
SYMBOLIC BACKWARDS-REACHABILITY ANALYSIS FOR HIGHER-ORDER PUSHDOWN SYSTEMS

MATTHEW HAGUE AND LUKE ONG

Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, UK, OX1 3QD
e-mail address: {Matthew.Hague,Luke.Ong}@comlab.ox.ac.uk

ABSTRACT. Higher-order pushdown systems (PDSs) generalise pushdown systems through the use of higher-order stacks, that is, a nested “stack of stacks” structure. These systems may be used to model higher-order programs and are closely related to the Caucal hierarchy of infinite graphs and safe higher-order recursion schemes.

We consider the backwards-reachability problem over higher-order Alternating PDSs (APDSs), a generalisation of higher-order PDSs. This builds on and extends previous work on pushdown systems and context-free higher-order processes in a non-trivial manner. In particular, we show that the set of configurations from which a regular set of higher-order APDS configurations is reachable is regular and computable in n -EXPTIME. In fact, the problem is n -EXPTIME-complete.

We show that this work has several applications in the verification of higher-order PDSs, such as linear-time model-checking, alternation-free μ -calculus model-checking and the computation of winning regions of reachability games.

1. INTRODUCTION

1.1. Pushdown Automata and Pushdown Systems. Pushdown automata are an extension of finite state automata. In addition to a finite set of control states, a pushdown automaton has a stack which can be manipulated with the usual push and pop operations. Transitions of the automaton depend on both the current control state and the top item of the stack. During the execution of a transition, a push or pop operation is applied to the stack. Since there is no bound on the size of the stack, the resulting automaton has an infinite number of “states” or configurations, which consist of the current control state and the contents of the stack. This allows the definition of such non-regular languages as the well known $\{ a^n b^n \mid n \geq 0 \}$.

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. An order- n

1998 ACM Subject Classification: F.1.1.

Key words and phrases: Model-checking, pushdown systems, higher-order, reachability, symbolic techniques, automata, games.

PDA has push and pop commands for every $1 \leq l \leq n$. When $l > 1$ a pop command removes the topmost order- l stack. Conversely, the push command duplicates the topmost order- l stack.

Higher-order PDA were originally introduced by Maslov [19] 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.* [28] 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 graph hierarchy [8]. Remarkably these infinite trees and graphs have decidable monadic second-order (MSO) theories [9, 5, 28].

1.2. Backwards Reachability. The decidability results discussed above only allow us to check that a property holds from a given configuration. Alternatively, 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. An important step in solving this problem is the backwards reachability problem. That is, given a set of configurations C_{Init} , compute the set 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: many properties required in industry are safety properties — that is, an undesirable program state (such as deadlock) is never reached.

This problem was solved for order-one pushdown systems 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} . A regular set of configurations is represented 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. $Pre^*(C_{Init})$ is computed through the 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*.

This idea was generalised by Bouajjani and Meyer to the case of higher-order context-free systems [1], 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- n nested store automaton is a finite automaton whose transitions are labelled by order- $(n-1)$ nested store automata. In this way the structure of a higher-order store is reflected. The procedure is similar to the algorithm for the order-one case: transitions are added until a fixed point is reached. Termination in this case is more subtle. Since products are formed when processing higher-order push commands, the state space increases. However, it can be shown that only a finite number of products will be constructed and that termination follows.

Bouajjani and Meyer also show that forward reachability analysis does not result in regular sets of configurations.

1.3. Our Contribution. Our paper is concerned with the non-trivial problem¹ 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 [25], 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 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.

The algorithm uses a similar form of nested automata to represent configurations and uses a similar routine of adding transitions determined by the transition relation of the higher-order APDS. However, naïve combinations of the multi-automaton and nested-store automaton techniques do not lead to satisfactory solutions. During our own efforts with simple combined techniques, it was unclear how to form the product of two automata and maintain a distinction between the different control states as required. To perform such an operation safely it seemed that additional states were required on top of those added by the basic product operation, invalidating the termination arguments. We overcome this problem by using alternating automata and by modifying the termination argument. Additionally, we reduce the complexity of Bouajjani and Meyer from a tower of exponentials twice the size of n , to a tower of exponentials as large as n . In fact, the problem is n -EXPTIME-complete.

Termination is reached through a cascading of fixed points. Given a (nested) store-automaton, we fix the order- n state-set. During a number of iterations, we add a finitely bounded number of new transitions to order n of the automaton. We also update the automata labelling the previously added transitions to reflect the new transition structure. Eventually we reach a stage where no new transitions are being added at order n , although the automata labelling their edges will continue to be replaced. At this point the updates become repetitive and we are able to freeze the state-set at the second highest order. This is done by adding possibly cyclical transitions between the existing states, instead of chains of transitions between an infinite set of new states. Because the state-set does not change, we reach another fixed point similar to that at order n . In this way the fixed points cascade to order-one, where the finite alphabet ensures that the automaton eventually becomes saturated. We are left with an automaton representing the set $Pre^*(C_{Init})$.

1.4. Related Work. In this section we discuss several areas of related work. These are higher-order pushdown games, alternative notions of regularity, and higher-order recursion schemes.

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]

1.4.1. *Higher-Order Pushdown Games.* The definition of higher-order PDSs may be extended to higher-order pushdown games. In this scenario, control states are partitioned into to sets \exists and \forall . When the current configuration contains a control state in \exists , the player Eloise chooses the next configuration with respect to the transition relation. Conversely, Abelard chooses the next transition from a control state in \forall . The winner of the game depends on the winning condition. A configuration is winning for Eloise if she can satisfy the winning condition regardless of the choices made by Abelard. A winning region for Eloise is the set of all configurations from which Eloise can force a win. Two particular problems for these games are calculating whether a given configuration is winning for Eloise and computing the winning region for Eloise.

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 [12]. 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 [25]. Techniques for computing winning regions in the order-one case when the winning condition is a parity condition have been discovered independently by both Cachat [25] and Serre [20]. These results for pushdown games have been extended to a number of winning conditions [27, 3, 11, 21, 7]. In the higher-order case with a parity winning condition, a method for deciding whether a configuration is winning has been provided by Cachat [25].

1.4.2. *C-Regularity.* 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, when confusion may arise, we shall call *BM-regularity*.

A set of order- n stacks is C-regular if it is obtained by a regular sequence of order- n stack operations. This notion of regularity is not equivalent to BM-regularity. For example, the set of order-2 stacks defined by the expression $(push_a)^*; push_2$ are all stacks of the form $[[a^n][a^n]]$. This set is clearly unrecognisable by any finite state automaton, and thus, it is not BM-regular.

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 [25].

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, 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.

1.4.3. *Higher-Order Recursion Schemes.* Higher-order recursion schemes (HORSs) represent a further area of related work. A long standing open problem is whether a condition called *safety* is a genuine restriction on the expressiveness of a HORS. If not, then HORSs are equivalent to higher-order PDSs. It is known that safety is not a restriction at order-two for word languages [15]. This is conjectured not to be the case at higher orders.

MSO decidability for trees generated by arbitrary (i.e. not necessarily safe) HORSs has been shown by Ong [22]. A variant kind of higher-order PDSs called *collapsible pushdown automata* (extending *panic automata* [29] or *pushdown automata with links* [15] to all finite orders) has recently been shown to be equi-expressive with HORSs for generating ranked trees [17]. These new automata are conjectured to enrich the class of higher-order systems and provide many new avenues of research.

1.5. Document Structure. In Section 2 we give the definitions of higher-order (A)PDS and n -store multi-automata. We describe the backwards-reachability algorithm in the order-two case in three stages in Section 3: firstly we use an example to give an intuitive explanation of the algorithm. We then give a description of its framework and explain how we can generate an infinite sequence of 2-store multi-automata capturing the set $Pre^*(C_{Init})$. Finally, we show how this sequence can be finitely represented (and constructed). The section finishes with a brief discussion of the order- n case, and the complexity of the algorithm. Section 4 discusses the applications of the main result to LTL model-checking, reachability games and alternation-free μ -calculus model-checking over higher-order PDSs. Finally, we conclude in Section 5. Additional proofs and algorithms are given in the appendix.

2. PRELIMINARIES

2.1. Alternation. 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 [13] and Chandra, Kozen and Stockmeyer [6]. Similar definitions have been used for the analysis of pushdown systems by Bouajjani *et al.* [2] and Cachat [25]. The alternating transition relation $\Delta \subseteq \mathcal{Q} \times \Gamma \times 2^{\mathcal{Q}}$ — where Γ is an 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 $\{Q_1, \dots, Q_m\}$ with $Q_i \in 2^{\mathcal{Q}}$ for $i \in \{1, \dots, m\}$. 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 $q \in Q$. The existential component of the automaton is reflected in Eloise’s selection of an element (q, γ, Q) from Δ for a given q and γ . Abelard’s choice of a state q from Q represents the universal aspect of the automaton.

2.2. (Alternating) Higher-Order Pushdown Systems. A higher-order pushdown system comprises a finite set of control states and a higher-order store. Transitions of the higher-order PDS depend on both the current control state and the top symbol of the higher-order store. Each transition changes the control state and manipulates the store.

The main result of this paper is presented over *alternating* higher-order pushdown systems. This is because, although we apply our results to higher-order PDSs, the power of alternation is required to provide solutions to reachability games and alternation-free μ -calculus model-checking over higher-order PDSs.

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

Definition 2.1 (n -Stores). The set C_1^Σ of 1-stores over an alphabet Σ is the set of words of the form $[a_1, \dots, a_m]$ with $m \geq 0$ and $a_i \in \Sigma$ for all $i \in \{1, \dots, m\}$, $[\notin \Sigma \text{ and }] \notin \Sigma$. For $n > 1$, $C_n^\Sigma = [w_1, \dots, w_m]$ with $m \geq 1$ and $w_i \in C_{n-1}^\Sigma$ for all $i \in \{1, \dots, m\}$.

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 $w \in \Sigma^*$),

$$\begin{aligned} \text{push}_w[a_1 \dots a_m] &= [wa_2 \dots a_m] \\ \text{top}_1[a_1 \dots a_m] &= a_1 \end{aligned}$$

We may define the abbreviation $\text{pop}_1 = \text{push}_\varepsilon$. When $n > 1$, we have,

$$\begin{aligned} \text{push}_w[\gamma_1 \dots \gamma_m] &= [\text{push}_w(\gamma_1)\gamma_2 \dots \gamma_m] \\ \text{push}_l[\gamma_1 \dots \gamma_m] &= [\text{push}_l(\gamma_1)\gamma_2 \dots \gamma_m] \quad \text{if } 2 \leq l < n \\ \text{push}_n[\gamma_1 \dots \gamma_m] &= [\gamma_1\gamma_1\gamma_2 \dots \gamma_m] \\ \text{pop}_l[\gamma_1 \dots \gamma_m] &= [\text{pop}_l(\gamma_1)\gamma_2 \dots \gamma_m] \quad \text{if } 1 \leq l < n \\ \text{pop}_n[\gamma_1 \dots \gamma_m] &= [\gamma_2 \dots \gamma_m] \quad \text{if } m > 1 \\ \text{top}_l[\gamma_1 \dots \gamma_m] &= \text{top}_l(\gamma_1) \quad \text{if } 1 \leq l < n \\ \text{top}_n[\gamma_1 \dots \gamma_m] &= \gamma_1 \end{aligned}$$

Note that we assume without loss of generality $\Sigma \cap \mathcal{N} = \emptyset$, where \mathcal{N} is the set of natural numbers. Furthermore, observe that when $m = 1$, pop_n is undefined. We define $\mathcal{O}_n = \{ \text{push}_w \mid w \in \Sigma^* \} \cup \{ \text{push}_l, \text{pop}_l \mid 1 < l \leq n \}$. The definition of higher-order PDSs follows,

Definition 2.2. An *order- n PDS* is a tuple $(\mathcal{P}, \mathcal{D}, \Sigma)$ where \mathcal{P} is a finite set of control states p , $\mathcal{D} \subseteq \mathcal{P} \times \Sigma \times \mathcal{O}_n \times \mathcal{P}$ is a finite set of commands d , and Σ is a finite alphabet.

A configuration of a higher-order 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 $(p, a, o, p') \in \mathcal{D}$, $\text{top}_1(\gamma) = a$ and $\gamma' = o(\gamma)$.

We define $\xrightarrow{*}$ 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, for some configuration $\langle p', \gamma' \rangle \in C_{Init}$, we have $\langle p, \gamma \rangle \xrightarrow{*} \langle p', \gamma' \rangle$.

We may generalise this definition to the case of Alternating higher-order PDSs.

Definition 2.3. An *order- n APDS* is a tuple $(\mathcal{P}, \mathcal{D}, \Sigma)$ where \mathcal{P} is a finite set of control states p , $\mathcal{D} \subseteq \mathcal{P} \times \Sigma \times 2^{\mathcal{O}_n \times \mathcal{P}}$ is a finite set of commands d , and Σ is a finite alphabet.

A configuration of a higher-order 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 $(p, a, OP) \in \mathcal{D}$, $\text{top}_1(\gamma) = a$, and

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

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

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

We define $\xrightarrow{*}$ 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 we have $\langle p, \gamma \rangle \xrightarrow{*} C$ and $C \subseteq C_{Init}$.

Example 2.4. We present an example to illustrate the definition of $Pre^*(C_{Init})$ for higher-order APDSs. Figure 1 shows an excerpt of the configuration graph of a higher-order APDS

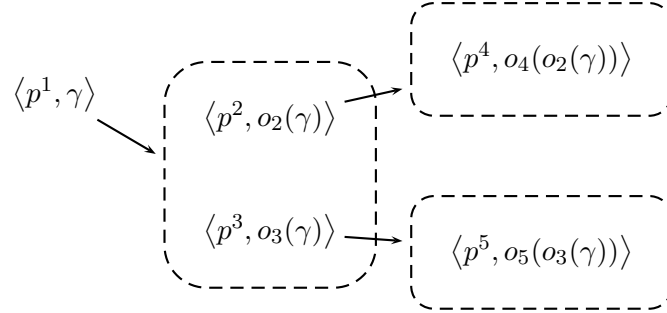


Figure 1: The configuration graph (excerpt) of an example higher-order APDS.

with the commands,

$$\begin{aligned} & (p^1, -, \{(o_2, p^2), (o_3, p^3)\}) \\ & (p^2, -, \{(o_4, p^4)\}) \\ & (p^3, -, \{(o_5, p^5)\}) \end{aligned}$$

We consider a number of different values of C_{Init} .

