelling of the other nodes is consistent with δ. Notice that in such a tree each sequence of edges going from the root to the leaves is labelled by w, and hence, all the edges starting at the same level of the tree have the same label, and all the leaves of the tree are at the same height.

It is immediate to show that AMA's are closed under boolean operations. We mention also that the membership problem for AMA's can be solved in polynomial time.

4.3 Calculating pre^*

Let $\mathcal{P} = (P, \Gamma, \Delta)$ be an alternating pushdown system. We show in this section that given a regular set of configurations C of $\mathcal{P}$, recognized by an *alternating*-multi-automaton $\mathcal{A}$, we can construct another AMA $\mathcal{A}_{pre^*}$ such that $Conf(\mathcal{A}_{pre^*}) = pre^*(C)$.

The construction is very similar to that of the non-alternating case. We assume without loss of generality that no transition of $\mathcal{A}$ leads to a set of states containing an initial state. We define a sequence of AMA's $\mathcal{A}_0, \mathcal{A}_1, \ldots$ such that $\mathcal{A}_0 = \mathcal{A}$. For every $i \geq 0$, $\mathcal{A}_{i+1}$ is obtained from $\mathcal{A}_i$ by conserving the same states and transitions, and adding for every transition rule

$$\langle p^j, \gamma \rangle \hookrightarrow \{\langle p^{k_1}, w_1 \rangle, \ldots, \langle p^{k_m}, w_m \rangle\}$$

and every set

$$s^{k_1} \xrightarrow{w_1}_i P_1 \ , \ \ldots \ , \ s^{k_m} \xrightarrow{w_m}_i P_m$$

a new transition

$$s^j \xrightarrow{\gamma}_{i+1} (P_1 \cup \ldots \cup P_m)$$

Then, define $Y_i = Conf(\mathcal{A}_i)$ for every $i \geq 0$.

The intuitive justification of the construction is that we add the configuration $\langle p^j, \gamma w\rangle$ to the set of predecessors of C whenever all the configurations $\langle p^{k_1}, w_1 w\rangle$, ..., $\langle p^{k_m}, w_m w\rangle$ are already in this set. So, if for every $\ell \in \{1, \ldots, m\}$, the word $w_i w$ is accepted by $\mathcal{A}_i$ starting from s^{k_ℓ}, which means that $s^{k_\ell} \xrightarrow{w_\ell}_i P_\ell$ and $\forall p \in P_\ell.\ p \xrightarrow{w}_i Q_i \subseteq F$, then, due to the new transition, the word γw is accepted by $\mathcal{A}_{i+1}$ starting from s^j. Notice that the new transition imposes that only words w that are accepted starting from all the states in the P_ℓ's can be considered (w is in the intersection of the languages of all these states). The use of alternating automata allows to represent this intersection without modification of the number of states of the original automaton $\mathcal{A}$. This is crucial for the termination argument of the construction.

The following theorem, which shows the correcteness of the construction of $\mathcal{A}_{pre^*}$, is proved in the full paper:

Theorem 4.1 *Given an APDS $\mathcal{P}$ and a regular set of configurations recognized by a $\mathcal{P}$-AMA $\mathcal{A}$, we can construct a $\mathcal{P}$-AMA $\mathcal{A}_{pre^*}$ recognizing $pre^*(Conf(\mathcal{A}))$.*

It follows easily from the facts below that the algorithm is polynomial on the size of $\mathcal{P}$ and (singly) exponential in the size of $\mathcal{A}$:

- $\mathcal{A}_{pre^*}$ has the same states as $\mathcal{A}$,
- a $\mathcal{P}$-AMA with k states has $O(n_{\mathcal{P}} \cdot k \cdot 2^k)$ transitions, where $n_{\mathcal{P}}$ is the size of $\mathcal{P}$, and
- during the construction of the sequence $\mathcal{A}_0, \mathcal{A}_1, \ldots$, polynomial time suffices to decide if a new transition can be added to the current automaton.

