Symbolic Backwards-Reachability Analysis for Higher-Order Pushdown Systems[⋆]

M. Hague C.-H. L. Ong

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, UK, OX1 3QD

Abstract. Higher-order pushdown systems (PDSs) generalise pushdown systems through the use of higher-order stacks, that is, a nested "stack of stacks" structure. We further generalise higher-order PDSs to higher-order Alternating PDSs (APDSs) and consider the backwards reachability problem over these systems. We prove that given an order-n APDS, the set of configurations from which a given regular set of configurations is reachable is itself regular and computable in n-EXPTIME. We show that the result has several useful applications in the verification of higher-order PDSs such as LTL model checking, alternation-free μ-calculus model checking, and the computation of winning regions of reachability games.

1 Introduction

Pushdown automata are an extension of finite state automata. In addition to a finite set of control states, a pushdown automaton has a stack that can be manipulated with the usual push and pop operations. *Higher-order pushdown automata* (PDA) generalise pushdown automata through the use of higher-order stacks. Whereas a stack in the sense of a pushdown automaton is an order-one stack — that is, a stack of characters — an order-two stack is a stack of order-one stacks. Similarly, an order-three stack is a stack of order-two stacks, and so on.

Higher-order PDA were originally introduced by Maslov [18] in the 1970s as generators of (a hierarchy of) finite word languages. *Higher-order pushdown systems* (PDSs) are higher-order PDA viewed as generators of infinite trees or graphs. These systems provide a natural infinite-state model for higher-order programs with recursive function calls and are therefore useful in software verification. Several notable advances in recent years have sparked off a resurgence of interest in higher-order PDA/PDSs in the Verification community. E.g. Knapik *et al.* [24] have shown that the ranked trees generated by deterministic order-n PDSs are exactly those that are generated by order-n recursion schemes satisfying the *safety* constraint; Carayol and Wöhrle [5] have shown that the ϵ-closure of the configuration graphs of higher-order PDSs exactly constitute Caucal's

[⋆] The full version [13] of this work is downloadable from the first author's web page.

graph hierarchy [10]. Remarkably these infinite trees and graphs have decidable monadic second-order (MSO) theories [11, 5, 24].

These MSO decidability results, though powerful, only allow us to check that a property holds from a given configuration. We may wish to compute the set of configurations that satisfy a given property, especially since there may be an infinite number of such configurations. In this paper, we consider a closely-related problem:

> *Backwards Reachability*: Given a set of configurations C_{Init}, compute the set $Pre^*(C_{Init})$ of configurations that can, via any number of transitions, reach a configuration in C_{Init}.

This is an important verification problem in its own right, since safety properties (i.e. undesirable program states – such as deadlock – are never reached) feature largely in practice.

The backwards reachability problem was solved for order-one PDSs by Bouajjani *et al.* [2]. In particular, they gave a method for computing the (regular) set of configurations $Pre^*(C_{Init})$ that could reach a given regular set of configurations C_{Init}. Regular sets of configurations are represented symbolically in the form of a finite multi-automaton. That is, a finite automaton that accepts finite words (representing stacks) with an initial state for each control state of the PDS. A configuration is accepted if the stack (viewed as a word) is accepted from the appropriate initial state. The set $Pre^*(C_{Init})$ is computed by the repeated addition of a number of transitions – determined by the transition relation of the PDS – to the automaton accepting C_{Init}, until a fixed point is reached. A fixed point is guaranteed since no states are added and the alphabet is finite: eventually the automaton will become *saturated*.

The approach was extended by Bouajjani and Meyer [1] to the case of *higher-order context-free processes*, which are higher-order PDSs with a single control state. A key innovation in their work was the introduction of a new class of (finite-state) automata called *nested store automata*, which captures an intuitive notion of regular sets of n-stores. An order-one nested store automaton is simply a finite automaton over words. An order-n nested store automaton is a finite automaton whose transitions are labelled by order-$(n-1)$ nested store automata.

Our paper is concerned with the non-trivial problem[1] of extending the backwards reachability result of Bouajjani and Meyer to the general case of higher-order PDSs (by taking into account a set of control states). In fact, we consider (and solve) the backwards reachability problem for the more general case of higher-order *alternating* pushdown systems (APDSs). Though slightly unwieldy, an advantage of the alternating framework is that it conveniently lends itself to a number of higher-order PDS verification problems. Following the work of Cachat [22], we show that the winning region of a reachability game played over a higher-order PDS can be computed by a reduction to the backwards reachability problem of an appropriate APDS. We also generalise results due to

[1] "This does not seem to be technically trivial, and naïve extensions of our construction lead to procedures which are not guaranteed to terminate." [1, p. 145]

Bouajjani *et al.* [2] to give a method for computing the set of configurations of a higher-order PDS that satisfy a given formula of the alternation-free μ-calculus or a linear-time temporal logic.

Related Work. Prompted by the fact that the set of configurations reachable from a given configuration of a higher-order PDS is not regular in the sense of Bouajjani and Meyer (the stack contents cannot be represented by a finite automaton over words), Carayol [4] has proposed an alternative definition of regularity for higher-order stacks, which we shall call *C-regularity.* Our notion of regularity coincides with that of Bouajjani and Meyer, which we shall call *BM-regularity.*

A set of order-n stacks is said to be *C-regular* if it is constructible from the empty n-stack by a regular sequence of order-n stack operations. Carayol shows that C-regularity coincides with MSO definability over the canonical structure Δ_2^n associated with order-n stacks. This implies, for instance, that the winning region of a parity game over an order-n pushdown graph is also C-regular, as it can be defined as an MSO formula [22].

In this paper we solve the backwards reachability problem for higher-order PDSs and apply the solution to reachability games and model-checking. In this sense we give a weaker kind of result that uses a different notion of regularity. Because C-regularity does not imply BM-regularity[2], our result is not subsumed by the work of Carayol. However, a detailed comparison of the two approaches may provide a fruitful direction for further research.

The definition of higher-order PDSs may be extended to higher-order pushdown games. In the order-one case, the problem of determining whether a configuration is winning for Eloise with a parity winning condition was solved by Walukiewicz in 1996 [14]. The order-one backwards reachability algorithm of Bouajjani *et al.* was adapted by Cachat to compute the winning regions of order-one reachability and Büchi games [22]. Results for pushdown games have been extended to a number of winning conditions [23, 3, 12, 20, 9] including parity conditions [22, 19]. In the higher-order case with a parity winning condition, a method for deciding whether a configuration is winning has been provided by Cachat [22].

Higher-order recursion schemes (HORSs) represent a further area of related work. MSO decidability for trees generated by arbitrary (i.e. not necessarily safe) HORSs has been shown by one of us [21]. A variant kind of higher-order PDSs called *collapsible pushdown automata* (extending *panic automata* [25] or *pushdown automata with links* [16] to all finite orders) has recently been shown to be equi-expressive with HORSs for generating ranked trees [8]. These new automata are conjectured to enrich the class of higher-order systems and provide many new avenues of research.