- (1) Let $C_{Init} = \{\langle p^2, o_2(\gamma) \rangle\}$. In this case $Pre^*(C_{Init}) = C_{Init}$. The configuration $\langle p^1, \gamma \rangle$ is not in $Pre^*(C_{Init})$ since the configuration $\langle p^3, o_3(\gamma) \rangle$ cannot be in $Pre^*(C_{Init})$.
- (2) Let $C_{Init} = \{\langle p^2, o_2(\gamma) \rangle, \langle p^3, o_3(\gamma) \rangle\}$. In this case $Pre^*(C_{Init}) = C_{Init} \cup \{\langle p^1, \gamma \rangle\}$. This is because the transition from $\langle p^1, \gamma \rangle$ reaches a set that is a subset of C_{Init} .
- (3) Let $C_{Init} = \{\langle p^4, o_4(o_2(\gamma)) \rangle\}$. In this case $Pre^*(C_{Init}) = C_{Init} \cup \{\langle p^2, o_2(\gamma) \rangle\}$. The configuration $\langle p^2, o_2(\gamma) \rangle$ is in the set because its transition moves to a set which is a subset of C_{Init} . The pair $\langle p^1, \gamma \rangle$ is not in the set because, although $\langle p^2, o_2(\gamma) \rangle$ is in $Pre^*(C_{Init})$, the configuration $\langle p^3, o_3(\gamma) \rangle$ is not.
- (4) Let $C_{Init} = \{\langle p^4, o_4(o_2(\gamma)) \rangle, \langle p^3, o_3(\gamma) \rangle\}$. In this case $Pre^*(C_{Init})$ is the set $C_{Init} \cup \{\langle p^2, o_2(\gamma) \rangle, \langle p^1, \gamma \rangle\}$. We have $\langle p^2, o_2(\gamma) \rangle \in Pre^*(C_{Init})$ as before. Furthermore, we have the following run from $\langle p^1, \gamma \rangle$,

$$\langle p^1, \gamma \rangle \hookrightarrow \{\langle p^2, o_2(\gamma) \rangle, \langle p^3, o_3(\gamma) \rangle\} \hookrightarrow \{\langle p^4, o_4(o_2(\gamma)) \rangle, \langle p^3, o_3(\gamma) \rangle\}$$

Hence, $\langle p^1, \gamma \rangle \in Pre^*(C_{Init})$.

Finally, suppose the higher-order APDS also has a command of the form,

$$(p^5, -, \{(push_l, p^4)\})$$

And it is the case that (only) $push_l(o_5(o_3(\gamma)))$ is undefined. If $C_{Init} = \{\langle p^5, \nabla \rangle\}$, then $Pre^*(C_{Init}) = C_{Init} \cup \{\langle p^5, o_5(o_3(\gamma)) \rangle, \langle p^3, o_3(\gamma) \rangle\}$.

Observe that since no transitions are possible from an “undefined” configuration $\langle p, \nabla \rangle$ 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].

2.3. n -Store Multi-Automata. To represent sets of configurations symbolically 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 *regular* iff it can be represented using an n -store multi-automaton. This notion of regularity coincides with the definition of Bouajjani and Meyer (see Appendix A). In Appendix B we give algorithms for enumerating runs of n -store automata, testing membership and performing boolean operations on the automata.

Definition 2.5.

- (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. It is the case that $\Delta \subseteq \mathcal{Q} \times \Sigma \times 2^{\mathcal{Q}}$ is a finite transition relation.
- (2) Let $\mathfrak{B}_{n-1}^{\Sigma}$ 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 $\Delta \subseteq \mathcal{Q} \times \mathfrak{B}_{n-1}^{\Sigma} \times 2^{\mathcal{Q}}$ is a *finite* transition relation. Furthermore, let $\mathfrak{B}_0^{\Sigma} = \Sigma$.
- (3) An n -store multi-automaton over the alphabet Σ is a tuple

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

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

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

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

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

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

A transition of the form $q^j \xrightarrow{\nabla} \{q_f^{\varepsilon}\}$ indicates that the undefined configuration $\langle p^j, \nabla \rangle$ is accepted. Runs of the automata from a state q take the form,

$$q \xrightarrow{\tilde{B}_0} \{q_1^1, \dots, q_{m_1}^1\} \xrightarrow{\tilde{B}_1} \dots \xrightarrow{\tilde{B}_m} \{q_1^{m+1}, \dots, q_{m_i}^{m+1}\}$$

where transitions between configurations $\{q_1^x, \dots, q_{m_x}^x\} \xrightarrow{\tilde{B}_x} \{q_1^{x+1}, \dots, q_{m_{x+1}}^{x+1}\}$ are such that we have $q_y^x \xrightarrow{B_y} Q_y$ for all $y \in \{1, \dots, m_x\}$ and $\bigcup_{y \in \{1, \dots, m_x\}} Q_y = \{q_1^{x+1}, \dots, q_{m_{x+1}}^{x+1}\}$ and additionally $\bigcup_{y \in \{1, \dots, m_x\}} \{B_y\} = \tilde{B}_x$. Observe that \tilde{B}_0 is necessarily a singleton set. A run over a word $\gamma_1 \dots \gamma_m$, denoted $q \xrightarrow{\gamma_1 \dots \gamma_m} Q$, exists whenever,

$$q \xrightarrow{\tilde{B}_0} \dots \xrightarrow{\tilde{B}_m} Q$$

and for all $0 \leq i \leq m$, $\gamma_i \in \mathcal{L}(\tilde{B}_i)$, where $\gamma \in \mathcal{L}(\tilde{B})$ iff $\gamma \in \mathcal{L}(B)$ (defined below) for all $B \in \tilde{B}$. If a run occurs in an automaton forming part of a sequence of automata A_0, A_1, \dots , we may write \rightarrow_i to indicate which automaton A_i the run belongs to.

We define $\mathcal{L}(a) = a$ for all $a \in \Sigma = \mathfrak{B}_0^{\Sigma}$. An n -store $[\gamma_1 \dots \gamma_m]$ is accepted by an n -store automaton A (that is $[\gamma_1 \dots \gamma_m] \in \mathcal{L}(A)$) iff we have a run $q_0 \xrightarrow{\gamma_1 \dots \gamma_m} Q$ in A with $Q \subseteq \mathcal{Q}_f$.

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

$$\begin{aligned} \mathcal{L}(A^{q^j}) &= \{ [\gamma_1 \dots \gamma_m] \mid q^j \xrightarrow{\gamma_1 \dots \gamma_m} Q \wedge Q \subseteq \mathcal{Q}_f \} \\ &\cup \{ \nabla \mid q^j \xrightarrow{\nabla} \{q_f^\varepsilon\} \} \end{aligned}$$

and

$$\mathcal{L}(A) = \{ \langle p^j, \gamma \rangle \mid j \in \{1, \dots, z\} \wedge \gamma \in \mathcal{L}(A^{q^j}) \}$$

Finally, we define the automata B_l^a for all $1 \leq l \leq n$ and $a \in \Sigma$ and the notation q^θ . The l -store automaton B_l^a accepts any l -store γ such that $\text{top}_1(\gamma) = a$. If θ represents a store automaton, the state q^θ refers to the initial state of the automaton represented by θ .

3. BACKWARDS REACHABILITY: THE ORDER-TWO CASE

Since the backwards reachability problem for higher-order PDSs permits a direct reduction to the same problem for higher-order APDSs, we solve the backwards reachability problem for higher-order APDSs. Due to space constraints we present the order-2 case. The general case is addressed briefly at the end of this section and is due to appear in Hague's Ph.D. thesis [16].

Theorem 3.1. *Given an 2-store multi-automaton A_0 accepting the set of configurations C_{Init} of an order-2 APDS, we can construct in 2-EXPTIME (in the size of A_0) an 2-store multi-automaton A_* accepting the set $\text{Pre}^*(C_{Init})$. Thus, $\text{Pre}^*(C_{Init})$ is regular. \square*

Fix an order-2 APDS. We begin by showing how to generate an infinite sequence of automata A_0, A_1, \dots , 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 $\text{Pre}^*(C_{Init})$; that is $\bigcup_{i \geq 0} \mathcal{L}(A_i) = \text{Pre}^*(C_{Init})$. 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, without loss of generality, that all initial states in A_0 have no incoming transitions and there exists 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 following two-state order-two PDS:

$$\begin{aligned} d_1 &= (p^1, a, \text{push}_2, p^1) \\ d_2 &= (p^1, a, \text{push}_\varepsilon, p^1) \\ d_3 &= (p^2, a, \text{push}_w, p^1) \\ d_4 &= (p^2, a, \text{pop}_2, p^1) \end{aligned}$$

And a 2-store multi-automaton A_0 shown in Figure 2 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, \dots . We construct A_{i+1} from A_i to reflect an additional inverse application of the commands d_1, \dots, d_4 . Rather than manipulating the 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 $\tilde{G}_{(q_1, q_2)}^1$. This set is a recipe for the construction of an order-1 store automaton that will ultimately label the edge. The set \mathcal{G}^1 is the set of all sets $\tilde{G}_{(q_1, q_2)}^1$ introduced. The resulting A_1 is given in Figure 3 where the contents of

$$\mathcal{G}^1 = \{ \tilde{G}_{(q^1, \circ)}^1, \tilde{G}_{(q^1, q_f)}^1, \tilde{G}_{(q^2, \circ)}^1, \tilde{G}_{(q^2, q^1)}^1 \}$$

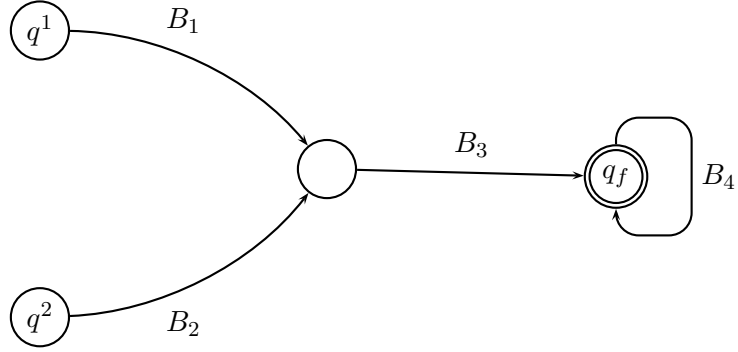
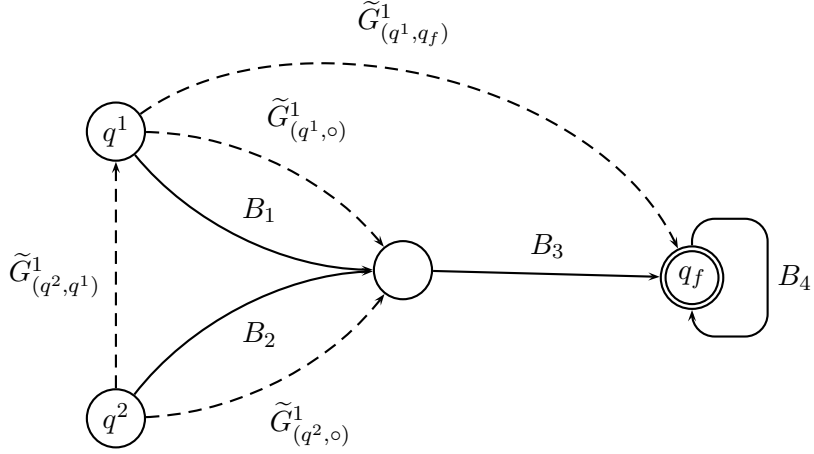


Figure 2: The initial 2-store multi-automaton

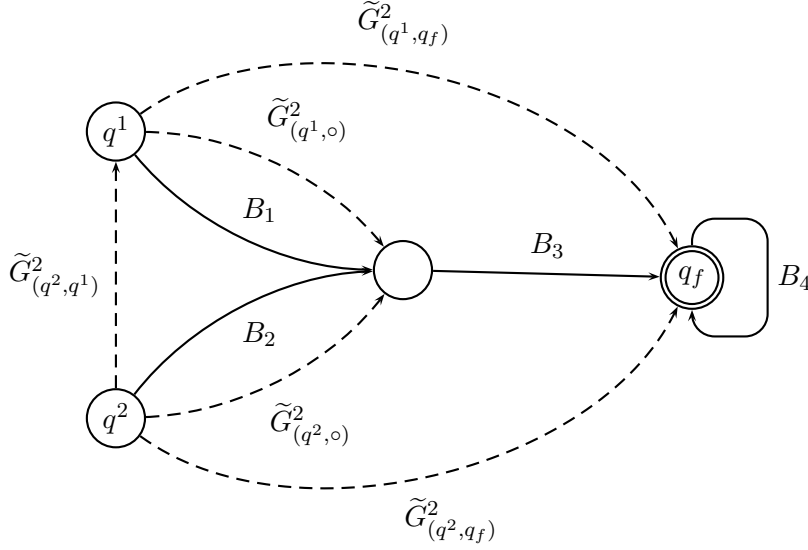
Figure 3: The automaton A_1

	d_1	d_2	d_3	d_4
$\tilde{G}_{(q^1, o)}^1$		$\{(a, push_\varepsilon, B_1)\}$		
$\tilde{G}_{(q^1, q_f)}^1$	$\{B_1^a, B_1, B_3\}$			
$\tilde{G}_{(q^2, o)}^1$			$\{(a, push_w, B_1)\}$	
$\tilde{G}_{(q^2, q^1)}^1$				$\{B_1^a\}$

Table 1: The contents of the sets in \tilde{G}^1 .

are given in Table 1. The columns indicate which command introduced each element to the set.

To process the command d_1 we need to add to the set of configurations accepted by A_1 all configurations of the form $\langle p_1, [\gamma_1 \dots \gamma_m] \rangle$ with $top_1(\gamma_1) = a$ for each configuration


 Figure 4: The automaton A_2 .

$\langle p_1, [\gamma_1 \gamma_1 \dots \gamma_m] \rangle$ accepted by A_0 . This is because $push_2[\gamma_1 \dots \gamma_m] = [\gamma_1 \gamma_1 \dots \gamma_m]$. Hence we add the transition from q^1 to q_f . The contents of $\tilde{G}_{(q^1, q_f)}^1$ indicate that this edge must accept the product of B_1^a , 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 $\tilde{G}_{(q^1, o)}^1$ and $\tilde{G}_{(q^2, o)}^1$ specify that the automaton B_1 must be manipulated to produce the automaton that will label these new transitions. Finally, since $pop_2[\gamma_1 \dots \gamma_m] = [\gamma_2 \dots \gamma_m]$, 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 A_2 is given in Figure 4 where the contents of

$$\mathcal{G}^2 = \{\tilde{G}_{(q^1, o)}^2, \tilde{G}_{(q^1, q_f)}^2, \tilde{G}_{(q^2, o)}^2, \tilde{G}_{(q^2, q^1)}^2, \tilde{G}_{(q^2, q_f)}^2\}$$

are given in Table 2. The columns indicate which command introduced each element to the set.

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 $\tilde{G}_{(q, q')}^3$ will match $\tilde{G}_{(q, q')}^2$ in everything but the indices of the labels $\tilde{G}_{(-, -)}^1$ appearing in the element sets. We may write $\tilde{G}_{(q, q')}^3 = \tilde{G}_{(q, q')}^2[2/1]$ where the notation $[2/1]$ indicates a substitution of the element indices.