5 Model-Checking Branching-Time Temporal Logics

5.1 The alternation-free (propositional) μ-calculus

Let *Prop* be a set of atomic propositions and $\mathcal{X}$ a finite set of variables. The set of formulas of the (propositional) μ-calculus is defined by the following grammar:

$$\varphi ::= \pi \in Prop \mid X \in \mathcal{X} \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists\bigcirc\varphi \mid \mu X.\, \varphi$$

where in formulas of the form $\mu X.\, \varphi$, the variable X must occur in φ under an even number of negations. In addition, we consider the usual abbreviations: the boolean connectives $\wedge$ and $\Rightarrow$, $\forall\bigcirc\varphi = \neg\exists\bigcirc\neg\varphi$, and $\nu X.\ \varphi(X) = \neg\mu X.\, \neg\varphi(\neg X)$. We write $\sigma X.\, \varphi(X)$ for either $\mu X.\, \varphi(X)$ or $\nu X.\, \varphi(X)$.

The notion of free occurrence of a variable in a formula is defined as usual by considering μ and ν as quantifiers. We suppose without loss of generality that in every formula each variable is bound at most once. We write $\varphi(X)$ to indicate that

X occurs free in φ. A formula φ is *closed* if no variable occurs free in it, otherwise it is *open*.

We interpret formulas on the set of configurations of a PDS $\mathcal{P} = (P, \Gamma, \Delta)$. We use a labelling function $\Lambda : P \to 2^{Prop}$, and a valuation $\mathcal{V}$ which assigns to each variable a set of configurations. The set of configurations of $\mathcal{P}$ satisfying a formula φ is denoted by $[\![\varphi]\!]_{\mathcal{P}}(\mathcal{V})$ and is defined by the following rules:

$$\begin{aligned}
[\![\pi]\!]_{\mathcal{P}}(\mathcal{V}) &= \Lambda^{-1}(\pi) \times \Gamma^* \\
[\![X]\!]_{\mathcal{P}}(\mathcal{V}) &= \mathcal{V}(X) \\
[\![\neg\phi]\!]_{\mathcal{P}}(\mathcal{V}) &= (P \times \Gamma^*) \setminus [\![\phi]\!]_{\mathcal{P}}(\mathcal{V}) \\
[\![\phi_1 \vee \phi_2]\!]_{\mathcal{P}}(\mathcal{V}) &= [\![\phi_1]\!]_{\mathcal{P}}(\mathcal{V}) \cup [\![\phi_2]\!]_{\mathcal{P}}(\mathcal{V}) \\
[\![\exists\bigcirc\varphi]\!]_{\mathcal{P}}(\mathcal{V}) &= pre([\![\varphi]\!]_{\mathcal{P}}(\mathcal{V})) \\
[\![\nu X.\phi]\!]_{\mathcal{P}}(\mathcal{V}) &= \bigcup\{\mathcal{C} \subseteq P \times \Gamma^* \mid \mathcal{C} \subseteq [\![\phi]\!]_{\mathcal{P}}(\mathcal{V}[\mathcal{C}/X])\}
\end{aligned}$$

where $\mathcal{V}[\mathcal{C}/X]$ is the valuation which coincides with $\mathcal{V}$ for all variables but X, where it takes the value $\mathcal{C}$.

The set of formulas in *positive normal form* is defined by the following syntax:

$$\varphi ::= \pi \mid \neg\pi \mid X \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists\bigcirc\varphi \mid \forall\bigcirc\varphi \mid \mu X.\, \varphi \mid \nu X.\, \varphi$$

It is easy to show that every formula is equivalent to a formula in positive normal form (push negations inside).

A σ-subformula of a formula $\sigma X.\ \phi(X)$ is *proper* if it does not contain any occurrence of X. The alternation-free μ-calculus is the set of formulas φ in positive normal form such that for every σ-subformula ϕ of φ the following holds:

– if ϕ is a μ-formula, then all its ν-subformulas are proper, and
– if ϕ is a ν-formula, then all its μ-subformulas are proper.

Given a formula φ, we define its *closure* $cl(\varphi)$ as the smallest set of formulas containing φ and such that

– if $\phi_1 \vee \phi_2 \in cl(\varphi)$ or $\phi_1 \wedge \phi_2 \in cl(\varphi)$ then $\phi_1 \in cl(\varphi)$ and $\phi_2 \in cl(\varphi)$,
– if $\exists\bigcirc\phi \in cl(\varphi)$ or $\forall\bigcirc\phi \in cl(\varphi)$ then $\phi \in cl(\varphi)$,
– if $\sigma X.\ \phi(X) \in cl(\varphi)$ then $\phi(\sigma X.\ \phi(X)) \in cl(\varphi)$.

It is easy to see that the closure of a formula is always a finite set, and that its cardinality is bounded by the length of the formula.