[2] For example $(push_a)^*; push_2$ defines all stacks of the form $[[a^n][a^n]]$.

2 Preliminaries

In the sequel we will introduce several kinds of alternating automata. For convenience, we will use a non-standard definition of alternating automata that is equivalent to the standard definitions of Brzozowski and Leiss [15] and Chandra, Kozen and Stockmeyer [6]. Similar definitions have been used for the analysis of pushdown systems by Bouajjani *et al.* [2] and Cachat [22]. The alternating transition relation $\Delta \subseteq \mathcal{Q} \times \Gamma \times 2^{\mathcal{Q}}$ — where Γ is a kind of alphabet and $\mathcal{Q}$ is a state-set — is given in disjunctive normal form. That is, the image $\Delta(q, \gamma)$ of $q \in \mathcal{Q}$ and $\gamma \in \Gamma$ is a set $\{Q_1, \ldots, Q_m\}$ with $Q_i \in 2^{\mathcal{Q}}$ for $i \in \{1, \ldots, m\}$. When the automaton is viewed as a game, Eloise — the existential player — chooses a set $Q \in \Delta(q, \gamma)$; Abelard — the universal player — then chooses a state $q \in Q$.

2.1 (Alternating) Higher-Order Pushdown Systems

We begin by defining higher-order stores and their operations. We will then define higher-order PDSs and APDSs in full.

The set C_1^{Σ} of 1-stores over an alphabet Σ is the set of words of the form $[a_1, \ldots, a_m]$ with $m \geq 0$ and $a_i \in \Sigma$ for all $i \in \{1, \ldots, m\}$, $[\notin \Sigma$ and $] \notin \Sigma$. For $n > 1$, $C_n^{\Sigma} = [w_1, \ldots, w_m]$ with $m \geq 1$ and $w_i \in C_{n-1}^{\Sigma}$ for all $i \in \{1, \ldots, m\}$. There are three types of operations applicable to n-stores: *push*, *pop* and *top*. These are defined inductively. Over a 1-store, we have (for all $w \in \Sigma^*$),

$$\begin{aligned} push_w[a_1 \ldots a_m] &= [w a_2 \ldots a_m] \\ top_1[a_1 \ldots a_m] &= a_1 \end{aligned}$$

We may define the abbreviation $pop_1 = push_{\varepsilon}$. When $n > 1$, we have

$$\begin{aligned} push_w[\gamma_1 \ldots \gamma_m] &= [push_w(\gamma_1)\gamma_2 \ldots \gamma_m] && \\ push_l[\gamma_1 \ldots \gamma_m] &= [push_l(\gamma_1)\gamma_2 \ldots \gamma_m] && \text{if } 2 \leq l < n \\ push_n[\gamma_1 \ldots \gamma_m] &= [\gamma_1 \gamma_1 \gamma_2 \ldots \gamma_m] && \\ pop_l[\gamma_1 \ldots \gamma_m] &= [pop_l(\gamma_1)\gamma_2 \ldots \gamma_m] && \text{if } 1 \leq l < n \\ pop_n[\gamma_1 \ldots \gamma_m] &= [\gamma_2 \ldots \gamma_m] && \text{if } m > 1 \\ top_l[\gamma_1 \ldots \gamma_m] &= top_l(\gamma_1) && \text{if } 1 \leq l < n \\ top_n[\gamma_1 \ldots \gamma_m] &= \gamma_1 && \end{aligned}$$

Note that we assume wlog $\Sigma \cap \mathcal{N} = \emptyset$, where $\mathcal{N}$ is the set of natural numbers. Further, observe that when $m = 1$, pop_n is undefined. We define $\mathcal{O}_n = \{\ push_w \mid w \in \Sigma^*\ \} \cup \{\ push_l, pop_l \mid 1 < l \leq n\ \}$.

Definition 1. An *order-n pushdown system* (PDS) is a tuple $(\mathcal{P}, \mathcal{D}, \Sigma)$ where $\mathcal{P}$ is a finite set of control states p^j, $\mathcal{D} \subseteq \mathcal{P} \times \Sigma \times \mathcal{O}_n \times \mathcal{P}$ is a finite set of commands d, and Σ is a finite alphabet.

A configuration of an order-n PDS is a pair $\langle p, \gamma \rangle$ where $p \in \mathcal{P}$ and γ is an n-store. We have a transition $\langle p, \gamma \rangle \hookrightarrow \langle p', \gamma' \rangle$ iff we have $(p, a, o, p') \in \mathcal{D}$, $top_1(\gamma) = a$ and $\gamma' = o(\gamma)$.

Definition 2. An *order-n alternating pushdown system* (APDS) is a tuple $(\mathcal{P}, \mathcal{D}, \Sigma)$ where $\mathcal{P}$ is a finite set of control states p^j, $\mathcal{D} \subseteq \mathcal{P} \times \Sigma \times 2^{\mathcal{O}_n \times \mathcal{P}}$ is a finite set of commands d, and Σ is a finite alphabet.

A configuration of an order-n APDS is a pair $\langle p, \gamma \rangle$ where $p \in \mathcal{P}$ and γ is an n-store. We have a transition $\langle p, \gamma \rangle \hookrightarrow C$ iff we have $(p, a, OP) \in \mathcal{D}$, $top_1(\gamma) = a$, and

$$\begin{aligned} C = \{\ & \langle p', \gamma' \rangle \mid (o, p') \in OP \ \wedge\ \gamma' = o(\gamma)\ \} \\ \cup\ \{\ & \langle p, \triangledown \rangle \mid \text{if } (o, p') \in OP \text{ and } o(\gamma) \text{ is not defined } \} \end{aligned}$$

The transition relation generalises to sets of configurations via the following rule:

$$\frac{\langle p, \gamma \rangle \ \hookrightarrow\ C}{C' \cup \langle p, \gamma \rangle \ \hookrightarrow\ C' \cup C} \quad \langle p, \gamma \rangle \notin C'$$

In both the alternating and the non-alternating cases, we define $\overset{*}{\hookrightarrow}$ to be the transitive closure of $\hookrightarrow$. For a set of configurations C_{Init} we define $Pre^*(C_{Init})$ as the set of configurations $\langle p, \gamma \rangle$ such that $\langle p, \gamma \rangle \overset{*}{\hookrightarrow} c$ and $c \in C_{Init}$ or $\langle p, \gamma \rangle \overset{*}{\hookrightarrow} C$ and $C \subseteq C_{Init}$ respectively.

Observe that since no transitions are possible from an "undefined" configuration $\langle p, \triangledown \rangle$ we can reduce the reachability problem for higher-order PDSs to the reachability problem over higher-order APDSs in a straightforward manner.