So far we have just constructed sets to label the transitions of A_1 and A_2 . To complete the construction of A_1 we need to construct the automata $G_{(q, q')}^1$ represented by the labels $\tilde{G}_{(q, q')}^1$ for the appropriate q, q' . Because each of these new automata will be constructed from B_1, \dots, B_4, B_1^a , we build them simultaneously, constructing a single (1-store multi-)automaton \mathcal{G}^1 with an initial state $g_{(q, q')}^1$ for each $\tilde{G}_{(q, q')}^1$. The automaton \mathcal{G}^1 is constructed

	d_1	d_2	d_3	d_4
$\tilde{G}_{(q^1, \circ)}^2$		$\{(a, push_\varepsilon, B_1)\}$ $\{(a, push_\varepsilon, \tilde{G}_{(q^1, \circ)}^1)\}$		
$\tilde{G}_{(q^1, q_f)}^2$	$\{B_1^a, B_1, B_3\}$ $\{B_1^a, \tilde{G}_{(q^1, q_f)}^1, B_4\}$ $\{B_1^a, \tilde{G}_{(q^1, \circ)}^1, B_3\}$	$\{(a, push_\varepsilon, \tilde{G}_{(q^1, q_f)}^1)\}$		
$\tilde{G}_{(q^2, \circ)}^2$			$\{(a, push_w, B_1)\}$ $\{(a, push_w, \tilde{G}_{(q^1, \circ)}^1)\}$	
$\tilde{G}_{(q^2, q^1)}^2$				$\{B_1^a\}$
$\tilde{G}_{(q^2, q_f)}^2$			$\{(a, push_w, \tilde{G}_{(q^1, q_f)}^1)\}$	

Table 2: The contents of the sets in $\tilde{\mathcal{G}}^2$.

through the addition of states and transitions to the disjoint union of B_1, \dots, B_4, B_1^a . Creating the automaton A_2 is analogous and \mathcal{G}^2 is built through the addition of states and transitions to \mathcal{G}^1 .

The automaton \mathcal{G}^1 is given in Figure 5. We do not display this automaton in full since the number of alternating transitions entails a diagram too complicated to be illuminating. Instead we will give the basic structure of the automaton with many transitions omitted. In particular we show a transition derived from $\{B_1^a, B_1, B_3\}$ (from state $g_{(q^1, q_f)}^1$), a transition derived from $\{(a, push_\varepsilon, B_1)\}$ (from state $g_{(q^1, \circ)}^1$) and a transition derived from $\{B_1^a\}$ (from state $g_{(q^2, q^1)}^1$). Notably, we have omitted any transitions derived from the $push_w$ command. This is simply for convenience since we do not wish to further explicate B_1, B_2, B_3 or B_4 . From this automaton we derive $G_{(q^1, \circ)}^1, G_{(q^1, q_f)}^1, G_{(q^2, \circ)}^1$ and $G_{(q^2, q^1)}^1$ by setting the initial state to $g_{(q^1, \circ)}^1, g_{(q^1, q_f)}^1, g_{(q^2, \circ)}^1$ and $g_{(q^2, q^1)}^1$ respectively.

The automaton \mathcal{G}^2 is shown in Figure 6. Again, due to the illegibility of a complete diagram, we omit many of the transitions. The new transition from $g_{(q^1, q_f)}^2$ is derived from the set $\{B_1^a, B_3, \tilde{G}_{(q^1, \circ)}^1\}$. One of the transitions from $g_{(q^1, \circ)}^2$ and the only transition from $g_{(q^2, q^1)}^2$ are inherited from their corresponding states in the previous automaton. This inheritance ensures that we do not lose information from the previous iteration. The uppermost transition from $g_{(q^1, \circ)}^2$ derives from $\{(a, push_\varepsilon, \tilde{G}_{(q^1, \circ)}^1)\}$. From this automaton we derive $G_{(q^1, \circ)}^1, G_{(q^1, q_f)}^1, G_{(q^2, \circ)}^1$ and $G_{(q^2, q^1)}^1$.

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

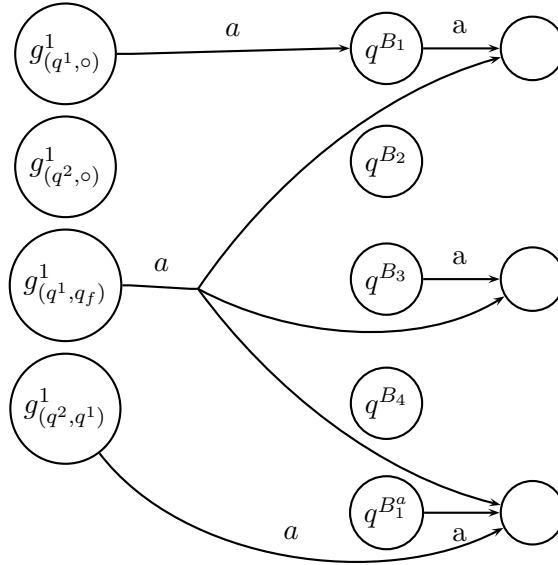


Figure 5: A selective view of \mathcal{G}^1 .

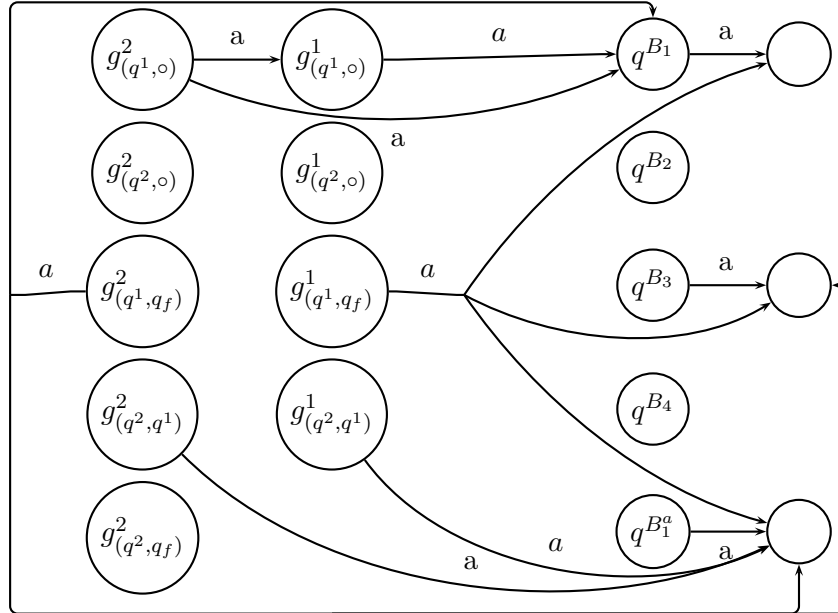


Figure 6: A selective view of \mathcal{G}^2 .

To construct A_* such that $\mathcal{L}(A_*) = \bigcup_{i \geq 0} \mathcal{L}(A_i)$ 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 7.

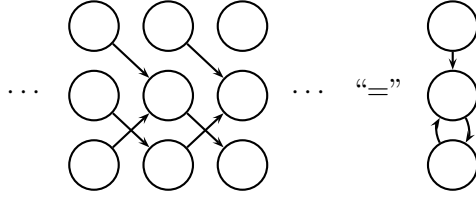
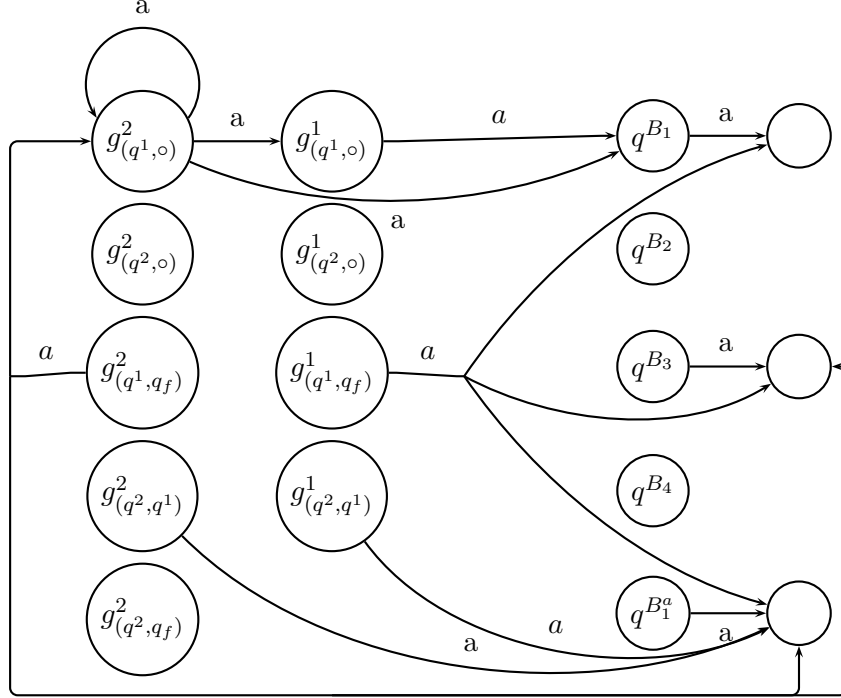


Figure 7: Collapsing a repetitive chain of new states.

Figure 8: A selective view of $\hat{\mathcal{G}}^*$.

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 $\tilde{G}_{(q, q')}^2[2/1]$ 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 $\hat{\mathcal{G}}^*$. This automaton will be sound and complete with respect to $Pre^*(\mathcal{L}(A_0))$.

An abbreviated diagram of $\hat{\mathcal{G}}^*$ is given in Figure 8. We have hidden, for clarity, the transition derived from $\{B_l^a, B_3, \tilde{G}_{(q^1, o)}^1\}$ in Figure 6. Instead, we show the transition introduced for the set $\{B_1^a, B_3, \tilde{G}_{(q^1, o)}^1\}[2/1] = \{B_1^a, B_3, \tilde{G}_{(q^1, o)}^2\}$ during the construction of $\hat{\mathcal{G}}^*$. We have also added the self-loop added by $\{(a, push_\varepsilon, G_{(q^1, o)}^1)\}[2/1] = \{(a, push_\varepsilon, G_{(q^1, o)}^2)\}$ that enabled the introduction of this transition.

3.2. Preliminaries. We now discuss the algorithm more formally. We begin by describing the transitions labelled by $G_{(q_1, Q_2)}^i$ before discussing the construction of the sequence A_0, A_1, \dots and the automaton A^* .

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. 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 $\tilde{G}_{(q_1, Q_2)}$. To describe our algorithm we will define sequences of automata, indexed by i . We superscript the identifier to indicate to which automaton in the sequence it belongs. The identifier $\tilde{G}_{(q_1, Q_2)}^i$ is associated with a set that acts as a recipe for updating the 1-store automaton described by $\tilde{G}_{(q_1, Q_2)}^{i-1}$ or creating a new automaton if $\tilde{G}_{(q_1, Q_2)}^{i-1}$ does not exist. Ultimately, the constructed 1-store automaton will label the new transition. In the sequel, we will confuse the notion of an identifier and its associated set. The intended usage should be clear from the context.

The sets are in a kind of disjunctive normal form. A set $\{S_1, \dots, S_m\}$ represents an automaton that accepts the union of the languages accepted by the automata described by S_1, \dots, S_m . Each set $S \in \{S_1, \dots, S_m\}$ corresponds to a possible effect of a command d at order-1 of the automaton. The automaton described by $S = \{\alpha_1, \dots, \alpha_m\}$ accepts the intersection of languages described by its elements α_t ($t \in \{1, \dots, m\}$). An element that is an automaton B refers directly to the automaton B . Similarly, an identifier $\tilde{G}_{(q_1, Q_2)}^i$ 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 the 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 $\tilde{G}_{(q_1, Q_2)}^i$ we attach a subset of

$${}_2\mathcal{B} \cup \tilde{\mathcal{G}}^{i-1} \cup (\Sigma \times \mathcal{O}_1 \times (\mathcal{B} \cup \tilde{\mathcal{G}}^{i-1}))$$

where \mathcal{B} is the set of all 1-store automata occurring in A_0 and all automata of the form B_1^a . Further, we denote the set of all identifiers $\tilde{G}_{(q, Q)}^i$ in A_i as $\tilde{\mathcal{G}}^i$. The sets \mathcal{B} and \mathcal{O}_1 are finite by definition. The size of the set $\tilde{\mathcal{G}}^i$ for any i is finitely bound by the (fixed) state-set of A_i .

We build the automata for all $\tilde{G}_{(q_1, Q_2)}^i \in \tilde{\mathcal{G}}^i$ simultaneously. That is, we create a single automaton \mathcal{G}^i associated with the set $\tilde{\mathcal{G}}^i$. This automaton has a state $g_{(q_1, Q_2)}^i$ for each $\tilde{G}_{(q_1, Q_2)}^i \in \tilde{\mathcal{G}}^i$. The automaton $G_{(q_1, Q_2)}^i$ labelling the transition $q_1 \rightarrow_i Q_2$ is the automaton \mathcal{G}^i with $g_{(q_1, Q_2)}^i$ 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 $\mathcal{G}^{i+1} = T_{\tilde{\mathcal{G}}^{i+1}}(\mathcal{G}^i)$ where $T_{\tilde{\mathcal{G}}^j}(\mathcal{G}^i)$ is given in Definition 3.2. In Section 3.4 it will be seen that j is not always $(i + 1)$.

Definition 3.2. Given an automaton $\mathcal{G}^i = (\mathcal{Q}^i, \Sigma, \Delta^i, -, \mathcal{Q}_f)$ and a set of identifiers (and associated sets) $\tilde{\mathcal{G}}_1^j$, we define,

$$\mathcal{G}^{i+1} = T_{\tilde{\mathcal{G}}_1^j}(\mathcal{G}^i) = (\mathcal{Q}^{i+1}, \Sigma, \Delta^{i+1}, -, \mathcal{Q}_f)$$

where $\mathcal{Q}^{i+1} = \mathcal{Q}^i \cup \{ g_{(q_1, Q_2)}^j \mid \tilde{G}_{(q_1, Q_2)}^j \in \tilde{\mathcal{G}}^j \}$, $\Delta^{i+1} = \Delta^{inherited} \cup \Delta^{new} \cup \Delta^i$, and,

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

where (1) requires $\{\alpha_1, \dots, \alpha_r\} \in \tilde{\mathcal{G}}_{(q_1, Q_2)}^j$, $Q = Q_1 \cup \dots \cup Q_r$ and for each $t \in \{1, \dots, r\}$ 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 .

There are two key parts to Definition 3.2. During the first stage we add a new initial state for each automaton forming a part of \mathcal{G}^{i+1} . By adding new initial states, rather than using the previous set of initial states, we guarantee that no unwanted cycles are introduced, which may lead to the erroneous acceptance of certain stores. We ensure that each 1-store accepted by \mathcal{G}^i is accepted by \mathcal{G}^{i+1} — and the set of accepted stores is increasing — by inheriting transitions from the previous set of initial states.

During the second stage we add transitions between the set of new initial states and the state-set of \mathcal{G}^i to capture the effect of a backwards application of the APDS commands to $\mathcal{L}(A_i)$. Intuitively, we only add new transitions to the initial states because all stack operations affect the top of the stack, leaving the remainder unchanged.

There are two different forms for the elements $\alpha_t \in \{\alpha_1, \dots, \alpha_r\}$. If α_t refers directly to an automaton, then we require that the new store is also accepted by the automaton referred to by α_t . We simply inherit the initial transitions of that automaton in a similar manner to the first stage of $T_{\tilde{\mathcal{G}}^j}(\mathcal{G}^i)$. If α_t is of the form $(a, push_w, \theta)$, then it corresponds to the effects of a command $(p, a, \{\dots, (push_w, p'), \dots\})$. The new store must have the character a as its top_1 character, and the store resulting from the application of the operation $push_w$ must be accepted by the automaton represented by θ . That is, the new state must accept all stores of the form aw' when the store ww' is accepted by θ .