The Model-Checker Consider a PDS $\mathcal{P} = (P, \Gamma, \Delta)$ and a labelling function $\Lambda : P \to 2^{Prop}$. Let φ be a formula of the alternation-free μ-calculus, and let $\mathcal{V}$ be a valuation of the free variables in φ.

We show how to construct an AMA $\mathcal{A}_\varphi$ recognizing $[\![\varphi]\!]_{\mathcal{P}}(\mathcal{V})$. From now on we drop the indices and write just $[\![\varphi]\!]$.

We start by considering the case where all the σ-subformulas of φ are μ-formulas. We construct an APDS $\mathcal{AP}$ which is, roughly speaking, the product of $\mathcal{P}$ and the alternating automaton corresponding to φ [14]; we then reduce the problem of computing $[\![\varphi]\!]$ to computing the value of $pre^*_{\mathcal{A}P}$ for a certain regular set of configurations. Intuitively, a configuration $\langle [p,\phi], w\rangle$ belongs to this set if ϕ is a basic formula of the form π, $\neg\pi$, or X, for X free in ϕ, and the configuration $\langle p, w\rangle$ of $\mathcal{P}$ satisfies ϕ. Observe that whether $\langle p, w\rangle$ satisfies ϕ or not can be decided by direct inspection of the labelling function Λ and the valuation $\mathcal{V}$. The AND-branching in the transition rules of $\mathcal{A}P$ is due to conjunctions and universal path quantifications (in $\forall\bigcirc$ operators) occurring in the formula φ.

Formally, we define the APDS $\mathcal{A}P = (P^{\varphi}_{\mathcal{P}}, \Gamma, \Delta^{\varphi}_{\mathcal{P}})$ where

- $P^{\varphi}_{\mathcal{P}} = P \times cl(\varphi)$,
- $\Delta^{\varphi}_{\mathcal{P}}$ is the smallest set of transition rules satisfying the following conditions for every control location $[p,\phi]$ and every stack symbol γ:
  - if $\phi = \phi_1 \vee \phi_2$, then $([p,\phi],\gamma) \hookrightarrow ([p,\phi_1],\gamma)$ and $([p,\phi],\gamma) \hookrightarrow ([p,\phi_2],\gamma)$,
  - if $\phi = \phi_1 \wedge \phi_2$, then $([p,\phi],\gamma) \hookrightarrow \{\ ([p,\phi_1],\gamma), ([p,\phi_2],\gamma)\ \}$,
  - if $\phi = \mu X.\, \psi(X)$, then $([p,\phi],\gamma) \hookrightarrow ([p,\psi(\phi)],\gamma)$,
  - if $\phi = \exists\bigcirc\psi$ and $(p,\gamma) \hookrightarrow (q,w)$ is a transition rule of $\mathcal{P}$, then $([p,\phi],\gamma) \hookrightarrow ([q,\psi],w)$,
  - if $\phi = \forall\bigcirc\psi$ then $([p,\phi],\gamma) \hookrightarrow \{([q,\psi],w) \mid (p,\gamma) \hookrightarrow (q,w)\}$.

Let $\mathcal{C}_t$ (where the index t stands for true) be the subset of configurations of $\mathcal{A}P$ containing all configurations of the form

- $\langle [p,\pi], w\rangle$, where $\pi \in \Lambda(p)$,
- $\langle [p,\neg\pi], w\rangle$, where $\pi \notin \Lambda(p)$,
- $\langle [p,X], w\rangle$, where X is free in φ and $\langle p, w\rangle \in \mathcal{V}(X)$.

Clearly, if $\mathcal{V}(X)$ is a regular set of configurations for every variable X free in φ, then $\mathcal{C}_t$ is also a regular set of configurations.

The following result can be easily proved using standard techniques based on the notion of signature [9]:

Proposition 5.1 *Let $\mathcal{A}P$ be the APDS obtained from $\mathcal{P}$ and φ using the construction above. A configuration $\langle p, w\rangle$ of $\mathcal{P}$ belongs to $[\![\varphi]\!]$ iff the configuration $\langle [p,\varphi], w\rangle$ of $\mathcal{A}P$ belongs to $pre^*_{\mathcal{A}P}(\mathcal{C}_t)$.*

Applying Theorem 2.1 we obtain a procedure to compute an AMA $\mathcal{A}_\varphi$ which accepts exactly the configurations of $\mathcal{P}$ that satisfy φ.