In the sequel, to ease the presentation, we assume $n > 1$. The case $n = 1$ was investigated by Bouajjani *et al.* [2].

2.2 n-Store Multi-Automata

To represent sets of configurations we will use *n-store multi-automata*. These are alternating automata whose transitions are labelled by $(n-1)$-*store automata*, which are also alternating. A set of configurations is said to be *regular* if it is accepted by an n-store multi-automaton.

Definition 3.

1. A *1-store automaton* is a tuple $(\mathcal{Q}, \Sigma, \Delta, q_0, \mathcal{Q}_f)$ where $\mathcal{Q}$ is a finite set of states, Σ is a finite alphabet, q_0 is the initial state and $\mathcal{Q}_f \subseteq \mathcal{Q}$ is a set of final states. $\Delta \subseteq \mathcal{Q} \times \Sigma \times 2^{\mathcal{Q}}$ is a finite transition relation.
2. Let $\mathfrak{B}^{\Sigma}_{n-1}$ be the (infinite) set of all $(n-1)$-store automata over the alphabet Σ. An *n-store automaton* over the alphabet Σ is a tuple $(\mathcal{Q}, \Sigma, \Delta, q_0, \mathcal{Q}_f)$ where $\mathcal{Q}$ is a finite set of states, $q_0 \notin \mathcal{Q}_f$ is the initial state, $\mathcal{Q}_f \subseteq \mathcal{Q}$ is a set of final states, and $\Delta \subseteq \mathcal{Q} \times \mathfrak{B}^{\Sigma}_{n-1} \times 2^{\mathcal{Q}}$ is a *finite* transition relation.
3. An *n-store multi-automaton* over the alphabet Σ is a tuple

$$(\mathcal{Q}, \Sigma, \Delta, \{q^1, \ldots, q^z\}, \mathcal{Q}_f)$$

where $\mathcal{Q}$ is a finite set of states, Σ is a finite alphabet, $q^i \notin \mathcal{Q}_f$ for $i \in \{1, \ldots, z\}$ are separate initial states and $\mathcal{Q}_f \subseteq \mathcal{Q}$ is a set of final states, and

$$\Delta \subseteq (\mathcal{Q} \times \mathfrak{B}^{\Sigma}_{n-1} \times 2^{\mathcal{Q}}) \cup (\{q^1, \ldots, q^z\} \times \{\triangledown\} \times \{q^{\varepsilon}_f\})$$

is a *finite* transition relation where $q^{\varepsilon}_f \in \mathcal{Q}_f$ has no outgoing transitions.

To indicate a transition $(q, B, \{q_1, \ldots, q_m\}) \in \Delta$, we write,

$$q \xrightarrow{B} \{q_1, \ldots, q_m\}$$

Paths of the automata from a state q take the form,

$$q \xrightarrow{\widetilde{B}_0} \{q_1^1, \ldots, q_{m_1}^1\} \xrightarrow{\widetilde{B}_1} \ldots \xrightarrow{\widetilde{B}_m} \{q_1^m, \ldots, q_{m_l}^m\}$$

where transitions between configurations $\{q_1^x, \ldots, q_{m_x}^x\} \xrightarrow{\widetilde{B}_x} \{q_1^{x+1}, \ldots, q_{m_{x+1}}^{x+1}\}$ are such that we have $q_y^x \xrightarrow{B_y} Q_y$ for all $y \in \{1, \ldots, m_x\}$ and $\bigcup_{y \in \{1,\ldots,m_x\}} Q_y = \{q_1^{x+1}, \ldots, q_{m_{x+1}}^{x+1}\}$ and $\bigcup_{y \in \{1,\ldots,m_x\}} \{B_y\} = \widetilde{B}_x$. Observe that $\widetilde{B}_0$ is necessarily a singleton set.

We will, by abuse of notation, abbreviate a run over the word w to

$$q \xrightarrow{w} \{q_1, \ldots, q_m\}$$

Further, when a run occurs in an automaton forming part of a sequence indexed by i (for example, $A_0, A_1, \ldots$), we may write $\longrightarrow_i$ to indicate which automaton the run belongs to.

A 1-store $[a_1 \ldots a_m]$ is accepted by a 1-store automaton A (that is $[a_1 \ldots a_m] \in \mathcal{L}(A)$) iff we have a run $q_0 \xrightarrow{a_1 \ldots a_m} Q$ in A with $Q \subseteq \mathcal{Q}_f$. For a given n-store automaton $A = (\mathcal{Q}, \Sigma, \Delta, q_0, \mathcal{Q}_f)$ we define

$$\mathcal{L}(A) = \{\ [\gamma_1 \ldots \gamma_m] \mid q_0 \xrightarrow{\widetilde{B}_0} \ldots \xrightarrow{\widetilde{B}_m} Q \ \wedge\ Q \subseteq \mathcal{Q}_f \ \wedge\ \forall 0 \leq i \leq m. \gamma_i \in \mathcal{L}(\widetilde{B}_i)\ \}$$

where $\gamma \in \mathcal{L}(\widetilde{B})$ iff $\gamma \in \mathcal{L}(B)$ for all $B \in \widetilde{B}$.

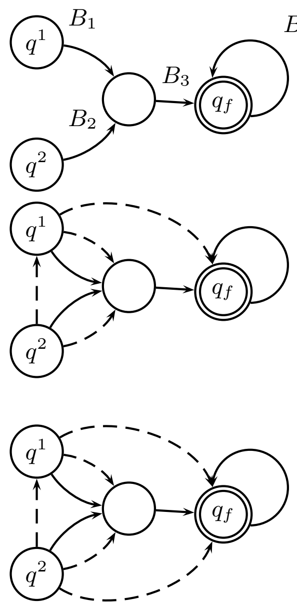
For an n-store multi-automaton $A = (Q, \Sigma, \Delta, \{q^1, \ldots, q^z\}, \mathcal{Q}_f)$ we define

$$\begin{aligned}
\mathcal{L}(A^{q^j}) = \{\ [\gamma_1 \ldots \gamma_m] \mid q^j \xrightarrow{\widetilde{B}_0} \ldots \xrightarrow{\widetilde{B}_m} Q & \\
\wedge\ Q \subseteq \mathcal{Q}_f \ \wedge\ \forall 0 \leq i \leq m. \gamma_i \in \mathcal{L}(\widetilde{B}_i)\ \} & \\
\cup \{\ \nabla \mid q^j \xrightarrow{\nabla} q_f^{\varepsilon}\ \} & \\
\mathcal{L}(A) = \{\ \langle p^j, \gamma \rangle \mid j \in \{1, \ldots, z\} \wedge \gamma \in \mathcal{L}(A^{q^j})\ \} &
\end{aligned}$$

Finally, we define the automata B_l^a and X_l^a for all $1 \leq l \leq n$ and $a \in \Sigma$ and the notation q^θ. B_l^a is the l-store automaton that accepts any l-store γ such that $top_1(\gamma) = a$. X_l^a is the $(n-1)$-store automaton accepting all $(n-1)$-stores such that $top_1(\gamma) = a$ and $top_{l+1}(\gamma) = [[w']]$ for some w'. That is, $pop_l(\gamma)$ is undefined. If θ represents a store automaton, the state q^θ refers to the initial state of the automaton represented by θ.