3.3. Constructing the Sequence A_0, A_1, \dots For a given order-2 APDS with commands \mathcal{D} we define $A_{i+1} = T_{\mathcal{D}}(A_i)$ where the operation $T_{\mathcal{D}}$ follows. We assume A_0 has a state q_f^ε with no outgoing transitions and a state q_f^* from which all stores are accepted.

Definition 3.3. Given an automaton $A_i = (\mathcal{Q}, \Sigma, \Delta^i, \{q^1, \dots, q^z\}, \mathcal{Q}_f)$ and a set of commands \mathcal{D} , we define,

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

where Δ^{i+1} is given below.

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

$$\tilde{\mathcal{G}}^{i+1} = \left\{ \tilde{G}_{(q^j, Q)}^{i+1} \mid (q^j \xrightarrow{\tilde{G}_{(q^j, Q)}^i} Q) \in \Delta^i \text{ and } j \in \{1, \dots, z\} \right\} \cup \left\{ \tilde{G}_{(q^j, Q)}^{i+1} \mid (2) \right\}$$

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

- If $o_t = push_2$, then $S_t = \{B_1^a\} \cup \tilde{\theta}_1 \cup \tilde{\theta}_2$ and there exists a path $q^{k_t} \xrightarrow{\tilde{\theta}_1}_i Q' \xrightarrow{\tilde{\theta}_2}_i Q_t$ in A_i .
- If $o_t = pop_2$, then $S_t = \{B_1^a\}$ and $Q_t = \{q^{k_t}\}$. Or, if $q^j \xrightarrow{\nabla}_i \{q_f^\varepsilon\}$ exists in A_i , we may have $S_t = \{B_1^a\}$ and $Q_t = \{q_f^\varepsilon\}$.
- 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} .

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

We can construct an automaton whose transitions are 1-store automata by replacing each set $\tilde{G}_{(q,Q)}^{i+1}$ with the automaton $G_{(q,Q)}^{i+1}$ which is \mathcal{G}^{i+1} with initial state $g_{(q,Q)}^{i+1}$, where $\mathcal{G}^{i+1} = T_{\tilde{\mathcal{G}}^{i+1}}(\mathcal{G}^i)$. Note that \mathcal{G}^i is assumed by induction. In the base case, \mathcal{G}^0 is the disjoint union of all automata in \mathcal{B} .

The above construction is similar to Definition 3.2. However, because we do not change the initial states of the automaton, we do not have to perform the inheritance step. Furthermore the set of commands \mathcal{D} specify how the automata should be updated, rather than a set $\tilde{\mathcal{G}}^i$. A command $(p^j, a, \{(o_1, p^{k_1}), \dots, (o_m, p^{k_m})\})$ takes the place of a set $\{\alpha_1, \dots, \alpha_m\}$.

The contents of S_t and Q_t depend on the operation o_t . If o_t is of a lower order than 2 (that is, a $push_w$ command) then $o_t(\gamma w) = o_t(\gamma)w$ for any store γw . Hence we inherit the first transition from the initial state of the automaton represented θ , but pass the required constraint (using $S_t = \{(a, o_t, B)\}$) to the lower orders of the automaton.

Otherwise o_t is a pop_2 or $push_2$ operation. If is a $push_2$ command, then $push_2(\gamma w') = \gamma \gamma w'$, and hence we use S_t to ensure that the top store γ of $\gamma w'$ is accepted by the first two transitions from the initial state of the automaton represented by θ and we use Q_t to ensure that the tails of the stores match.

In case o_t is a pop_2 operation and the new store is simply the old store with an additional 2-store on top (that is $pop_2(\gamma w') = w'$). Thus, Q_t is the initial state of the automaton represented by θ and S_t contains the automaton B_1^a , which ensures that the top_1 character of the new store is a . We also need to consider the undefined store ∇ . This affects the processing of pop_2 operations since their result is not always defined. Hence, when considering which new stores may be accepted by A_{i+1} , we check whether the required undefined configuration is accepted by A_i . This is witnessed by the presence of a ∇ transition from p^j . If the result may be undefined, we accept all stores that do not have an image under the pop_2 operation. That is, all stores of the form $[\gamma]$.

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

Property 3.4. *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 $\langle p^j, \gamma \rangle \in Pre^*(C_{Init})$.*

Proof. From Property C.8 and Property C.9. □

3.4. Constructing the Automaton A_* . We need to construct a finite representation of the sequence A_0, A_1, \dots 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 $\tilde{G}_{(q_1, Q_2)}^i$.

Definition 3.5.

(1) Given $\theta \in \mathcal{B} \cup \tilde{\mathcal{G}}^{i'}$ for some i' , let

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

(2) For a set S we define $S[j/i]$ such that,

(a) We have $\theta \in S$ iff we have $\theta[j/i] \in S[j/i]$, and

(b) We have $(a, o, \theta) \in S$ iff we have $(a, o, \theta[j/i]) \in S[j/i]$.

(3) We extend the notation $[j/i]$ to nested sets of sets structures in a point-wise fashion.

Definition 3.6.

(1) We write $\tilde{G}_{(q_1, Q_2)}^i \lesssim \tilde{G}_{(q_1, Q_2)}^j$ iff for each $S \in \tilde{G}_{(q_1, Q_2)}^i$ we have $S[j-1/i-1] \in \tilde{G}_{(q_1, Q_2)}^j$.

(2) If $\tilde{G}_{(q_1, Q_2)}^i \lesssim \tilde{G}_{(q_1, Q_2)}^j$ and $\tilde{G}_{(q_1, Q_2)}^j \lesssim \tilde{G}_{(q_1, Q_2)}^i$, then we write $\tilde{G}_{(q_1, Q_2)}^i \simeq \tilde{G}_{(q_1, Q_2)}^j$.

(3) Furthermore, we extend the notation to sets. That is, $\tilde{\mathcal{G}}_l^i \lesssim \tilde{\mathcal{G}}_l^j$ iff for all $\tilde{G}_{(q_1, Q_2)}^i \in \tilde{\mathcal{G}}_l^i$ we have $\tilde{G}_{(q_1, Q_2)}^j \in \tilde{\mathcal{G}}_l^j$ and $\tilde{G}_{(q_1, Q_2)}^i \lesssim \tilde{G}_{(q_1, Q_2)}^j$.

We now show that a fixed point is reached at order-2. That we reach a fixed point is important, since, when $\tilde{\mathcal{G}}^i \simeq \tilde{\mathcal{G}}^{i+1}$ there are two key consequences. Firstly, for all q_1 and Q_2 , we have $\tilde{G}_{(q_1, Q_2)}^i \in \tilde{\mathcal{G}}^i$ iff we also have $\tilde{G}_{(q_1, Q_2)}^{i+1} \in \tilde{\mathcal{G}}^{i+1}$. This means that, if we ignore the automata labelling the edges of A_i and A_{i+1} , the two automata have the same transition structure. The second consequence follows from the first: we have $\tilde{G}_{(q_1, Q_2)}^i \simeq \tilde{G}_{(q_1, Q_2)}^{i+1}$ for all q_1 and Q_2 . That is, the automata labelling the edges of A_i and A_{i+1} will be updated in the same manner. It is this repetition that allows us to fix the state-set at order-1, and thus reach a final fixed point.

Property 3.7. *There exists $i_1 > 0$ such that $\tilde{\mathcal{G}}^i \simeq \tilde{\mathcal{G}}^{i_1}$ 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 $\tilde{\mathcal{G}}^i \simeq \tilde{\mathcal{G}}^{i_1}$ for all $i \geq i_1$ follows since the contents of $\tilde{G}_{(q_1, Q_2)}^i$ and $\tilde{G}_{(q_1, Q_2)}^{i_1}$ are derived from the same transition structure. \square

Once a fixed point has been reached at order-2, we can fix the state-set at order-1.

Lemma 3.8. *Suppose we have constructed, as above, a sequence of automata $\mathcal{G}^0, \mathcal{G}^1, \dots$ with the associated sets $\tilde{\mathcal{G}}^0, \tilde{\mathcal{G}}^1, \dots$. Further, suppose there exists an i_1 such that for all $i \geq i_1$ we have $\tilde{\mathcal{G}}^i \simeq \tilde{\mathcal{G}}^{i_1}$. We can define a sequence of automata $\hat{\mathcal{G}}^{i_1}, \hat{\mathcal{G}}^{i_1+1}, \dots$ such that the state-set in $\hat{\mathcal{G}}^i$ remains constant and there exists i_0 such that $\hat{\mathcal{G}}^{i_0}$ characterises the sequence — that is, the following are equivalent for all w ,*

- (1) *The run $g_{(q, Q)}^{i_1} \xrightarrow{w}_i Q_1$ with $Q_1 \subseteq \mathcal{Q}_f$ exists in $\hat{\mathcal{G}}^i$ for some i .*
- (2) *The run $g_{(q, Q)}^{i_1} \xrightarrow{w}_{i_0} Q_2$ with $Q_2 \subseteq \mathcal{Q}_f$ exists in $\hat{\mathcal{G}}^{i_0}$.*
- (3) *The run $g_{(q, Q)}^{i'} \xrightarrow{w}_{i'} Q_3$ with $Q_3 \subseteq \mathcal{Q}_f$ exists in $\mathcal{G}^{i'}$ for some i' .*

Proof. Follows from the definition of $\hat{\mathcal{G}}^{i+1} = T_{\tilde{\mathcal{G}}^{i_1}[i_1/i_1-1]}(\hat{\mathcal{G}}_1^i)$, Lemma D.2, Lemma D.3 and Lemma D.4. \square

We use $\hat{\mathcal{G}}^{i+1} = T_{\tilde{\mathcal{G}}^{i_1}[i_1/i_1-1]}(\hat{\mathcal{G}}^i)$ to construct the sequence $\hat{\mathcal{G}}^{i_1}, \hat{\mathcal{G}}^{i_1+1}, \dots$ with $\hat{\mathcal{G}}^{i_1} = \mathcal{G}^{i_1}$. 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. The substitution $\tilde{\mathcal{G}}^{i_1}[i_1/i_1-1]$ makes the sets in $\tilde{\mathcal{G}}^{i_1}$ self-referential. This generates the loops shown in Figure 7. Since the state-set of this new sequence does not change and the alphabet Σ is finite, the transition structure will become saturated.

We define $\hat{\mathcal{G}}^* = \hat{\mathcal{G}}^{i_0}$ letting $g_{(q_1, Q_2)}^* = g_{(q_1, Q_2)}^{i_1}$ for each $g_{(q_1, Q_2)}^{i_1}$. Finally, we show that we can construct the automaton A_* .

Property 3.9. *There exists an automaton A_* which is sound and complete with respect to A_0, A_1, \dots and hence computes the set $Pre^*(C_{Init})$.*

Proof. By Property 3.7 there is some i_1 with $\tilde{\mathcal{G}}^i \simeq \tilde{\mathcal{G}}^{i_1}$ for all $i \geq i_1$. By Lemma 3.8, we have $\hat{\mathcal{G}}^* = \hat{\mathcal{G}}^{i_0}$. We then define A_* from A_{i_1} with each transition $q \rightarrow_* Q'$ in A_* labelled with the automaton $G_{(q, Q')}^*$ from $\hat{\mathcal{G}}^* = \hat{\mathcal{G}}^{i_0}$. \square

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 $\hat{\mathcal{G}}^{i+1} = T_{\tilde{\mathcal{G}}^{i_1}[i_1/i_1-1]}(\hat{\mathcal{G}}^i)$ to generate the fixed point $\hat{\mathcal{G}}^*$ from \mathcal{G}^{i_1} .
- (3) Construct A_* by labelling the transitions of A_{i_1} with automata derived from $\hat{\mathcal{G}}^*$.

3.5. The General Case and Complexity. We may generalise our algorithm to order- n for all n by extending Definition 3.2 to n -store automata using similar techniques to those used in Definition 3.3. 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 3.7 and Lemma 3.8 to find a sequence of fixed points i_n, \dots, i_0 , from which A_* can be constructed. For a complete description of this procedure, we refer the reader to Hague's forthcoming Ph.D. thesis [16].

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. This algorithm is optimal since reachability games over higher-order PDSs are n -EXPTIME-complete [26]. An alternative proof of n -EXPTIME-hardness — by reduction from the non-emptiness of order- $(n+1)$ PDA — is due to appear in Hague's Ph.D. thesis [16]. It was shown by Engelfriet that the non-emptiness problem for order- $(n+1)$ PDSs is n -EXPTIME-complete [10].

When the higher-order PDS is nondeterministic (rather than alternating), we add at most $|\mathcal{Q}|^2$ transitions at order- n . Hence, the complexity is $(n-1)$ -EXPTIME, matching the lower-bound of the non-emptiness problem for higher-order PDA (as acceptors of word languages).

4. APPLICATIONS

In this section we discuss some of the applications of our algorithm to decision problems over higher-order PDSs.

4.1. Model-Checking Linear-Time Temporal Logics. Bouajjani *et al.* use their backwards reachability algorithm to provide a model-checking algorithm for linear-time temporal logics over the configuration graphs of pushdown systems [2]. In this section we show that this work permits a simple generalisation to higher-order PDSs.

Let $Prop$ be a finite set of atomic propositions and $(\mathcal{P}, \mathcal{D}, \Sigma)$ be a higher-order PDS with a labelling function $\Lambda : \mathcal{P} \rightarrow 2^{Prop}$ which assigns to each control state a set of propositions deemed to be *true* at that state. Given formula ϕ of an ω -regular logic such as LTL or μ TL, we calculate the set of configurations C of $(\mathcal{P}, \mathcal{D}, \Sigma)$ such that every run from each $c \in C$ satisfies ϕ .

It is well known that any formula of an ω -regular logic has a Büchi automaton representation [31, 18, 30] etc.. We form the product of the higher-order PDS and the Büchi automaton corresponding to the negation of ϕ . This gives us a higher-order Büchi PDS; that is, a higher-order PDS with a set \mathcal{F} of accepting control states. Thus, model-checking reduces to the non-emptiness problem for higher-order Büchi PDSs. Specifically, we compute the set of configurations from which there is an infinite run visiting configurations with control states in \mathcal{F} infinitely often. Note that C is the complement of this set.

This problem can be reduced further to a number of applications of the reachability problem. We present a generalisation of the reduction of Bouajjani *et al.*. Let $[^1a]^1$ denote the order-1 stack consisting of a single character a and $[^l a]^l$ for $l > 1$ denote the stack consisting of a single order- $(l-1)$ stack $[^{(l-1)}a]^{(l-1)}$.

Proposition 4.1. *Let c be a configuration of an order- n Büchi PDS BP . It is the case that BP has an accepting run from c iff there exist distinct configurations $\langle p^j, [^n a]^n \rangle$ and $\langle p^j, \gamma_2 \rangle$ with $top_1(\gamma_2) = a$ and a configuration $\langle p^f, \gamma_1 \rangle$ such that $p^f \in \mathcal{F}$ and,*

- (1) $c \xrightarrow{*} \langle p^j, \gamma_3 \rangle$ for some γ_3 with $top_1(\gamma_3) = a$, and
- (2) $\langle p^j, [^n a]^n \rangle \xrightarrow{*} \langle p^f, \gamma_1 \rangle \xrightarrow{*} \langle p^j, \gamma_2 \rangle$

Proof. See Appendix E. □

We reformulate these conditions as follows, where C_n^Σ is the set of all order- n stacks over the alphabet Σ . We remind the reader that B_n^a is the n -store automaton accepting all n -stores γ such that $top_1(\gamma) = a$.