The case in which all the σ-subformulas of φ are ν-subformulas is now easy to solve: the negation of φ is equivalent to a formula φ' in positive normal form whose σ-subformulas are all μ-subformulas. Applying Theorem 2.1 we construct an AMA which accepts the configurations of $\mathcal{P}$ that satisfy φ'. We then just use the fact that AMA's are closed under complementation.

Let us now consider the general case of in which φ is an arbitrary formula of the alternation-free μ-calculus. We can assume without loss of generality that φ is a σ-formula (otherwise a "dummy" fixpoint can be added). The following property (which does *not* hold for the full μ-calculus) follows easily from the definitions, and allows us to construct the AMA $\mathcal{A}_\varphi$. We use the following notation: given a family $\Phi = \{\phi_i\}_{i=1}^n$ of subformulae of φ, which are pairwise incomparable with respect to the subformula relation, and a family $U = \{U_i\}_{i=1}^n$ of fresh variables, $\varphi[U/\Phi]$ denotes the result of simultaneously substituting U_i for ϕ_i in φ.

Proposition 5.2 *Let φ be a μ-formula (ν-formula) of the alternation-free μ-calculus, and let $\Phi = \{\phi_i\}_{i=1}^n$ be the family of maximal ν-subformulas (μ-subformulas) of ϕ with respect to the subformula relation. Then*

$$[\![\varphi]\!] = [\![\varphi[U/\Phi]]\!](\mathcal{V}')$$

where $U = \{U_i\}_{i=1}^n$ is a suitable family of fresh variables, and $\mathcal{V}'$ is the valuation which extends $\mathcal{V}$ by assigning to each U_i the set $[\![\phi_i]\!]$.

Observe that if φ is a μ-formula (ν-formula), then all the σ-subformulas of $\varphi[U/\Phi]$ are also μ-formulas (ν-formulas). Together with Proposition 5.1, this leads immediately to a recursive algorithm for computing $\mathcal{A}_\varphi$: for every $\phi \in \Phi$, compute recursively AMA's $\mathcal{A}_\phi$ recognizing $[\![\phi]\!]$, and then use them and Proposition 5.2 to compute $\mathcal{A}_\varphi$. Consequently we have:

Theorem 5.1 *Let $\mathcal{P}$ be a PDS, let φ a formula of the alternation-free μ-calculus, and let $\mathcal{V}$ be a valuation of the free variables of φ. We can construct an AMA $\mathcal{A}_\varphi$ such that $Conf(\mathcal{A}_\varphi) = [\![\varphi]\!]_\mathcal{P}(\mathcal{V})$.*

Complexity Walukiewicz has shown in [22] that there exists a formula of the alternation-free μ-calculus such that the model checking problem for PDS's and this formula is DEXPTIME-complete. This implies that all model-checking algorithms must have exponential complexity in the size of the system. We show that the algorithm we have obtained (which is very different from the one presented in [22]) has this complexity.

Let $n_\mathcal{P}$ be the size of $\mathcal{P}$ and let n_φ be the length of φ. We define a tree of σ-subformulas of φ: the root of the tree is φ; the children of a μ-subformula (ν-subformula) ϕ are the maximal ν-subformulas (μ-subformulas) of ϕ. Clearly, the number of nodes of the tree does not exceed n_φ.

Let ϕ be a leaf of the tree. The AMA $\mathcal{A}_\phi$ recognizing $[\![\phi]\!]$ is obtained by applying the *pre** construction to the AMA recognizing the set $\mathcal{C}_t$. Since the latter has $O(n_\mathcal{P} \cdot n_\varphi)$ states, $\mathcal{A}_\phi$ has also $O(n_\mathcal{P} \cdot n_\varphi)$ states.

Now, let ϕ be an internal node of the tree with children $\phi_1, \ldots, \phi_k$. If the AMA recognizing $[\![\phi_i]\!]$ has n_i states, then the AMA recognizing $[\![\phi]\!]$ has $O(\Sigma_{i=1}^n n_i + n_\mathcal{P} \cdot n_\varphi)$. Since the number of nodes of the tree does not exceed n_φ, the AMA $\mathcal{A}_\varphi$ recognizing $[\![\varphi]\!]$ has $O(n_\mathcal{P} \cdot n_\varphi^2)$ states. Since each AMA can be constructed in exponential time in the number of states, the algorithm is singly exponential in $n_\mathcal{P}$ and n_φ.