3 Backwards Reachability

Theorem 1. *Given an n-store multi-automaton A_0 accepting the set of configurations C_{Init} of an order-n APDS, we can construct in n-EXPTIME (in the size of A_0) an n-store multi-automaton A_* accepting the set $Pre^*(C_{Init})$. Thus, $Pre^*(C_{Init})$ is regular.*



$$d_1 = (p^1, a, push_2, p^1)$$
$$d_2 = (p^1, a, push_\varepsilon, p^1),$$
$$d_3 = (p^2, a, push_w, p^1)$$
$$d_4 = (p^2, a, pop_2, p^1)$$

$$\begin{aligned}
\widetilde{G}^1_{(q^1,\circ)} &= \{\{(a, push_\varepsilon, B_1)\}\} \\
\widetilde{G}^1_{(q^1,q_f)} &= \{\{B_1^a, B_1, B_3\}\} \\
\widetilde{G}^1_{(q^2,\circ)} &= \{\{(a, push_w, B_1)\}\} \\
\widetilde{G}^1_{(q^2,q^1)} &= \{\{B_1^a\}\}
\end{aligned}$$

$$\begin{aligned}
\widetilde{G}^2_{(q^1,\circ)} &= \left\{\{(a, push_\varepsilon, B_1)\}, \{(a, push_\varepsilon, \widetilde{G}^1_{(q^1,\circ)})\}\right\} \\
\widetilde{G}^2_{(q^1,q^f)} &= \left\{\begin{array}{l} \{B_1^a, B_1, B_3\}, \{(a, push_\varepsilon, \widetilde{G}^1_{(q^1,q_f)})\}, \\ \{B_1^a, \widetilde{G}^1_{(q^1,q_f)}, B_4\}, \{B_1^a, \widetilde{G}^1_{(q^1,\circ)}, B_3\} \end{array}\right\} \\
\widetilde{G}^2_{(q^2,\circ)} &= \left\{\{(a, push_w, B_1)\}, \{(a, push_w, \widetilde{G}^1_{(q^1,\circ)})\}\right\} \\
\widetilde{G}^2_{(q^2,q^1)} &= \{\{B_1^a\}\} \\
\widetilde{G}^2_{(q^2,q_f)} &= \left\{\{(a, push_w, \widetilde{G}^1_{(q^1,q_f)})\}\right\}
\end{aligned}$$

Fig. 1. The automata A_0, A_1 and A_2.

Due to space constraints, we restrict our attention in the sequel to the order-2 case. We give a brief description of the order-n construction in Section 3.5. For a formal treatment of the general case, we refer the reader to the full version of this paper [13].

Fix an order-2 APDS. We begin by showing how to generate an infinite sequence of automata $A_0, A_1, \ldots$, where A_0 is such that $\mathcal{L}(A_0) = C_{Init}$. This sequence is increasing in the sense that $\mathcal{L}(A_i) \subseteq \mathcal{L}(A_{i+1})$ for all i, and sound and complete with respect to $Pre^*(C_{Init})$; that is $\bigcup_{i \geq 0} \mathcal{L}(A_i) = Pre^*(C_{Init})$. To conclude the algorithm, we construct a single automaton A_* such that $\mathcal{L}(A_*) = \bigcup_{i \geq 0} \mathcal{L}(A_i)$.

We assume wlog that all initial states in A_0 have no incoming transitions and there exist in A_0 a state q_f^* from which all valid 2-stores are accepted and a state $q_f^\varepsilon \in \mathcal{Q}_f$ that has no outgoing transitions.

3.1 Example

We give an intuitive explanation of the algorithm by means of an example. Fix the 2-state order-2 PDS and 2-store multi-automaton A_0 shown in Figure 1 with some B_1, B_2, B_3 and B_4.

We proceed via a number of iterations, generating the automata $A_0, A_1, \ldots$. We construct A_{i+1} from A_i to reflect an additional inverse application of the commands $d_1, \ldots, d_4$. Rather than manipulating order-1 store automata labelling the edges of A_0 directly, we introduce new transitions (at most one between each pair of states q_1 and q_2) and label these edges with the set $\widetilde{G}^1_{(q_1,q_2)}$. This set is a recipe for the construction of an order-1 store automaton that will ultimately label the edge. The resulting A_1 is given in Figure 1 along with the contents of the sets.

To process the command d_1 we need to add all configurations of the form $\langle p_1, [\gamma_1 \dots \gamma_m]\rangle$ with $top_1(\gamma_1) = a$ to the set of configurations accepted by A_1 for each configuration $\langle p_1, [\gamma_1\gamma_1 \dots \gamma_m]\rangle$ accepted by A_0. This results in the transition from q^1 to q_f. The contents of $\widetilde{G}^1_{(q^1,q_f)}$ indicate that this edge must accept the product of B^a_1, B_1 and B_3.

The commands d_2 and d_3 update the top_2 stack of any configuration accepted from q^1 or q^2 respectively. In both cases this updated stack must be accepted from q^1 in A_0. Hence, the contents of $\widetilde{G}^1_{(q^1,\circ)}$ and $\widetilde{G}^1_{(q^2,\circ)}$ specify that the automaton B_1 must be manipulated to produce the automaton that will label these new transitions. Finally, d_4 requires an additional top_2 stack with a as its top_1 element to be added to any stack accepted from q^1. Thus, we introduce the transition from q^2 to q^1.

To construct A_2 from A_1 we repeat the above procedure, taking into account the additional transitions in A_1. Observe that we do not add additional transitions between pairs of states that already have a transition labelled by a set. Instead, each labelling set may contain several element sets. The resulting automaton is given in Figure 1.

If we were to repeat this procedure to construct A_3 we would notice that a kind of fixed point has been reached. In particular, the transition structure of A_3 will match that of A_2 and each $\widetilde{G}^3_{(q,q')}$ will match $\widetilde{G}^2_{(q,q')}$ in everything but the indices of the labels $\widetilde{G}^1_{(\_,\_)}$ appearing in the element sets. We may write $\widetilde{G}^3_{(q,q')} = \widetilde{G}^2_{(q,q')}[2/1]$ where the notation $[2/1]$ indicates a substitution of the element indices.