- (1) $c \in Pre^*(\{p^j\} \times \mathcal{L}(B_n^a))$,
- (2) $\langle p^j, [^n a]^n \rangle \in Pre^*((\mathcal{F} \times C_n^\Sigma) \cap Pre^+(\{p^j\} \times \mathcal{L}(B_n^a)))$

We can compute the set of pairs $\langle p^j, [^n a]^n \rangle$ satisfying (2) in n -EXPTIME by calculating $Pre^*(\{p^j\} \times \mathcal{L}(B_n^a))$ over the following higher-order PDS:

Definition 4.2. Given an order- n Büchi PDS $BP = (\mathcal{P}, \mathcal{D}, \Sigma, \mathcal{F})$ we define $BP' = (\mathcal{P} \times \{0, 1\}, \mathcal{D}', \Sigma)$ where,

$$\begin{aligned} \mathcal{D}' = & \{ ((p, 0), b, o, (p', 0)) \mid p \in \mathcal{P} \cap \overline{\mathcal{F}} \wedge (p, b, o, p') \in \mathcal{D} \} \cup \\ & \{ ((p, 0), b, o, (p', 1)) \mid p \in \mathcal{F} \wedge (p, b, o, p') \in \mathcal{D} \} \cup \\ & \{ ((p, 1), b, o, (p', 1)) \mid (p, b, o, p') \in \mathcal{D} \} \end{aligned}$$

Lemma 4.3. *There exists a run $\langle (p, 0), [^n a]^n \rangle \xrightarrow{*} \langle (p, 1), w' \rangle$ with $w' \in \mathcal{L}(B_n^a)$ in BP' iff $\langle p, [^n a]^n \rangle$ satisfies (2).*

Proof. See Appendix E.2. Since BP' is twice as large as BP , $Pre^*(\{p^j\} \times \mathcal{L}(B_n^a))$ for BP' can be calculated in n -EXPTIME. This gives the set of configurations satisfying (2). \square

To construct an n -store automaton accepting all configurations from which there is an accepting run, we calculate the configurations $\langle p^j, [^n a]^n \rangle$ satisfying the second condition. Since there are only finitely many $p^j \in \mathcal{P}$ and $a \in \Sigma$ we can perform a simple enumeration. We then construct an n -store automaton A corresponding to the n -store automata accepting configurations satisfying (2) and compute $Pre^*(\mathcal{L}(A))$.

Theorem 4.4. *Given an order- n Büchi PDS $BP = (\mathcal{P}, \mathcal{D}, \Sigma, \mathcal{F})$, we can calculate in n -EXPTIME the set of configurations C such that from all $c \in C$ there is an accepting run of BP .*

Proof. Let $exp_0(x) = x$ and $exp_n(x) = 2^{exp_{n-1}(x)}$. We appeal to Lemma 4.3 for each p^j and a (of which there are polynomially many) to construct an n -store automaton $\mathcal{O}(exp_n(2 \times |\mathcal{P}|))$ in size which accepts $\langle p^j, [^n a]^n \rangle$ iff it satisfies (2). Membership can be checked in polynomial time (Proposition B.3).

It is straightforward to construct an automaton A polynomial in size which accepts $\langle p, w \rangle$ iff $\langle p, [^n top_1(w)]^n \rangle$ satisfies (2). We can construct $Pre^*(\mathcal{L}(A))$ in n -EXPTIME. Thus, the algorithm requires n -EXPTIME. \square

Corollary 4.5. *Given an order- n PDS $(\mathcal{P}, \mathcal{D}, \Sigma)$ with a labelling function $\Lambda : \mathcal{P} \rightarrow 2^{Prop}$ and a formula ϕ of an ω -regular logic, we can calculate in $(n + 2)$ -EXPTIME the set of configurations C of $(\mathcal{P}, \mathcal{D}, \Sigma)$ such that every run from each $c \in C$ satisfies ϕ .*

Proof. The construction of BP is exponential in size. Hence, we construct the n -store multi-automaton A that accepts the set of configurations from which there is a run satisfying the negation of ϕ as described above in time $\mathcal{O}(exp_n(2^{|\phi|}))$. To calculate C we complement A as described in Appendix B.3. This may include an exponential blow-up in the transition relation of A , hence we have $(n + 2)$ -EXPTIME. \square

Observe that since we can test $c \in C$ by checking $c \notin \mathcal{L}(A)$ where A is defined as above, we may avoid the complementation step, giving us an $(n + 1)$ -EXPTIME algorithm.

4.2. Reachability Games. Our algorithm may be used to compute the winning region for a player in a two-player reachability game over higher-order PDSs. This generalises a result due to Cachat [25]. We call our players Eloise and Abelard.

Definition 4.6. Given an order- n PDS $(\mathcal{P}, \mathcal{D}, \Sigma)$, an order- n *Pushdown Reachability Game* (PRG) $(\mathcal{P}, \mathcal{D}, \Sigma, \mathcal{R})$ over the order- n PDS is given by a partition $\mathcal{P} = \mathcal{P}_A \uplus \mathcal{P}_E$ and a set \mathcal{R} of configurations considered winning for Eloise.

We write $\langle p, \gamma \rangle \in \mathcal{C}_E$ iff $p \in \mathcal{P}_E$ and $\langle p, \gamma \rangle \in \mathcal{C}_A$ iff $p \in \mathcal{P}_A$. From a configuration $\langle p, \gamma \rangle$ play proceeds as follows:

- If $\langle p, \gamma \rangle \in \mathcal{C}_A$, Abelard chooses a move $(p, a, o, p') \in \mathcal{D}$ with $top_1(\gamma) = a$ and $o(\gamma)$ defined. Play moves to the configuration $\langle p', o(\gamma) \rangle$.
- If $\langle p, \gamma \rangle \in \mathcal{C}_E$, Eloise chooses a move $(p, a, o, p') \in \mathcal{D}$ with $top_1(\gamma) = a$ and $o(\gamma)$ defined. Play moves to the configuration $\langle p', o(\gamma) \rangle$.

Eloise wins the game iff play reaches a configuration $\langle p, \gamma \rangle$ where $\langle p, \gamma \rangle \in \mathcal{R}$ or $p \in \mathcal{P}_A$ and Abelard is unable to choose a move. Abelard wins otherwise.

The *winning region* for a given player is the set of all configurations from which that player can force a win. The winning region for Eloise can be characterised using an *attractor* $Attr_E(\mathcal{R})$ defined as follows,

$$\begin{aligned} Attr_E^0(\mathcal{R}) &= \mathcal{R} \\ Attr_E^{i+1}(\mathcal{R}) &= Attr_E^i(\mathcal{R}) \cup \{ c \in \mathcal{C}_E \mid \exists c'. c \hookrightarrow c' \wedge c' \in Attr_E^i(\mathcal{R}) \} \\ &\quad \cup \{ c \in \mathcal{C}_A \mid \forall c'. c \hookrightarrow c' \Rightarrow c' \in Attr_E^i(\mathcal{R}) \} \\ Attr_E(\mathcal{R}) &= \bigcup_{i \geq 0} Attr_E^i(\mathcal{R}) \end{aligned}$$

Conversely, the winning region for Abelard is $\overline{Attr_E(\mathcal{R})}$. Intuitively, from a position in $Attr_E^i(\mathcal{R})$, Eloise's winning strategy is to simply choose a move such that the next configuration is in $Attr_E^{i-1}(\mathcal{R})$. Abelard's strategy is to avoid Eloise's winning region.

We can use backwards-reachability for order- n APDSs to calculate $Attr_E(\mathcal{R})$, and hence the winning regions of both Abelard and Eloise. To simplify the reduction, we make a *totality* assumption. That is, we assume a bottom-of-the-stack symbol \perp that is never popped nor pushed, and for all $a \in \Sigma \cup \{\perp\}$ and control states $p \in \mathcal{P}$, there exists a command $(p, a, o, p') \in \mathcal{D}$. This can be ensured by adding sink states p_{lose}^E and p_{lose}^A from which Eloise and Abelard lose the game. In particular, for every $p \in \mathcal{P}$ and $a \in \Sigma \cup \{\perp\}$ we have $(p, a, push_a, p_{lose}^x)$ where $x = E$ if $p \in \mathcal{P}_E$ or $x = A$ otherwise. Furthermore, the only commands available from p_{lose}^x are of the form $(p_{lose}^x, a, push_a, p_{lose}^x)$ for $x \in \{A, E\}$. To ensure that p_{lose}^A is losing for Abelard, we set $\langle p_{lose}^A, \gamma \rangle \in \mathcal{R}$ for all γ . Conversely, $\langle p_{lose}^E, \gamma \rangle \notin \mathcal{R}$ for all γ .

Definition 4.7. Given an order- n PRG $(\mathcal{P}, \mathcal{D}, \Sigma, \mathcal{R})$ we define an order- n APDS $(\mathcal{P}, \mathcal{D}', \Sigma)$ where,

$$\begin{aligned} \mathcal{D}' &= \{ (p, a, \{(o, p')\}) \mid (p, a, o, p') \in \mathcal{D} \wedge p \in \mathcal{P}_E \} \\ &\quad \cup \{ (p, a, \{(o, p')\}) \mid (p, a, o, p') \in \mathcal{D} \} \mid p \in \mathcal{P}_A \} \end{aligned}$$

Furthermore, let \mathcal{R}_{stuck} be the set of configurations $\langle p, \nabla \rangle$ such that $p \in \mathcal{P}_A$. The set \mathcal{R}_{stuck} is regular and represents the configurations reached if Abelard performs an move with an undefined next stack.

Let C_A^∇ be the set of order- n configurations with an undefined stack and a control state belonging to Abelard.

Theorem 4.8. *Given an order- n PRG, where \mathcal{R} is a regular set of configurations, and an order- n APDS as defined above, $Attr_E(\mathcal{R})$ is regular and equivalent to $Pre^*(\mathcal{R} \cup \mathcal{R}_{stuck}) \setminus C_A^\nabla$. Hence, computing the winning regions in the order- n PRG is n -EXPTIME.*

4.3. Model-Checking Branching-Time Temporal Logics. Generalising a further result of Bouajjani *et al.* [2], we show that backwards-reachability for higher-order APDSs may be used to perform model-checking for the alternation-free (propositional) μ -calculus over higher-order PDSs. Common logics such as CTL are sub-logics of the alternation-free μ -calculus.

4.3.1. *Preliminaries.* Given a set of atomic propositions $Prop$ and a finite set of variables χ , the propositional μ -calculus is defined by the following grammar,

$$\phi := \pi \in Prop \mid X \in \chi \mid \neg\phi \mid \phi_1 \cup \phi_2 \mid \diamond\phi \mid \mu X.\phi$$

with the condition that, for a formula $\mu X.\phi$, X must occur under an even-number of negations. This ensures that the logic is monotonic. As well as the usual abbreviations for \Rightarrow and \wedge , we may also use, $\Box\phi = \neg\diamond\neg\phi$, $\nu X.\phi(X) = \neg\mu X.\neg\phi(\neg X)$ and σ for either μ or ν . A σ -formula is of the form $\sigma X.\phi$.

A variable X is bound in ϕ if it occurs as part of a sub-formula $\sigma X.\phi'(X)$. We call an unbound variable *free* and write $\phi(X)$ to indicate that X is free in ϕ . A *closed* formula has no variables occurring free, otherwise the formula is *open*.

Formulae in *positive normal form* are defined by the following syntax,

$$\phi := \pi \in Prop \mid \neg\pi \mid X \in \chi \mid \phi_1 \cup \phi_2 \mid \phi_1 \cap \phi_2 \mid \diamond\phi \mid \Box\phi \mid \mu X.\phi \mid \nu X.\phi$$

We can translate any formula into positive normal form by “pushing in” the negations using the abbreviations defined above.

A σ -sub-formula of $\sigma X.\phi(X)$ is *proper* iff it does not contain any occurrence of X . We are now ready to define the alternation-free μ -calculus:

Definition 4.9. The alternation-free μ -calculus is the set of formulae in positive normal form such that for every σ -sub-formula ψ of ϕ we have,

- If ψ is a μ -formula, then all ν -sub-formulae of ψ are proper, and
- If ψ is a ν -formula, then all μ -sub-formulae of ψ are proper.

The *closure* $cl(\phi)$ of a formula ϕ is the smallest set such that,

- If $\psi_1 \wedge \psi_2 \in cl(\phi)$ or $\psi \vee \psi \in cl(\phi)$, then $\psi_1 \in cl(\phi)$ and $\psi_2 \in cl(\phi)$, and
- If $\diamond\psi \in cl(\phi)$ or $\Box\psi \in cl(\phi)$, then $\psi \in cl(\phi)$, and
- If $\sigma X.\psi(X) \in cl(\phi)$, then $\psi(\sigma X.\psi(X)) \in cl(\phi)$.

The closure of any formula is a finite set whose size is bounded by the length of the formula.

Finally, we give the semantics of the μ -calculus over higher-order PDSs. Given a formula ϕ , an order- n PDS $(\mathcal{P}, \mathcal{D}, \Sigma)$, a labelling function $\Lambda : \mathcal{P} \rightarrow 2^{Prop}$, and a valuation function \mathcal{V} assigning a set of configurations to each variable $X \in \chi$, the set of configurations $\llbracket \phi \rrbracket_{\mathcal{V}}$ satisfying ϕ is defined,

$$\begin{aligned} \llbracket \pi \rrbracket_{\mathcal{V}} &= \Lambda^{-1}(\pi) \times C_n^{\Sigma} \\ \llbracket X \rrbracket_{\mathcal{V}} &= \mathcal{V}(X) \\ \llbracket \neg\psi \rrbracket_{\mathcal{V}} &= (\mathcal{P} \times C_n^{\Sigma}) \setminus \llbracket \psi \rrbracket_{\mathcal{V}} \\ \llbracket \psi_1 \vee \psi_2 \rrbracket_{\mathcal{V}} &= \llbracket \psi_1 \rrbracket_{\mathcal{V}} \cup \llbracket \psi_2 \rrbracket_{\mathcal{V}} \\ \llbracket \diamond\psi \rrbracket_{\mathcal{V}} &= Pre(\llbracket \psi \rrbracket_{\mathcal{V}}) \\ \llbracket \mu X.\psi \rrbracket_{\mathcal{V}} &= \bigcap \{ C \subseteq \mathcal{P} \times C_n^{\Sigma} \mid \llbracket \psi \rrbracket_{\mathcal{V}[X \mapsto C]} \subseteq C \} \end{aligned}$$

where $\mathcal{V}[X \mapsto C]$ is the valuation mapping all variables $Y \neq X$ to $\mathcal{V}(Y)$ and X to C .

4.3.2. *Model-Checking the Alternation-Free μ -Calculus.* Given an order- n PDS $(\mathcal{P}, \mathcal{D}, \Sigma)$ with a labelling function $\Lambda : \mathcal{P} \rightarrow 2^{Prop}$, a formula ϕ of the alternation-free μ -calculus, and a valuation \mathcal{V} we show that we can generalise the construction of Bouajjani *et al.* to produce an n -store multi-automata A_{ϕ} accepting the set $\llbracket \phi \rrbracket_{\mathcal{V}}$.