5.2 The logic EF

The alternation-free μ-calculus is a rather powerful logic. Proper sublogics, like CTL, are considered to be sufficiently expressive for many applications. This raises the question whether the model-checking problem for PDS's and some interesting fragment of the alternation-free μ-calculus may lie in some complexity class below DEXPTIME. In this section we show that this is the case: we prove that the model-checking problem for the logic EF (propositional logic plus the temporal operator EF) is in PSPACE.[4] However, the problem turns out to be PSPACE-complete,

[4] We assume $PSPACE \neq DEXPTIME$

even PSPACE-complete in the size of the system. Therefore, the complexity gap between the alternation-free μ-calculus and its sublogics is rather small.

Given a set *Prop* of atomic propositions, the set of formulas of EF is defined by the following grammar:

$$\varphi ::= \pi \in Prop \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists\bigcirc\varphi \mid EF\varphi$$

The semantics of formulas of the form π, $\neg\varphi$, $\varphi_1 \vee \varphi_2$, and $\exists\bigcirc\varphi$ is defined as for the alternation-free μ-calculus. A configuration c satisfies a formula $EF\varphi$ if there exists a configuration c' reachable from c that satisfies φ.

We consider the proof of membership in PSPACE first, since this is the part that makes use of our reachability analysis. The hardness part is a standard reduction from the acceptance problem for linearly bounded Turing machines.

Fix a PDS $\mathcal{P}(P, \Gamma, \Delta)$ and a configuration c of $\mathcal{P}$. Denote by $R(c)$ the set of words $pw \in P\Gamma^*$ such that $\langle p, w\rangle$ is reachable from c. We have the following result (the proof is in the full paper):

Theorem 5.2 *The set $R(c)$ is regular. Moreover, $R(c)$ is recognized by a finite multi automaton having polynomially many states in the sum of the sizes of c and $\mathcal{P}$.*

Given a formula φ of the alternation-free μ-calculus, denote by $R(c, \varphi)$ the set of words $pw \in P\Gamma^*$ such that $\langle p, w\rangle$ is reachable from c *and* satisfies φ. By Theorem 5.2 and Theorem 5.1, we have:

Corollary 5.1 *Let φ be a formula of the alternation-free μ-calculus. The set $R(c, \varphi)$ is regular, and is recognized by an alternating finite automaton having polynomially many states in the sum of the sizes of c, $\mathcal{P}$ and φ.*

We can now obtain our first result concerning the logic EF:

Theorem 5.3 *The model checking problem for the logic EF and pushdown automata is in PSPACE.*

Proof. Let $\mathcal{P}$ be a PDS and let φ be a formula of EF. We show by induction on the structure of φ that the problem of deciding if a given configuration c of $\mathcal{P}$ satisfies ϕ can be solved in nondeterministic polynomial space in the size of c, $\mathcal{P}$ and φ.

The cases $\varphi = \pi, \varphi_1 \vee \varphi_2, \neg\varphi_1$ are trivial. So let $\varphi = EF\varphi_1$, and assume that we can decide whether a configuration c satisfies φ_1 using nondeterministic polynomial space in the size of c, $\mathcal{P}$ and φ_1.

By the definition of the semantics of EF, $\mathcal{P}$ satisfies φ iff there exists a configuration c_1 reachable from c which satisfies φ_1.

If an AMA with n states recognizes a nonempty set, then it recognizes some word of length at most n. Therefore, by Corollary 5.1, we can assume that c_1 has polynomial size in $\mathcal{P}$ and φ_1. The following nondeterministic algorithm decides in polynomial space if $\mathcal{P}$ satisfies φ:

- Guess a configuration c_1 of polynomial size in $\mathcal{P}$ and φ;
- Check in polynomial time in the size of c_1 and $\mathcal{P}$ that c_1 is reachable from c;

– Check in polynomial space in the size of c_1, $\mathcal{P}$ and φ_1 that c_1 satisfies φ_1.