To complete the construction of A_1 (and A_2) we need to construct the automata $G^1_{(q,q')}$ (and $G^2_{(q,q')}$) represented by the labels $\widetilde{G}^1_{(q,q')}$ (and $\widetilde{G}^2_{(q,q')}$) for the appropriate q, q'. Because these new automata will be constructed from the automata labelling the edges of A_0 (and A_1) we construct them simultaneously, constructing a single (1-store multi-)automaton $\mathcal{G}^1$ (resp. $\mathcal{G}^2$) with an initial state $g^1_{(q,q')}$ for each $G^1_{(q,q')}$. The automaton $\mathcal{G}^1$ is constructed through the addition of states and transitions to the disjoint union of $B_1, \dots, B_4, B^a_1$. Similarly, $\mathcal{G}^2$ is built through the addition of states and transitions to $\mathcal{G}^1$. This procedure is illustrated in Figure 2. For the sake of clarity, many states and transitions have been omitted. All transitions are labelled a.

In Figure 2, the innermost frame gives the disjoint union of the automata B_1, B_3 and B^a_1. The middle frame shows an excerpt of $\mathcal{G}^1$. The transition from $g^1_{(q^1,\circ)}$ is derived from the $push_\varepsilon$ command applied to B_1, which behaves as a pop command. We can then construct $\mathcal{G}^1_{(q^1,\circ)}$ directly from $\mathcal{G}^1$ taking $g^1_{(q^1,\circ)}$ as the initial state.

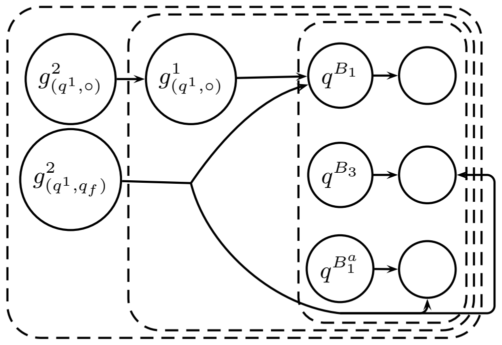


Fig. 2. The automata $\mathcal{G}^0$, $\mathcal{G}^1$ and $\mathcal{G}^2$.

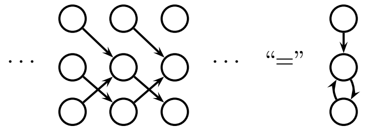


Fig. 3. Collapsing a repetitive chain of new states.

The outermost frame gives a partial representation of the automaton $\mathcal{G}^2$. The transition shown from $g^2_{(q^1,\circ)}$ derives from the *push*$_\varepsilon$ command applied to $\mathcal{G}^1_{(q^1,\circ)}$. Omitted from the diagram is an a-transition from $g^2_{(q^1,\circ)}$ to q^{B_1} resulting from the *push*$_\varepsilon$ command applied to B_1. The branching transition from $g^2_{(q^1,q_f)}$ derives from the set $\{B_1^a, \widetilde{G}^1_{(q^1,q_f)}, B_3\}$ in $\widetilde{G}^2_{(q^1,\circ)}$. That is, we use the power of alternation to construct the product of the automata $B_1^a, \mathcal{G}^1_{(q^1,q_f)}$ and B_3.

We have now constructed the automata A_1 and A_2. We could then repeat this procedure to generate $A_3, A_4, \ldots$, resulting in an infinite sequence of automata that is sound and complete with respect to $Pre^*(\mathcal{L}(A_0))$.

To construct A_* we observe that since a fixed point was reached at A_2, the update to each $\mathcal{G}^i$ to create $\mathcal{G}^{i+1}$ will use similar recipes and hence become repetitive. This will lead to an infinite chain with an unvarying pattern of edges. This chain can be collapsed as shown in Figure 3.

In particular, we are no longer required to add new states to $\mathcal{G}^2$ to construct $\mathcal{G}^i$ for $i > 2$. Instead, we fix the update instructions $\widetilde{G}^2_{(q,q')}[2/1]$ for all q, q' and manipulate $\mathcal{G}^2$ as we manipulated the order-2 structure of A_0 to create A_1 and A_2. We write $\hat{\mathcal{G}}^i$ to distinguish these automata from the automata $\mathcal{G}^i$ generated without fixing the state-set.

Because Σ and the state-set are finite (and remain unchanged), this procedure will reach another fixed point $\hat{\mathcal{G}}^*$ when the transition relation is *saturated* and $\hat{\mathcal{G}}^i = \hat{\mathcal{G}}^{i+1}$. The automaton A_* has the transition structure that became fixed at A_2 labelled with automata derived from the fixed point $\hat{\mathcal{G}}^*$. This automaton will be sound and complete with respect to $Pre^*(\mathcal{L}(A_0))$.

3.2 Preliminaries

To aid in the construction of an automaton representing $Pre^*(C_{Init})$, we introduce a new kind of transition to the 2-store automata. These new transitions are introduced during the processing of the APDS commands. Furthermore, they are labelled with place-holders that will eventually be converted into 1-store automata.

Between any state q_1 and set of states Q_2 we add at most one transition. We associate this transition with an identifier $\widetilde{G}_{(q_1,Q_2)}$. To describe our algorithm we will define sequences of automata, indexed by i. The identifier $\widetilde{G}^i_{(q_1,Q_2)}$ is associated with a set that acts as a recipe for updating the 1-store automaton described by $\widetilde{G}^{i-1}_{(q_1,Q_2)}$ or creating a new automaton if $\widetilde{G}^{i-1}_{(q_1,Q_2)}$ does not exist. Ultimately, the constructed 1-store automaton will label the new transition.

The sets are in a kind of disjunctive normal form. A set $\{S_1, \ldots, S_m\}$ represents an automaton that accepts the union of the languages accepted by the automata described by $S_1, \ldots, S_m$. Each set $S \in \{S_1, \ldots, S_m\}$ corresponds to a possible effect of a command d at order-1 of the automaton. The automaton described by S accepts the intersection of languages described by its elements. An element that is an automaton B refers directly to the automaton B. Similarly, an identifier $\widetilde{G}^i_{(q_1,Q_2)}$ refers to its corresponding automaton. Finally, an element of the form $(a, push_w, \theta)$ refers to an automaton capturing the effect of applying the inverse of the $push_w$ command to the stacks accepted by the automaton represented by θ; moreover, the top_1 character of the stacks accepted by the new automaton will be a. It is a consequence of construction that for any S added during the algorithm, if $(a, push_w, \theta) \in S$ and $(a', push_{w'}, \theta') \in S$ then $a = a'$.

Formally, to each $\widetilde{G}^i_{(q_1,Q_2)}$ we attach a subset of

$$2^{\mathcal{B} \cup \widetilde{\mathcal{G}}^{i-1} \cup (\Sigma \times \mathcal{O}_1 \times (\mathcal{B} \cup \widetilde{\mathcal{G}}^{i-1}))}$$

where $\mathcal{B}$ is the union of the set of all 1-store automata occurring in A_0 and all automata of the form B^a_1 or X^a_1. Further, we denote the set of all order-1 identifiers $\widetilde{G}^i_{(q,Q)}$ in A_i as $\widetilde{\mathcal{G}}^i$. The sets $\mathcal{B}$ and $\mathcal{O}_1$ are finite by definition. If the state-set at order-1 is fixed, there is a finite bound on the size of the set $\widetilde{\mathcal{G}}^i$ for any i.