Initially, we only consider formulae whose σ -sub-formulae are μ -formulae. We construct a product of the higher-order PDS and the usual “game” interpretation of ϕ [23, 24] as

follows: observing that commands of the form $(-, a, push_a, -)$ do not alter the contents of the stack, we construct the order- n PRG $A = (\mathcal{P}^{(\mathcal{P}, \phi)}, \mathcal{D}_{\mathcal{P}}^{\phi}, \Sigma, \mathcal{R})$ where $\mathcal{P}_A^{(\mathcal{P}, \phi)}$, $\mathcal{P}_E^{(\mathcal{P}, \phi)}$ and $\mathcal{D}_{\mathcal{P}}^{\phi}$ are the smallest sets such that for every $(p, \psi) \in \mathcal{P} \times cl(\phi)$ and $a \in \Sigma$,

- If $\psi = \psi_1 \vee \psi_2$, then we have $(p, \psi) \in \mathcal{P}_E^{(\mathcal{P}, \phi)}$ and $((p, \psi), a, push_a, (p, \psi_1)) \in \mathcal{D}_{\mathcal{P}}^{\phi}$ and $((p, \psi), a, push_a, (p, \psi_2)) \in \mathcal{D}_{\mathcal{P}}^{\phi}$,
- If $\psi = \psi_1 \wedge \psi_2$, then we have $(p, \psi) \in \mathcal{P}_A^{(\mathcal{P}, \phi)}$ and $((p, \psi), a, push_a, (p, \psi_1)) \in \mathcal{D}_{\mathcal{P}}^{\phi}$ and $((p, \psi), a, push_a, (p, \psi_2)) \in \mathcal{D}_{\mathcal{P}}^{\phi}$,
- If $\psi = \mu X. \psi'(X)$, then $(p, \psi) \in \mathcal{P}_A^{(\mathcal{P}, \phi)}$ and $((p, \psi), a, push_a, (p, \psi'(\psi))) \in \mathcal{D}_{\mathcal{P}}^{\phi}$,
- If $\psi = \diamond \psi'$ and $(p, a, o, p') \in \mathcal{D}$, then $(p, \psi) \in \mathcal{P}_E^{(\mathcal{P}, \phi)}$ and $((p, \psi), a, o, (p', \psi')) \in \mathcal{D}_{\mathcal{P}}^{\phi}$,
- If $\psi = \square \psi'$, then $(p, \psi) \in \mathcal{P}_A^{(\mathcal{P}, \phi)}$ and for every $(p, a, o, p') \in \mathcal{D}$ it is the case that $((p, \psi), a, o, (p', \psi')) \in \mathcal{D}_{\mathcal{P}}^{\phi}$.

Finally, we define the set of configurations \mathcal{R} that indicate that the formula ϕ is satisfied by $(\mathcal{P}, \mathcal{D}, \Sigma)$, Λ and \mathcal{V} . The set \mathcal{R} contains all configurations of the form,

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

If $\mathcal{V}(X)$ is regular for all X free in ϕ , then \mathcal{R} is also regular.

Commands of the form $(-, a, push_a, -)$ are designed to deconstruct sub-formulae into literals that can be evaluated immediately. These commands require that the top order-one stack is not empty — otherwise play would be unable to proceed. Correctness of the construction requires the top order-one stack to contain at least one stack symbol. This condition may be ensured with a special “bottom of the stack” symbol $\perp \in \Sigma$. This symbol marks the bottom of all order-one stacks and is never pushed or popped, except in the case of a command $(-, \perp, push_{\perp}, -)$. The use of such a symbol is common throughout the literature [12, 28, 25] etc..

Proposition 4.10. *Given the order- n PRG $A = (\mathcal{P}^{(\mathcal{P}, \phi)}, \mathcal{D}_{\mathcal{P}}^{\phi}, \Sigma, \mathcal{R})$ constructed from the order- n PDS $(\mathcal{P}, \mathcal{D}, \Sigma)$, a labelling function Λ , a valuation \mathcal{V} , and a formula ϕ of the alternation-free μ -calculus such that all σ -sub-formulae of ϕ are μ -sub-formulae, we have $\langle p, \gamma \rangle \in \llbracket \phi \rrbracket_{\mathcal{V}}$ iff $\langle (p, \phi), \gamma \rangle \in Attr_E(\mathcal{R})$.*

Proof. (Sketch) The result follows from the fundamental theorem of the propositional μ -calculus [23, 14]. If $\langle (p, \phi), \gamma \rangle \in Attr_E(\mathcal{R})$, then there is a winning strategy for Eloise in A . In the absence of ν -sub-formulae, this winning strategy defines a well-founded choice function and hence a well-founded pre-model for $(\mathcal{P}, \mathcal{D}, \Sigma)$, Λ , \mathcal{V} and ϕ with initial state $\langle p, \gamma \rangle$. Thus, by the fundamental theorem, $\langle p, \gamma \rangle$ satisfies ϕ .

In the opposite direction, if $\langle p, \gamma \rangle$ satisfies ϕ , then — by the fundamental theorem — there is a well-founded pre-model with choice function f . Since there are no $\nu X. \psi$ sub-formula in ϕ , all paths in the pre-model are finite and all leaves are of a form accepted by \mathcal{R} . Hence, a winning strategy for Eloise is defined by f and we have $\langle (p, \phi), \gamma \rangle \in Attr_E(\mathcal{R})$. \square

In the dual case — when all σ -sub-formulae of ϕ are ν -sub-formulae — we observe that the negation $\bar{\phi}$ of ϕ has only μ -sub-formulae. We construct $Attr_E(\mathcal{R})$ for $\bar{\phi}$ and complement the resulting n -store multi-automaton (see Appendix B.3) to construct the set of configurations satisfying ϕ .

We are now ready to give a recursive algorithm for model-checking with the alternation-free μ -calculus. We write $\Phi = \{\phi_i\}_{i=1}^m$ to denote a set of sub-formulae such that no ϕ_i is a sub-formula of another. Furthermore, we write $\phi[U/\Phi]$ where $U = \{U_i\}_{i=1}^m$ is a set of fresh variables to denote the simultaneous substitution in ϕ of ϕ_i with U_i for all $i \in \{1, \dots, m\}$. The following proposition is taken directly from [2]:

Proposition 4.11. *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 -sub-formulae (μ -sub-formulae) of ϕ with respect to the sub-formula relation. Then,*

$$\llbracket \phi \rrbracket_{\mathcal{V}} = \llbracket \phi[U/\Phi] \rrbracket_{\mathcal{V}'}$$

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 $\llbracket \phi_i \rrbracket_{\mathcal{V}}$. \square

Since, given a μ -formula (ν -formula) ϕ , the formula $\phi[U/\Phi]$ has only μ -sub-formulae (ν -sub-formulae) we can calculate $\llbracket \phi_i \rrbracket_{\mathcal{V}}$ for all $\phi_i \in \Phi$, using the above propositions to calculate an automaton recognising $\llbracket \phi \rrbracket_{\mathcal{V}}$.

Theorem 4.12. *Given an order- n PDS $(\mathcal{P}, \mathcal{D}, \Sigma)$, a labelling function Λ , a valuation function \mathcal{V} and a formula ϕ of the alternation-free μ -calculus, we can construct an n -store multi-automaton A such that $\mathcal{L}(A) = \llbracket \phi \rrbracket_{\mathcal{V}}$. \square*

4.3.3. Complexity. Let $exp_0(x) = x$ and $exp_n(x) = 2^{exp_{n-1}(x)}$. A formula ϕ can be described as a tree structure with ϕ at the root. Each node in the tree is a μ -sub-formula or a ν -sub-formula ψ of ϕ . The children of the node are all maximal ν -sub-formulae or μ -sub-formulae of ψ respectively. There are at most n_ϕ nodes in the tree, where n_ϕ is the length of ϕ . Let $n_{\mathcal{R}}$ be the number of states in the n -store automaton recognising \mathcal{R} . The size of this automata is linear in the size of the automata specifying \mathcal{V} for each variable X .

The n -store multi-automaton recognising $\llbracket \psi \rrbracket_{\mathcal{V}}$ for a leaf node ψ has $\mathcal{O}(exp_n(n_{\mathcal{R}}))$ states. Together with a possible complementation step (which does not increase the state-set) we require $\mathcal{O}(exp_{n+1}(n_{\mathcal{P}} \cdot n_\phi))$ time and \mathcal{B} may be of size $\mathcal{O}(exp_{n+1}(n_{\mathcal{V}}))$.

Similarly, the n -store multi-automaton recognising $\llbracket \psi \rrbracket_{\mathcal{V}}$ for an internal node ψ with children ϕ_1, \dots, ϕ_m has $\mathcal{O}(exp_n(\sum_{i=1}^m n_i + n_{\mathcal{R}}) \times 2^{b_i})$ states, where n_i is the size of the automaton recognising $\llbracket \phi_i \rrbracket_{\mathcal{V}_i}$ for $i \in \{1, \dots, m\}$ and b_i is the size of \mathcal{B} for that automaton. Due to the final complementation step, $|\mathcal{B}|$ may be of size $\mathcal{O}(exp_{n+1}(\sum_{i=1}^m n_i + n_{\mathcal{R}}))$, which is also the total time required.

Subsequently, the automaton A recognising $\llbracket \phi \rrbracket_{\mathcal{V}}$ has $\mathcal{O}(exp_{n_\phi \cdot n}(n_{\mathcal{R}}))$ states and can be constructed in $\mathcal{O}(exp_{(n_\phi \cdot n)+1}(n_{\mathcal{R}}))$ time. Since we may test $c \in C$ for any configuration c and set of configurations C by checking $c \notin \overline{C}$, we may avoid the final complementation step to give us an $\mathcal{O}(exp_{n_\phi \cdot n}(n_{\mathcal{R}}))$ time algorithm.

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. We plan to investigate the applications of this work to higher-order pushdown games with more general winning conditions. In his Ph.D. thesis, Cachat adapts the reachability algorithm of Bouajjani *et al.* [2] to calculate the winning regions in Büchi games over pushdown processes [25]. It is likely that our work will permit similar extensions. We also intend to generalise this work to higher-order collapsible pushdown automata, which can be used to study higher-order recursion schemes [29, 17]. This may provide the first steps into the study of the global model-checking problem over these structures. Finally, an alternative definition of higher-order pushdown systems defines the higher-order pop operation as the inverse of the push operation. That is, a stack may only be popped if it matches the stack below. The results of Carayol [4] show that the set $Pre^*(C_{Init})$ over these structures is regular, using Carayol's notion of regularity. However, the complexity of computing this set is unknown. We may attempt to adapt our algorithm to this setting, proving the required complexity bounds.

ACKNOWLEDGMENTS

We thank Olivier Serre and Arnaud Carayol for helpful discussions. We also thank the anonymous referees for their invaluable remarks.

REFERENCES

- [1] A. Bouajjani and A. Meyer. Symbolic Reachability Analysis of Higher-Order Context-Free Processes. In *Proc. 24rd Conf. on Found. of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *Lecture Notes in Computer Science*, Madras, India, December 2004. Springer Pub.
- [2] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *International Conference on Concurrency Theory*, pages 135–150, 1997.
- [3] A. Bouquet, O. Serre, and I. Walukiewicz. Pushdown games with the unboundedness and regular conditions. In *FSTTCS'03*, volume 2914 of *lncs*, pages 88–99. Springer-Verlag, 2003.
- [4] A. Carayol. Regular sets of higher-order pushdown stacks. In *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 *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] C. Löding, P. Madhusudan, and O. Serre. Visibly pushdown games. In *Proceedings of FSTTCS'04*, volume 3328 of *LNCS*, pages 408–420. Springer-Verlag, 2004.
- [8] D. Caucal. On infinite terms having a decidable monadic theory. In *Proc. MFCS'02*, pages 165–176, 2002. LNCS 2420.
- [9] D. E. Muller and P. E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theor. Comput. Sci.*, 37:51–75, 1985.
- [10] J. Engelfriet. Iterated pushdown automata and complexity classes. In *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 365–373, New York, NY, USA, 1983. ACM Press.
- [11] H. Gimbert. Parity and exploration games on infinite graphs. In *Proceedings of CSL'04*, volume 3210 of *LNCS*, pages 56–70. Springer-Verlag, 2004.

- [12] I. Walukiewicz. Pushdown processes: Games and model checking. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the Eighth International Conference on Computer Aided Verification CAV*, volume 1102, pages 62–74, New Brunswick, NJ, USA, / 1996. Springer Verlag.
- [13] J. A. Brzozowski and E. L. Leiss. On equations for regular languages, finite automata, and sequential networks. *Theor. Comput. Sci.*, 10:19–35, 1980.
- [14] J. Bradfield and C. Stirling. Modal logics and mu-calculi: an introduction, 2001.
- [15] K. Aehlig, J. G. de Miranda, and C.-H. L. Ong. Safety is not a restriction at level 2 for string languages. In *FoSSaCS*, pages 490–504, 2005.
- [16] M. Hague. *Saturation Methods for Global Model Checking Pushdown Systems*. PhD thesis, Oxford University Computing Laboratory. To appear.
- [17] M. Hague, A. S. Murawski, O. Serre, and C.-H. L. Ong. Collapsible pushdown automata and recursion schemes, 2006. Preprint, 13 pages.
- [18] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Banff Higher Order Workshop*, pages 238–266, 1995.
- [19] A. N. Maslov. Multilevel stack automata. *Problems of Information Transmission*, 15:1170–1174, 1976.
- [20] O. Serre. Note on winning positions on pushdown games with ω -regular conditions. *Information Processing Letters*, 85:285–291, 2003.
- [21] O. Serre. Games with winning conditions of high Borel complexity. In *Proceedings of ICALP'04*, volume 3142 of *LNCS*, pages 1150–1162. Springer-Verlag, 2004.
- [22] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS '06: Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science*, pages 81–90, Washington, DC, USA, 2006. IEEE Computer Society.
- [23] R. S. Streett and E. A. Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Inf. Comput.*, 81(3):249–264, 1989.
- [24] C. Stirling. Bisimulation, modal logic and model checking games. *Logic Journal of the IGPL*, 7(1):103–124, 1999.
- [25] T. Cachat. *Games on Pushdown Graphs and Extensions*. PhD thesis, RWTH Aachen, 2003.
- [26] T. Cachat and I. Walukiewicz. The complexity of games on higher order pushdown automata, 2007.
- [27] T. Cachat, J. Duparc, and W. Thomas. Solving pushdown games with a Σ_3 winning condition. In *Proceedings of the 11th Annual Conference of the European Association for Computer Science Logic (CSL'02)*, volume 2471 of *LNCS*, pages 322–336. Springer-Verlag, 2002.
- [28] T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-order pushdown trees are easy. In *FoSSaCS '02: Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures*, pages 205–222, London, UK, 2002. Springer-Verlag.
- [29] T. Knapik, D. Niwinski, P. Urzyczyn, and I. Walukiewicz. Unsafe grammars and panic automata. In *ICALP*, pages 1450–1461, 2005.
- [30] M. Y. Vardi. A temporal fixpoint calculus. In *POPL '88: Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 250–259, New York, NY, USA, 1988. ACM Press.
- [31] W. Thomas. Automata on infinite objects. *Handbook of theoretical computer science (vol. B): formal models and semantics*, pages 133–191, 1990.