The membership of the model checking in PSPACE follows now from NPSPACE = PSPACE. □

In the full paper, we give the proof of the following hardness result:

Theorem 5.4 *The model checking problem for the logic EF and pushdown systems is PSPACE-hard.*

Finally, from the proof of Theorem 5.4, we can deduce that the following stronger result also holds:

Corollary 5.2 *There is a formula φ of EF such that the problem of deciding if a PDS satisfies φ is PSPACE-complete.*

6 Conclusion

We have applied the "*symbolic*" analysis principle to a class of infinite state systems, namely pushdown systems. We have represented (possibly infinite) sets of configurations using finite-state automata, and have proposed a simple procedure to compute sets of predecessors. Using this procedure and the automata-theoretic approach to model-checking, we have obtained model-checking algorithms for both linear and branching-time properties. From these results we have derived upper bounds for several model-checking problems. We have also provided matching lower bounds by means of some reductions based on Walukiewicz's ideas [22].

The model-checking problem for pushdown systems and the modal μ-calculus (or its alternation-free fragment) has been studied in several papers [11, 12, 22]. The main advantage of our approach (apart from an homogeneous treatment of both branching-time and linear-time logics) is the simplicity of our algorithms: only well known concepts from automata theory are needed to understand them. They constitute smooth generalizations of global model-checking algorithms for branching-time logics and finite-state systems.

An approach similar to ours, based on automata representation of the stack contents, has been adopted in [15]. However, the techniques used there are different from ours, and the branching-time properties are expressed there in a logic (CTL*) which is incomparable with the alternation-free modal μ-calculus.

We do not know whether our approach can be extended to the full modal μ-calculus. The exact complexity of the model-checking problem for pushdown systems and CTL is also open: it lies somewhere between PSPACE and DEXPTIME. These questions, and the extension of the symbolic analysis principle to other classes of systems with infinite state spaces, are left for future investigations.

References

1. A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1976.

2. R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *TCS*, 138, 1995.
3. R. Alur and D. Dill. A Theory of Timed Automata. *TCS*, 126, 1994.
4. E. Asarin, O. Maler, and A. Pnueli. Symbolic Controller Synthesis for Discrete and Timed Systems. In *Hybrid Systems II.* LNCS 999, 1995.
5. B. Boigelot and P. Godefroid. Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs. In *CAV'96.* LNCS 1102, 1996.
6. R.V. Book and F. Otto. *String-Rewriting Systems.* Springer-Verlag, 1993.
7. A. Bouajjani, J. Esparza, and O. Maler. Reachability Analysis of Pushdown Automata: Application to Model Checking. Tech. Rep. VERIMAG, 1997.
`ftp://ftp.imag.fr/imag/SPECTRE/ODED/pda.ps.gz`,
`http://papa.informatik.tu-muenchen.de/forschung/sfb342_a3/refs.html`.
8. A. Bouajjani and O. Maler. Reachability Analysis of Pushdown Automata. In *Infinity'96.* tech. rep. MIP-9614, Univ. Passau, 1996.
9. J.C. Bradfield. *Verifying Temporal Properties of Systems.* Birkhauser, 1992.
10. R. Bryant. Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. *ACM Computing Surveys*, 24, 1992.
11. O. Burkart and B. Steffen. Model Checking for Context-Free Processes. In *CONCUR'92*, 1992. LNCS 630.
12. O. Burkart and B. Steffen. Composition, Decomposition and Model-Checking of Pushdown Processes. *Nordic Journal of Computing*, 2, 1995.
13. E.M. Clarke, E.A. Emerson, and E. Sistla. Automatic Verification of Finite State Concurrent Systems using Temporal Logic Specifications: A Practical Approach. In *POPL'83.* ACM, 1983.
14. E.A. Emerson. Automated Temporal Reasoning about Reactive Systems. In *Logics for Concurrency.* LNCS 1043, 1996.
15. A. Finkel, B. Willems, and P. Wolper. A Direct Symbolic Approach to Model Checking Pushdown Systems. In *Personal communication*, 1997.
16. G. Holzmann. Basic SPIN manual. Technical report, Bell Laboratories, 1994.
17. K.L. McMillan. *Symbolic Model-Checking: an Approach to the State-Explosion Problem.* Kluwer, 1993.
18. A. Pnueli. The Temporal Logic of Programs. In *FOCS'77.* IEEE, 1977.
19. M.Y. Vardi. A Temporal Fixpoint Calculus. In *POPL'88.* ACM, 1988.
20. M.Y. Vardi. Alternating Automata and Program Verification. In *Computer Science Today.* LNCS 1000, 1995.
21. M.Y. Vardi and P. Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *LICS'86.* IEEE, 1986.
22. I. Walukiewicz. Pushdown Processes: Games and Model Checking. In *CAV'96.* LNCS 1102, 1996.