Given $\widetilde{\mathcal{G}}^i$, we build the automata for all $\widetilde{G}^i_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^i$ simultaneously. That is, we create a single automaton $\mathcal{G}^i$ associated with the set $\widetilde{\mathcal{G}}^i$. This automaton has a state $g^i_{(q_1,Q_2)}$ for each $\widetilde{G}^i_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^i$. The automaton $G^i_{(q_1,Q_2)}$ labelling the transition $q_1 \longrightarrow_i Q_2$ is the automaton $\mathcal{G}^i$ with $g^i_{(q_1,Q_2)}$ as its initial state.

The automaton $\mathcal{G}^i$ is built inductively. We set $\mathcal{G}^0$ to be the disjoint union of all automata in $\mathcal{B}$. We define $\mathcal{G}^{i+1} = T_{\widetilde{\mathcal{G}}^{i+1}}(\mathcal{G}^i)$ where $T_{\widetilde{\mathcal{G}}^j}(\mathcal{G}^i)$ is given in Definition 4. In Section 3.4 it will be seen that j is not always $(i+1)$.

Definition 4. Given an automaton $\mathcal{G}^i = (\mathcal{Q}^i, \Sigma, \Delta^i, \_, \mathcal{Q}_f)$ and a set of identifiers $\widetilde{\mathcal{G}}^j_1$, we define,

$$\mathcal{G}^{i+1} = T_{\widetilde{\mathcal{G}}^j}(\mathcal{G}^i) = (\mathcal{Q}^{i+1}, \Sigma, \Delta^{i+1}, \_, \mathcal{Q}_f)$$

where $\mathcal{Q}^{i+1} = \mathcal{Q}^i \cup \{ g^j_{(q_1,Q_2)} \mid \widetilde{G}^j_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^j \}$, $\Delta^{i+1} = \Delta^{old} \cup \Delta^{new} \cup \Delta^i$, and,

$$\begin{aligned} \Delta^{old} &= \{ g^j_{(q_1,Q_2)} \xrightarrow{a} Q \mid (g^{j-1}_{(q_1,Q_2)} \xrightarrow{a} Q) \in \Delta^i \} \\ \Delta^{new} &= \left\{ g^j_{(q_1,Q_2)} \xrightarrow{b} Q \mid \widetilde{G}^j_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^j \text{ and } b \in \Sigma \text{ and } (1) \right\} \end{aligned}$$

where (1) requires $\{\alpha_1, \ldots, \alpha_r\} \in \widetilde{G}^j_{(q_1,Q_2)}$, $Q = Q_1 \cup \ldots \cup Q_r$ and for each $t \in \{1, \ldots, r\}$ we have,

- If $\alpha_t = \theta$, then $(q^\theta \xrightarrow{b} Q_t) \in \Delta^i$.
- If $\alpha_t = (a, push_w, \theta)$, then $b = a$ and $q^\theta \xrightarrow{w} Q_t$ is a run of $\mathcal{G}^i$.

3.3 Constructing the Sequence $A_0, A_1, \ldots$

For a given order-n APDS with commands $\mathcal{D}$ we define $A_{i+1} = T_{\mathcal{D}}(A_i)$ where the operation $T_{\mathcal{D}}$ follows.

Definition 5. Given an automaton $A_i = (\mathcal{Q}, \Sigma, \Delta^i, \{q^1, \ldots, q^z\}, \mathcal{Q}_f)$ and a set of commands $\mathcal{D}$, we define,

$$A_{i+1} = T_{\mathcal{D}}(A_i) = (\mathcal{Q}, \Sigma, \Delta^{i+1}, \{q^1, \ldots, q^z\}, \mathcal{Q}_f)$$

where Δ^{i+1} is given below.

We begin by defining the set of labels $\widetilde{G}^{i+1}$. This set contains labels on transitions present in A_i, and labels on transitions derived from $\mathcal{D}$. That is,

$$\widetilde{G}^{i+1} = \left\{ \widetilde{G}^{i+1}_{(q,Q)} \mid (q \xrightarrow{\widetilde{G}^i_{(q,Q)}} Q) \in \Delta^i \right\} \cup \left\{ \widetilde{G}^{i+1}_{(q^j,Q)} \mid (2) \right\}$$

The contents of the sets $\widetilde{G}^{i+1}_{(q,Q)} \in \widetilde{G}^{i+1}$ are defined $\widetilde{G}^{i+1}_{(q^j,Q)} = \{ S \mid (2) \}$ where (2) requires $(p^j, a, \{(o_1, p^{k_1}), \ldots, (o_m, p^{k_m})\}) \in \mathcal{D}$, $Q = Q_1 \cup \ldots \cup Q_m$, $S = S_1 \cup \ldots \cup S_m$ and for each $t \in \{1, \ldots, m\}$ we have,

- If $o_t = push_2$, then $S_t = \{B^a_1\} \cup \widetilde{\theta}_1 \cup \widetilde{\theta}_2$ and there exists a path $q^{k_t} \xrightarrow{\widetilde{\theta}_1}_i Q' \xrightarrow{\widetilde{\theta}_2}_i Q_t$ in A_i.
- If $o_t = pop_2$, then $S_t = \{B^a_1\}$ and $Q_t = \{q^{k_t}\}$. Or, if $q^j \xrightarrow{\triangledown}_i \{q^\varepsilon_f\}$ exists in A_i, we may have $S_t = \{B^a_1\}$ and $Q_t = \{q^\varepsilon_f\}$.
- If $o_t = push_w$ then $S_t = \{(a, push_w, \theta)\}$ and there exists a transition $q^{k_t} \xrightarrow{\theta}_i Q_t$ in A_i.

Finally, we give the transition relation Δ^{i+1}.

$$\Delta^{i+1} = \left\{ q \xrightarrow{B} Q \mid \begin{array}{l} (q \xrightarrow{B} Q) \in \Delta^i \\ \text{and } B \in \mathcal{B} \end{array} \right\} \cup \left\{ q \xrightarrow{\widetilde{G}^{i+1}_{(q,Q)}} Q \mid \widetilde{G}^{i+1}_{(q,Q)} \in \widetilde{\mathcal{G}}^{i+1} \right\}$$

We can construct an automaton whose transitions are 1-store automata by replacing each set $\widetilde{G}^{i+1}_{(q,Q)}$ with the automaton $G^{i+1}_{(q,Q)}$ which is $\mathcal{G}^{i+1}$ with initial state $g^{i+1}_{(q,Q)}$, where $\mathcal{G}^{i+1} = T_{\widetilde{\mathcal{G}}^{i+1}}(\mathcal{G}^i)$. Note that $\mathcal{G}^i$ is assumed by induction.