APPENDIX A. NOTIONS OF REGULARITY

We show that our notion of a regular set of n -stores coincides with the definition of Bouajjani and Meyer [1]. Bouajjani and Meyer show that a set of n -stores is regular iff it is accepted by a *level n nested store automata*.

Because we are considering n -stores rather than configurations, we assume that there is only one control state, and hence, an n -store multi-automaton has only a single initial state. We also disregard the undefined store ∇ , since it is not strictly a store. Observe that we are left with n -store automata.

In the absence of alternation, the set of n -store automata is definitionally equivalent to the set of level n nested store automata in the sense of Bouajjani and Meyer. Hence, it is the case that every level n nested store automaton is also an n -store automaton.

We need to prove that every n -store automaton has an equivalent level n nested store automata. We present the following definition:

Definition A.1. Given an n -store automaton $A = (\mathcal{Q}, \Sigma, \Delta, q_0, \mathcal{Q}_f)$ we define a level n nested store automaton $\hat{A} = (2^{\mathcal{Q}}, \Sigma, \hat{\Delta}, \{q_0\}, 2^{\mathcal{Q}_f})$, where, if $n = 1$,

$$\hat{\Delta} = \{ (\{q_1, \dots, q_m\}, a, Q') \mid \forall i \in \{1, \dots, m\}. (\exists (q_i, a, Q_i) \in \Delta) \wedge Q' = Q_1 \cup \dots \cup Q_m \}$$

and if $n > 1$,

$$\hat{\Delta} = \left\{ (\{q_1, \dots, q_m\}, \hat{B}, Q') \mid \begin{array}{l} \forall i \in \{1, \dots, m\}. (\exists (q_i, B_i, Q_i) \in \Delta) \wedge \\ Q' = Q_1 \cup \dots \cup Q_m \wedge B = B_1 \cap \dots \cap B_m \end{array} \right\}$$

where \hat{B} is defined recursively and the construction of $B_1 \cap \dots \cap B_m$ is given in section B.3.

Property A.2. For any w , the run $\{q_1, \dots, q_m\} \xrightarrow{w} Q'$ exists in the n -store automaton A iff the run $\{q_1, \dots, q_m\} \xrightarrow{w} Q'$ exists in \hat{A} .

Proof. The proof is by induction over n and then by a further induction over the length of w .

Suppose $n = 1$. When $w = \varepsilon$ the proof is immediate. When $w = aw'$ we have in one direction,

$$\{q_1, \dots, q_m\} \xrightarrow{a} Q_1 \xrightarrow{w'} Q'$$

in A , and by induction over the length of the run, $Q_1 \xrightarrow{w'} Q'$ in \hat{A} . By definition of the runs of A we have $q_i \xrightarrow{a} Q_1^i$ for each $i \in \{1, \dots, m\}$ with $Q_1 = Q_1^1 \cup \dots \cup Q_1^m$. Hence, by definition of \hat{A} we have the transition $\{q_1, \dots, q_m\} \xrightarrow{a} Q_1^1 \cup \dots \cup Q_1^m = Q_1$. Hence we have the run $\{q_1, \dots, q_m\} \xrightarrow{w} Q'$ in \hat{A} as required.

In the other direction we have a run of the form

$$\{q_1, \dots, q_m\} \xrightarrow{a} Q_1 \xrightarrow{w'} Q'$$

in \hat{A} , and by induction over the length of the run, $Q_1 \xrightarrow{w'} Q'$ in A . By definition of the transition relation of \hat{A} we have $q_i \xrightarrow{a} Q_1^i$ in A for each $i \in \{1, \dots, m\}$ with $Q_1 = Q_1^1 \cup \dots \cup Q_1^m$. Hence, we have the transition $\{q_1, \dots, q_m\} \xrightarrow{a} Q_1^1 \cup \dots \cup Q_1^m = Q_1$ in A . Thus, we have the run $\{q_1, \dots, q_m\} \xrightarrow{w} Q'$ in A as required.

When $n > 1$, when $w = \varepsilon$ the proof is immediate. When $w = \gamma w'$ we have in one direction,

$$\{q_1, \dots, q_m\} \xrightarrow{\gamma} Q_1 \xrightarrow{w'} Q'$$

in A , and by induction over the length of the run, $Q_1 \xrightarrow{w'} Q'$ in \hat{A} . By definition of the runs of A we have $q_i \xrightarrow{B_i} Q_1^i$ with $\gamma \in \mathcal{L}(B_i)$ for each $i \in \{1, \dots, m\}$ with $Q_1 = Q_1^1 \cup \dots \cup Q_1^m$. Consequently, we have $\gamma \in \mathcal{L}(B)$ where $B = B_1 \cap \dots \cap B_m$. By induction over n we have $\gamma \in \mathcal{L}(\hat{B})$. Hence, by definition of \hat{A} we have the transition $\{q_1, \dots, q_m\} \xrightarrow{\gamma} Q_1^1 \cup \dots \cup Q_1^m = Q_1$. Hence we have the run $\{q_1, \dots, q_m\} \xrightarrow{w} Q'$ in \hat{A} as required.

In the other direction we have a run of the form

$$\{q_1, \dots, q_m\} \xrightarrow{\gamma} Q_1 \xrightarrow{w'} Q'$$

in \hat{A} . In particular, we have $\{q_1, \dots, q_m\} \xrightarrow{\hat{B}} Q_1$ in \hat{A} with $\gamma \in \mathcal{L}(\hat{B})$. By induction over the length of the run, $Q_1 \xrightarrow{w'} Q'$ in A . By definition of the transition relation of \hat{A} we have $q_i \xrightarrow{B_i} Q_1^i$ for each $i \in \{1, \dots, m\}$ with $B = B_1 \cap \dots \cap B_m$ and $Q_1 = Q_1^1 \cup \dots \cup Q_1^m$. By induction over n we have $\gamma \in \mathcal{L}(B)$ and hence $\gamma \in \mathcal{L}(B_i)$ for all $i \in \{1, \dots, m\}$. Therefore, we have $q_i \xrightarrow{\gamma} Q_1^i$ in A for all $i \in \{1, \dots, m\}$. Thus, we have the transition $\{q_1, \dots, q_m\} \xrightarrow{\gamma} Q_1^1 \cup \dots \cup Q_1^m = Q_1$ in A and the run $\{q_1, \dots, q_m\} \xrightarrow{w} Q'$ as required. \square

Corollary A.3. *A set of n -stores is definable by an n -store automaton iff it is definable by a level n nested store automaton. \square*

APPENDIX B. ALGORITHMS OVER n -STORE (MULTI-)AUTOMATA

In this section we describe several algorithms over n -store automata and n -store multi-automata. Observe that an n -store automaton is a special case of an n -store multi-automaton.

B.1. Enumerating Runs.

Proposition B.1. *Given a 1-store (multi-)automaton $A = (\mathcal{Q}, \Sigma, \Delta, -, \mathcal{Q}_f)$, a set of states Q and word w , the set of all Q' reachable via a run $Q \xrightarrow{w} Q'$ can be calculated in time $\mathcal{O}(2^{|\mathcal{Q}|})$.*

Proof. We define the following procedure, which given a set of sets of states Q_1 computes the set of sets Q' with $Q \in Q_1$ and $Q \xrightarrow{a} Q'$.

```

EXPAND( $a, Q_1$ )
  let  $Q_{next} = \emptyset$ 
  for each  $\{q_1, \dots, q_m\} \in Q_1$ 
    let  $ok = (\exists(q_1, a, -) \in \Delta)$ 
    let  $Q = \Delta(q_1, a)$ 
    for  $i = 2$  to  $m$ 
       $ok = ok \wedge (\exists(q_i, a, -) \in \Delta)$ 
       $Q = \{ Q' \cup Q'' \mid Q' \in Q \wedge (q_i, a, Q'') \in \Delta \}$ 
    if  $ok$  then  $Q_{next} = Q_{next} \cup Q$ 
  return  $Q_{next}$ 
    
```

The outer loop repeats $\mathcal{O}(2^{|\mathcal{Q}|})$ times and the inner loop $\mathcal{O}(|\mathcal{Q}|)$. Since the number of $Q' \in Q$ is $\mathcal{O}(2^{|\mathcal{Q}|})$ and the number of $(q_i, a, Q'') \in \Delta$ is also $\mathcal{O}(2^{|\mathcal{Q}|})$, construction of Q takes time $\mathcal{O}(2^{|\mathcal{Q}|})$. Hence the procedure takes time $\mathcal{O}(2^{|\mathcal{Q}|} \times |\mathcal{Q}| \times 2^{|\mathcal{Q}|})$, that is $\mathcal{O}(2^{|\mathcal{Q}|})$.

EXPAND is correct since $Q \in Q_{next}$ at the end of the procedure iff we have $\{q_1, \dots, q_m\} \in Q_1$ and some $(q_i, a, Q_{next}^i) \in \Delta$ for each $i \in \{1, \dots, m\}$ with $Q_{next} = Q_{next}^1 \cup \dots \cup Q_{next}^m$.

Over a word $w = a_1 \dots a_m$ we define the following procedure,

```

EXPANDWORD( $a_1 \dots a_m, Q$ )
  let  $Q_1 = \{Q\}$ 
  for  $i = 1$  to  $m$ 
     $Q_1 = \text{EXPAND}(a_i, Q_1)$ 
  return  $Q_1$ 
    
```

This procedure requires m runs of EXPAND and consequently runs in time $\mathcal{O}(2^{|\mathcal{Q}|})$.

We prove the correctness of EXPANDWORD by induction over the length of the word. When $w = a_1$ correctness follows from the correctness of EXPAND. In the inductive case $w = a_1 \dots a_m$. We have all runs of the form $Q \xrightarrow{a_1} Q_1$ as before, and all runs over $a_2 \dots a_m$ from all Q_1 by induction. We have all runs of the form $Q \xrightarrow{w} Q'$ therefrom. \square

Proposition B.2. *Given an l -store (multi-)automaton $A = (\mathcal{Q}, \Sigma, \Delta, _, \mathcal{Q}_f)$ with $l > 1$, and a set of states Q , the set of all Q' reachable via a run $Q \xrightarrow{\tilde{B}_1} Q' \xrightarrow{\tilde{B}_2} Q''$ can be calculated in time $\mathcal{O}(2^{|\Delta|+|\mathcal{Q}|})$.*

Proof. We define the following procedure, which given a set of states Q_1 computes the set of sets Q' and set of $(l-1)$ -store automata \tilde{B} with $Q \in Q_1$ and $Q \xrightarrow{\tilde{B}} Q'$.

```

EXPAND( $Q_1$ )
  let  $Q_{next} = \emptyset$ 
  for each  $\{q_1, \dots, q_m\} \in Q_1$ 
    for each set  $\{(q_1, B_1, Q_{next}^1), \dots, (q_m, B_m, Q_{next}^m)\} \subseteq \Delta$ 
       $Q_{next} = Q_{next} \cup \{(\{B_1, \dots, B_m\}, Q_{next}^1 \cup \dots \cup Q_{next}^m)\}$ 
  return  $Q_{next}$ 

```

The outer loop repeats at most $\mathcal{O}(2^{|\mathcal{Q}|})$ times. At most $\mathcal{O}(2^{|\Delta|})$ sets need to be enumerated during the inner loop. Hence, EXPAND runs in time $\mathcal{O}(2^{|\Delta|+|\mathcal{Q}|})$. The correctness of EXPAND is immediate.

To complete the algorithm, we define the following procedure,

```

EXPANDEXTIMES( $e, Q$ )
  let  $Q_1 = \text{EXPAND}(\{Q\})$ 
  for  $h = 1$  to  $e$ 
    for each  $(\tilde{B}_1, \dots, \tilde{B}_h, Q') \in Q_1$ 
       $Q_1 = Q_1 \cup (\{\tilde{B}_1, \dots, \tilde{B}_h\} \times \text{EXPAND}(\{Q'\}))$ 
  return  $Q_1 \cap ((\mathcal{B}_l)^e \times 2^{\mathcal{Q}})$ 

```

This procedure requires $\mathcal{O}(e \times (e \times 2^{|\Delta|}) \times 2^{|\mathcal{Q}|})$ iterations of the loop. Each iteration requires time $\mathcal{O}(2^{|\Delta|+|\mathcal{Q}|})$ and consequently the procedure runs in time $\mathcal{O}(2^{|\Delta|+|\mathcal{Q}|})$.

By the correctness of EXPAND we have $(\tilde{B}, Q') \in Q_1$ iff we have the path $Q \xrightarrow{\tilde{B}} Q'$ in A . After execution of the loop we have, by correctness of EXPAND, $(\tilde{B}_1, \dots, \tilde{B}_e, Q') \in Q_1$ iff we have the following path in A : $Q \xrightarrow{\tilde{B}_1} \dots \xrightarrow{\tilde{B}_e} Q'$. \square

B.2. Membership.

Proposition B.3. *Given an n -store (multi-)automaton $A = (\mathcal{Q}, \Sigma, \Delta, _, \mathcal{Q}_f)$ and an n -store w we can determine whether there is an accepting run over w in A from a given state $q \in \mathcal{Q}$ in time $\mathcal{O}(|w||\Delta||\mathcal{Q}|)$.*

Proof. When $w = \nabla$ we can check membership immediately. Otherwise the algorithm is recursive. In the base case, when $n = 1$ and $w = a_1 \dots a_m$, we present the following well-known algorithm,

```

let  $Q = Q_f$ 
for  $i = m$  downto 1
     $Q = \{ q' \mid (q', a_i, Q') \in \Delta \wedge Q' \subseteq Q \}$ 
return  $(q \in Q)$ 
    
```

This algorithm requires time $\mathcal{O}(m|\Delta||\mathcal{Q}|)$. We prove that this algorithm is correct at order-1 by induction over m . When $m = 1$, we have $q \in Q$ at the end of the algorithm iff there exists a transition $(q, a_1, Q') \in \Delta$ where $Q' \subseteq Q_f$. When $w = a_1 a_2 \dots a_m$ we have $q \in Q$ at the end of the algorithm iff there exists a transition (q, a_1, Q') where, by induction if $q' \in Q'$ then the word $a_2 \dots a_m$ is accepted from q' . Hence, we have $q \in Q$ iff there is an accepting run over w from q .

When $n > 1$ we generalise the algorithm given above. Let $w = \gamma_1 \dots \gamma_m$,

```

let  $Q = Q_f$ 
for  $i = m$  downto 1
     $Q = \{ q' \mid (q', B, Q') \in \Delta \wedge \gamma \in \mathcal{L}(B) \wedge Q' \subseteq Q \}$ 
return  $(q \in Q)$ 
    
```

The outer loop of the program repeats m times, there are $|\Delta|$ transitions to be checked. By considering all labelling automata as a single automaton with an initial state for each (as in the backwards reachability construction), we make a single recursive call (for each γ in w), obtaining all states accepting γ . Checking $\gamma \in \mathcal{L}(B)$ then requires checking whether the appropriate initial state is in the result of the recursive call. We have $|w| = |\gamma_1| + \dots + |\gamma_m|$, hence the algorithm requires $\mathcal{O}(|\gamma_1||\Delta_1||\mathcal{Q}| + \dots + |\gamma_m||\Delta_1||\mathcal{Q}|) = \mathcal{O}(|w||\Delta_1||\mathcal{Q}|)$ time for the pre-computation, then $\mathcal{O}(m|\Delta_2||\mathcal{Q}|)$ time for the body of the algorithm, where $\Delta = \Delta_1 \cup \Delta_2$ is the partition of Δ into the order- n and lower-order parts. Hence, we require $\mathcal{O}(|w||\Delta||\mathcal{Q}|)$ time.