By repeated applications of $T_{\mathcal{D}}$ we construct the sequence $A_0, A_1, \ldots$ which is sound and complete with respect to $Pre^*(C_{Init})$.

Property 1. For any configuration $\langle p^j, \gamma \rangle$ it is the case that $\gamma \in \mathcal{L}(A_i^{q^j})$ for some i iff $\langle p^j, \gamma \rangle \in Pre^*(C_{Init})$.

3.4 Constructing the Automaton A_*

We need to construct a finite representation of the sequence $A_0, A_1, \ldots$ in a finite amount of time. To do this we will construct an automaton A_* such that $\mathcal{L}(A_*) = \bigcup_{i \geq 0} \mathcal{L}(A_i)$. We begin by introducing some notation and a notion of subset modulo i for the sets $\widetilde{G}^i_{(q_1,Q_2)}$.

Definition 6. Given $\theta \in \mathcal{B} \cup \widetilde{\mathcal{G}}^i$ for some i, let

$$\theta[j/i] = \begin{cases} \theta & \text{if } \theta \in \mathcal{B} \\ G^j_{(q_1,Q_2)} & \text{if } \theta = G^i_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^i \end{cases}$$

For a set S we define $S[j/i]$ such that, $\theta \in S$ iff we have $\theta[j/i] \in S[j/i]$, and $(a, push_w, \theta) \in S$ iff we have $(a, push_w, \theta[j/i]) \in S[j/i]$.We extend the notation $[j/i]$ point-wise to nested sets of sets structures. Finally, we define,

1. $\widetilde{G}^i_{(q_1,Q_2)} \lesssim \widetilde{G}^j_{(q_1,Q_2)}$ iff for each $S \in \widetilde{G}^i_{(q_1,Q_2)}$ we have $S[j-1/i-1] \in \widetilde{G}^j_{(q_1,Q_2)}$.
2. $\widetilde{\mathcal{G}}^i \lesssim \widetilde{\mathcal{G}}^j$ iff for all $\widetilde{G}^i_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^i$ we have $\widetilde{G}^j_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^j$ and $\widetilde{G}^i_{(q_1,Q_2)} \lesssim \widetilde{G}^j_{(q_1,Q_2)}$.

Writing $A \simeq B$ to mean $A \lesssim B$ and $B \lesssim A$, we now show that the sets labelling the transitions of $A_0, A_1, \ldots$ reach a fixed point. Once a fixed point $\widetilde{\mathcal{G}}^i \simeq \widetilde{\mathcal{G}}^{i_1}$ has been reached, we can stop adding new states during the construction of $\mathcal{G}^{i_1}, \mathcal{G}^{i_1+1}, \ldots$.

Property 2. There exists $i_1 > 0$ such that $\widetilde{\mathcal{G}}^i \simeq \widetilde{\mathcal{G}}^{i_1}$ for all $i \geq i_1$.

Proof. (Sketch) Since the order-1 state-set in A_i remains constant and we add at most one transition between any state q_1 and set of states Q_2, there is some i_1 where no more transitions are added at order-2. That $\widetilde{\mathcal{G}}^i \simeq \widetilde{\mathcal{G}}^{i_1}$ for all $i \geq i_1$ follows since the contents of $\widetilde{G}^i_{(q_1,Q_2)}$ and $\widetilde{G}^{i_1}_{(q_1,Q_2)}$ are derived from the same transition structure.

Lemma 1. *Suppose we have a sequence of automata $\mathcal{G}^0, \mathcal{G}^1, \ldots$ and associated sets $\widetilde{\mathcal{G}}^0, \widetilde{\mathcal{G}}^1, \ldots$. Further, suppose there exists an i_1 such that for all $i \geq i_1$ we have $\widetilde{\mathcal{G}}^i \simeq \widetilde{\mathcal{G}}^{i_1}$. We can define a sequence of automata $\hat{\mathcal{G}}^{i_1}, \hat{\mathcal{G}}^{i_1+1}, \ldots$ such that the state-set in $\hat{\mathcal{G}}^i$ remains constant. The following are equivalent for all w,*

1. *The run $g^{i_1}_{(q_1,Q_2)} \xrightarrow{w}_i Q$ with $Q \subseteq \mathcal{Q}_f$ exists in $\hat{\mathcal{G}}^i$ for some i.*
2. *The run $g^{i'}_{(q_1,Q_2)} \xrightarrow{w}_{i'} Q'$ with $Q' \subseteq \mathcal{Q}_f$ exists in $\mathcal{G}^{i'}$ for some i'.*

We use $\hat{\mathcal{G}}^{i+1} = T_{\widetilde{\mathcal{G}}^{i_1}[i_1/i_1-1]}(\hat{\mathcal{G}}^i)$ to construct the sequence $\hat{\mathcal{G}}^{i_1}, \hat{\mathcal{G}}^{i_1+1}, \ldots$. Intuitively, since the transitions from the states introduced to define $\mathcal{G}^i$ for $i \geq i_1$ are derived from similar sets, we can compress the subsequent repetition into a single set of new states as shown in Figure 3. Since the state-set of this new sequence does not change and the alphabet Σ is finite, the transition structure will become saturated.

Property 3. For a sequence of automata $\mathcal{G}^0, \mathcal{G}^1, \ldots$ such that the state-set of $\mathcal{G}^i$ remains constant there exists $i_0 > 0$ such that $\mathcal{G}^i = \mathcal{G}^{i_0}$ for all $i \geq i_0$.

Thus, we have the following algorithm for constructing A_*:

1. Given A_0, iterate $A_{i+1} = T_{\mathcal{D}}(A_i)$ until the fixed point A_{i_1} is reached.
2. Iterate $\mathcal{G}^{i+1} = T_{\widetilde{\mathcal{G}}_l^{i_1}[i_1/i_1-1]}(\mathcal{G}^i)$ to generate the fixed point $\mathcal{G}^{i_0}$ from $\mathcal{G}^{i_1}$.
3. Construct A_* by labelling the transitions of A_{i_1} with automata derived from $\mathcal{G}^{i_0}$.

Property 4. There exists an automaton A_* which is sound and complete with respect to $A_0, A_1, \ldots$ and hence computes the set $Pre^*(C_{Init})$.

3.5 The General Case

We may generalise our algorithm to order-n for all n by extending Definition 4 to l-store automata using similar techniques to those used in Definition 5. Termination is reached through a cascading of fixed points. As we fixed the state-set at order-1 in the order-2 case, we may fix the state-set at order-$(n-1)$ in the order-n case. We may then generalise Property 2 and Lemma 1 to find a sequence of fixed points $i_n, \ldots, i_0$, from which A_* can be constructed. For a complete description of this procedure, we refer the reader to the long version of this paper [13].

We claim our algorithm runs in n-EXPTIME. Intuitively, when the state-set $\mathcal{Q}$ is fixed at order-1 of the store automaton, we add at most $\mathcal{O}(2^{|\mathcal{Q}|})$ transitions (since we never remove states, it is this final stage that dominates the complexity). At orders $l > 1$ we add at most $\mathcal{O}(2^{|\mathcal{Q}|})$ new transitions, which exponentially increases the state-set at order-$(l-1)$. Hence, the algorithm runs in n-EXPTIME.

4 Applications

We give a brief description of a number of applications of our result:

– *Reachability Games.* Given an order-n pushdown reachability game with a regular set of goal configurations $\mathcal{R}$, we can calculate the winning region (which is regular) for the existential player in n-EXPTIME.
– *Linear-Time Model Checking.* Given an order-n PDS $(\mathcal{P}, \mathcal{D}, \Sigma)$ and a formula ϕ of an ω-regular logic, we can calculate in $(n+2)$-EXPTIME the set of configurations C such that every run from each $c \in C$ satisfies ϕ.
– *The Alternation-Free μ-Calculus.* Given an order-n PDS $(\mathcal{P}, \mathcal{D}, \Sigma)$ and a formula ϕ of the alternation-free μ-calculus, we can compute the regular set of configurations satisfying ϕ in $((|\phi| \cdot n) + 1)$-EXPTIME.

5 Conclusion

Given an automaton representation of a regular set of higher-order APDS configurations C_{Init}, we have shown that the set $Pre^*(C_{Init})$ is regular and computable via automata-theoretic methods. This builds upon previous work on pushdown systems [2] and higher-order context-free processes [1]. The main innovation of this generalisation is the careful management of a complex automaton construction. This allows us to identify a sequence of cascading fixed points, resulting in a terminating algorithm.

Our result has many applications. We have shown that it can be used to provide a solution to the model checking problem for linear-time temporal logics and the alternation-free μ-calculus. In particular we compute the set of configurations of a higher-order PDS satisfying a given constraint. We also show that the winning regions can be computed for a reachability game played over an higher-order PDS.

There are several possible extensions to this work. Firstly, we intend to complete the complexity analysis with corresponding hardness results. Although this result is widely accepted to follow from the work of Engelfriet [7], we intend to give an alternative proof in the long version of this paper. Secondly, we plan to investigate the applications of this work to higher-order pushdown games with more general winning conditions. In his PhD thesis, Cachat adapts the reachability algorithm of Bouajjani *et al.* [2] to calculate the winning regions in Büchi games over pushdown processes [22]. It is likely that our work will permit similar extensions. Finally, we intend to generalise this work to higher-order collapsible pushdown automata, which can be used to study higher-order recursion schemes [25, 8]. This may provide the first steps into the study of games over these structures.

Acknowledgments. We thank Olivier Serre and Arnaud Carayol for helpful discussions and the anonymous referees for their detailed comments.

References

1. A. Bouajjani and A. Meyer. Symbolic Reachability Analysis of Higher-Order Context-Free Processes. In *Proc. FSTTCS'04*, 2004. LNCS 3328.
2. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proc. CONCUR '97*, pp. 135–150, 1997.
3. A. Bouquet, O. Serre, and I. Walukiewicz. Pushdown games with the unboundedness and regular conditions. In *Proc. FSTTCS'03*, pages 88–99, 2003.
4. A. Carayol. Regular sets of higher-order pushdown stacks. In *Proc. MFCS*, pages 168–179, 2005.
5. A. Carayol and S. Wöhrle. The caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *Proc. FSTTCS*, pages 112–123, 2003.
6. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.
7. J. Engelfriet. Iterated push-down automata and complexity classes. In *Proc. STOC*, pages 365–373, 1983.

8. M. Hague, A. S. Murawski, O. Serre and C.-H. L. Ong. Collapsible pushdown automata and recursion schemes, 2006. Preprint, 13 pages.
9. C. Löding, P. Madhusudan, and O. Serre. Visibly pushdown games. In *Proc. FSTTCS'04*, pages 408–420. 2004. LNCS 3328.
10. D. Caucal. On infinite terms having a decidable monadic theory. In *Proc. MFCS'02*, pages 165–176, 2002. LNCS 2420.
11. D. E. Muller and P. E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theor. Comput. Sci.*, 37:51–75, 1985.
12. H. Gimbert. Parity and exploration games on infinite graphs. In *Proc. CSL'04*, pages 56–70, 2004. LNCS 3210.
13. M. Hague and C.-H. L. Ong. Symbolic backwards-reachability analysis for higher-order pushdown systems. Preprint, 54 pages, `www.comlab.ox.ac.uk/oucl/work/matthew.hague/FoSSaCS07-long.pdf`, 2006.
14. I. Walukiewicz. Pushdown processes: Games and model checking. In *Proc. CAV '96*, pages 62–74. 1996.
15. J. A. Brzozowski and E. L. Leiss. On equations for regular languages, finite automata, and sequential networks. *Theor. Comput. Sci.*, 10:19–35, 1980.
16. K. Aehlig, J. G. de Miranda, and C.-H. L. Ong. Safety is not a restriction at level 2 for string languages. In *Proc. FoSSaCS*, pages 490–504, 2005.
17. M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Banff Higher Order Workshop*, pages 238–266, 1995.
18. A. N. Maslov. Multilevel stack automata. *Problems of Information Transmission*, 15:1170–1174, 1976.
19. O. Serre. Note on winning positions on pushdown games with ω-regular conditions. *Information Processing Letters*, 85:285–291, 2003.
20. O. Serre. Games with winning conditions of high Borel complexity. In *Proc. ICALP'04*, pages 1150–1162. Springer-Verlag, 2004. LNCS 3142.
21. C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proc. LICS '06*, pages 81–90. IEEE Computer Society, 2006.
22. T. Cachat. *Games on Pushdown Graphs and Extensions*. PhD thesis, RWTH Aachen, 2003.
23. T. Cachat, J. Duparc, and W. Thomas. Solving pushdown games with a Σ_3 winning condition. In *Proc. CSL'02*, pages 322–336. Springer-Verlag, 2002. LNCS 2471.
24. T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-order pushdown trees are easy. In *Proc. FoSSaCS '02*, pages 205–222, London, UK, 2002. Springer-Verlag.
25. T. Knapik, D. Niwinski, P. Urzyczyn, and I. Walukiewicz. Unsafe grammars and panic automata. In *Proc. ICALP '05*, pages 1450–1461, 2005.
26. M. Y. Vardi. A temporal fixpoint calculus. In *Proc. POPL '88*, pages 250–259, New York, NY, USA, 1988. ACM Press.
27. W. Thomas. Automata on infinite objects. *Handbook of theoretical computer science (vol. B): formal models and semantics*, pages 133–191, 1990.