We prove that this algorithm is correct at order $n > 1$ by induction over m . When $m = 1$, we have $q \in Q$ at the end of the algorithm iff there exists a transition $(q, B, Q') \in \Delta$ with $\gamma \in \mathcal{L}(B)$ and $Q' \subseteq Q_f$. When $w = \gamma_1 \gamma_2 \dots \gamma_m$ we have $q \in Q$ at the end of the algorithm iff there exists a transition (q, B, Q') where $\gamma \in \mathcal{L}(B)$ and, by induction, if $q' \in Q'$ then the word $a_2 \dots a_m$ is accepted from q' . Hence, we have $q \in Q$ iff there is an accepting run over w from q . \square

B.3. Boolean Operations. We can form the intersection, union and complement of n -store automata. Intersection and union are straightforward. We omit the details. To complement n -store multi-automata, we begin by defining an operation on sets of sets.

Definition B.4. Given a set of sets $\{Q_1, \dots, Q_m\}$ we define,

$$\text{invert}(\{Q_1, \dots, Q_m\}) = \{ \{q_1, \dots, q_m\} \mid q_i \in Q_i \wedge 1 \leq i \leq m \}$$

Definition B.5. Given an n -store multi-automaton $A = (\mathcal{Q}, \Sigma, \Delta, \{q^1, \dots, q^z\}, \mathcal{Q}_f)$, we define \bar{A} as follows.

- When $n = 1$ we assume A is *total* (this is a standard assumption that can easily be satisfied by adding a sink state). We define $\bar{A} = (\mathcal{Q}, \Sigma, \Delta', \{q^1, \dots, q^z\}, \mathcal{Q} \setminus \mathcal{Q}_f)$ where Δ' is the smallest set such that for each $q \in \mathcal{Q}$ and $a \in \Sigma$ we have,
 - (1) The transitions from q in Δ over a are $(q, a, Q_1), \dots, (q, a, Q_m)$, and
 - (2) $Q_a = \text{invert} \left(\bigcup_{i \in \{1, \dots, m\}} \{Q_i\} \right)$, and
 - (3) $\Delta'(q, a) = Q_a$.

Since Q_a may be exponential in size, the construction runs in exponential time when $n = 1$.

- When $n > 1$ we define $\bar{A} = (\mathcal{Q} \cup \{q_f^*, q_f^\varepsilon\}, \Sigma, \Delta', \{q^1, \dots, q^z\}, (\mathcal{Q} \cup \{q_f^*, q_f^\varepsilon\}) \setminus \mathcal{Q}_f)$ where $q_f^*, q_f^\varepsilon \notin \mathcal{Q}$, all n -stores are accepted from q_f^* and q_f^ε has no outgoing transitions.

Furthermore Δ' is the smallest set such that for each $q \in \mathcal{Q}$ we have,

- (1) The non- ∇ transitions from q in Δ are $(q, B_1, Q_1), \dots, (q, B_m, Q_m)$ (we assume $m \geq 1$), and
- (2) For all $\tilde{B} \in 2^{\{B_1, \dots, B_m\}}$ we have,

$$Q_{\tilde{B}} = \begin{cases} \{q_f^*\} & \text{if } \tilde{B} = \emptyset \\ \text{invert} \left(\bigcup_{B_i \in \tilde{B}} \{Q_i\} \right) & \text{otherwise} \end{cases}$$

$$B_{\tilde{B}} = \bigcap_{B_i \in \tilde{B}} B_i \cap \bigcap_{B_i \notin \tilde{B}} \bar{B}_i$$

Note we have \bar{B}_i recursively; and

- (3) $\Delta'(q, B_{\tilde{B}}) = Q_{\tilde{B}}$, and
- (4) For all $j \in \{1, \dots, z\}$ we have $(q^j, \nabla, \{q_f^\varepsilon\}) \in \Delta'$ iff there is no ∇ -transition from q^j in A .

Overall, when $n > 1$ there may be an exponential blow up in the number of transitions and the construction of each $B_{\tilde{B}}$ may take exponential time. The construction is therefore exponential.

We now show that the above definition is correct.

Property B.6. *Given an n -store multi-automaton A , we have $\mathcal{L}(\bar{A}^{q^j}) = \overline{\mathcal{L}(A^{q^j})}$ for all $q^j \in \{q^1, \dots, q^z\}$.*

Proof. We propose the following induction hypothesis: an accepting run $q \xrightarrow{w} Q$ exists in \bar{A} iff there is no accepting run $q \xrightarrow{w} Q'$ in A . We proceed first by induction over n and then by induction over the length of the run.

When $n = 1$, and the length of the run is zero, the induction hypothesis follows since $\mathcal{Q}_f \cap (\mathcal{Q} \setminus \mathcal{Q}_f) = \emptyset$. When the length of the run is larger than zero, we begin by showing the if direction. Assume we have an accepting run,

$$q \xrightarrow{a} Q^1 \xrightarrow{w} Q$$

in \bar{A} for some a and w . Suppose for contradiction we have a run,

$$q \xrightarrow{a} Q^2 \xrightarrow{w} Q'$$

in A with $Q' \subseteq \mathcal{Q}_f$. Then, by induction over the length of the run, there are no accepting runs over w in \bar{A} from any state in Q^2 . In Δ we have the transition (q, a, Q^2) . By definition there is some $q' \in Q^2$ with $q' \in Q^1$ and consequently the accepting run $Q^1 \xrightarrow{w} Q$ cannot exist in \bar{A} . We have a contradiction.

In the only-if direction, assume there is no run,

$$q \xrightarrow{a} Q^1 \xrightarrow{w} Q'$$

with $Q' \subseteq \mathcal{Q}_f$ in A . For all transitions of the form $q \xrightarrow{a} Q^1$ (guaranteed to exist since A is total) there is no accepting run $Q^1 \xrightarrow{w} Q'$. Hence, there is some $q' \in Q^1$ with no accepting run over w , and by induction over the length of the run, there is an accepting run from q' over w in \bar{A} .

Let $\{(q, a, Q_1^\top), \dots, (q, a, Q_e^\top)\}$ be the set of all transitions in Δ from q over a . For each $i \in \{1, \dots, e\}$, let $q_i^\top \in Q_i^\top$ be the state from which there is no accepting run over w in A and hence an accepting run over w in \bar{A} . By definition of Δ' the transition $q \xrightarrow{a} \{q_1^\top, \dots, q_e^\top\}$ exists in \bar{A} . Hence we have the accepting run,

$$q \xrightarrow{a} \{q_1^\top, \dots, q_e^\top\} \xrightarrow{w} Q'$$

in \bar{A} as required.

We now consider the inductive case $n > 1$. If $q = q_f^*$ or q_f^ε the result is immediate. Similarly, when the length of the run is zero, then the property follows since $Q_f \cap (Q \cup \{q_f^\varepsilon, q_f^*\}) \setminus Q_f = \emptyset$. Furthermore, since we have an (accepting) ∇ -transition from q^j for all $j \in \{1, \dots, z\}$ in A iff there is no (accepting) ∇ -transition from q^j in \bar{A} the result is also straightforward in this case.

Otherwise, in the if direction, assume we have an accepting run,

$$q \xrightarrow{\gamma} Q^1 \xrightarrow{w} Q$$

in \bar{A} for some γ and w . Suppose for contradiction we have a run,

$$q \xrightarrow{\gamma} Q^2 \xrightarrow{w} Q'$$

in A with $Q' \subseteq Q_f$. Then, by induction over the length of the run, there are no accepting runs over w in \bar{A} from any state in Q^2 . In Δ we have the transition (q, B, Q^2) with $\gamma \in \mathcal{L}(B)$, hence B must appear positively on the transition in Δ' from q to Q^1 (else \bar{B} appears, and by induction over n , $\gamma \notin \mathcal{L}(\bar{B})$). By definition there is some $q' \in Q^2$ with $q' \in Q^1$ and consequently the run $Q^1 \xrightarrow{w} Q$ cannot exist in \bar{A} . We have a contradiction.

In the only-if direction, assume there is no run,

$$q \xrightarrow{\gamma} Q^1 \xrightarrow{w} Q'$$

with $Q' \subseteq Q_f$ in A . There are two cases.

- If there are no transitions $q \xrightarrow{\gamma} Q^1$ in A then for all $q \xrightarrow{B} Q^1$ we have $\gamma \in \bar{B}$ by induction over n . Hence, in \bar{A} we have a run,

$$q \xrightarrow{\gamma} q_f^* \xrightarrow{w} Q^*$$

which is an accepting run as required.

- If there are transitions of the form $q \xrightarrow{\gamma} Q^1$ in A then for each of these runs there is no accepting run $Q^1 \xrightarrow{w} Q'$. Hence, there is some $q' \in Q^1$ with no accepting run over w , and by induction over the length of the run, there is an accepting run from q' over w in \bar{A} .

Let $\{(q, B_1^t, Q_1^t), \dots, (q, B_e^t, Q_e^t), (q, B_1^f, Q^f), \dots, (q, B_h^f, Q_h^f)\}$ be the set of all transitions in Δ from q such that $\gamma \in B_i^t$ for all $i \in \{1, \dots, e\}$ and $\gamma \notin B_i^f$ for all $i \in \{1, \dots, h\}$ (and consequently $\gamma \in \bar{B}_i^f$). For each $i \in \{1, \dots, e\}$ let $q_i^t \in Q_i^t$ be the state from which \bar{A} has no accepting run over w in A and hence has an accepting run over w in \bar{A} . By definition of Δ' the transition $q \xrightarrow{B} \{q_1^t, \dots, q_e^t\}$ with $B = B_1^t \cap \dots \cap B_e^t \cap \bar{B}_1^f \cap \dots \cap \bar{B}_h^f$ exists in \bar{A} . Hence we have the accepting run,

$$q \xrightarrow{\gamma} \{q_1^t, \dots, q_e^t\} \xrightarrow{w} Q'$$

in \bar{A} as required.

We have shown that \bar{A} has an accepting run from any state iff there is no accepting run from that state in A as required. \square

APPENDIX C. SOUNDNESS AND COMPLETENESS FOR A_0, A_1, \dots

In this section we show that the sequence A_0, A_1, \dots is sound and complete with respect to $Pre^*(C_{Init})$, where $C_{Init} = \mathcal{L}(A_0)$.

C.1. Preliminaries. We begin by proving some useful properties of the automaton construction. These properties assert that the automata constructed from the sets in $\tilde{\mathcal{G}}_i^i$ are well-behaved. Once this has been established, we need only consider order- n of the automata A_0, A_1, \dots to show soundness and completeness. Note that since no $g_{(q_1, Q_2)}^i$ is accepting, any store accepted by some $G_{(q_1, Q_2)}^i$ has a top_1 element.

In order to reason about a particular transition, we need to know its origin. Hence we introduce the notion of an *inherited* and a *derived* transition. The remaining lemmata fall into four categories:

- (1) Lemma C.3 shows that inherited runs are sound.
- (2) Lemma C.2 shows the completeness of inherited runs.
- (3) Lemma C.4 and Lemma C.5 show that derived runs are sound.
- (4) Lemma C.6 shows the completeness of derived runs.

Definition C.1. A non-empty run $g_{(q_1, Q_2)}^i \xrightarrow{w}_i Q$ of \mathcal{G}^i or $q^j \xrightarrow{w}_i Q$ of A_i can be characterised by its initial transition. There are two cases,

- A run of \mathcal{G}^i : Then $w = aw'$ and we have $g_{(q_1, Q_2)}^i \xrightarrow{a}_i Q'$. If the transition was inherited from $g_{(q_1, Q_2)}^{i-1}$ then we say that the run is an **inherited** run. Otherwise the transition was introduced by some $S \in \tilde{G}_{(q_1, Q_2)}^i$. We say that the run was **derived** from S .
- A run of A_i : We have $w = \gamma w'$ and $q^j \xrightarrow{\gamma}_i Q'$ with $\gamma \in \mathcal{L}(G)$.

If the accepting run of G was inherited, then the run is **inherited**. If the accepting run of G is derived from some $S' \in \tilde{G}$ and S' was added to \tilde{G} by $T_{\mathcal{D}}$ and the command d , then the run $g_{(q_1, Q_2)}^i \xrightarrow{w}_i Q$ is **derived** from d .

The language accepted by the sequence A_0, A_1, \dots or $\mathcal{G}^0, \mathcal{G}^1, \dots$ is increasing. In particular, if $q \xrightarrow{w}_i Q$ exists in A_i , then $q \xrightarrow{w}_{i+1} Q$ exists in A_{i+1} .

Lemma C.2.

- (1) If $g_{(q_1, Q_2)}^i \xrightarrow{w}_i Q$ is a run of $G_{(q_1, Q_2)}^i$ for some i (and $w \neq \varepsilon$), then $g_{(q_1, Q_2)}^{i+1} \xrightarrow{w}_{i+1} Q$ is a run of $G_{(q_1, Q_2)}^{i+1}$.
- (2) For all transitions $q \xrightarrow{\gamma}_i Q'$ in A_i for some i , we have the transition $q \xrightarrow{\gamma}_{i+1} Q'$ in A_{i+1} .
- (3) For all runs $q \xrightarrow{w}_i Q'$ of A_i for some i , we have the run $q \xrightarrow{w}_{i+1} Q'$ in A_{i+1} .

Proof. To prove (2) we observe that there are two cases. In the first case, the transition from q to Q' is labelled by an automaton $B \in \mathcal{B}$ or ∇ . Because this transition will remain unchanged by $T_{\mathcal{D}}$, the lemma follows immediately. In the second case, the transition is

labelled by $G_{(q,Q')}^i$ and the property follows directly from (1) and the run $g_{(q,Q')}^i \xrightarrow{w_\gamma}_i Q$ with $Q \subseteq \mathcal{Q}_f$ for $[w_\gamma] = \gamma$. Since $g_{(q,Q')}^i$ is not an accepting state, it is the case that $w_\gamma \neq \varepsilon$.

We note that (3) can be shown by repeated applications of (2).

Finally, we show (1). The automaton $G_{(q_1,Q_2)}^i$ has the run,

$$g_{(q_1,Q_2)}^i \xrightarrow{a}_i Q^1 \xrightarrow{w'}_i Q$$

where $w = aw'$.

By definition the automaton $G_{(q_1,Q_2)}^{i+1}$ has the transition $g_{(q_1,Q_2)}^{i+1} \xrightarrow{a}_i Q^2$ for every transition $g_{(q_1,Q_2)}^i \xrightarrow{a}_i Q^2$. Hence we have the run,

$$g_{(q_1,Q_2)}^{i+1} \xrightarrow{a}_{i+1} Q^1 \xrightarrow{w'}_i Q$$

as required. \square

Lemma C.3. *If a run $g_{(q_1,Q_2)}^{i+1} \xrightarrow{w}_{i+1} Q$ in \mathcal{G}^{i+1} is inherited, then the run $g_{(q_1,Q_2)}^i \xrightarrow{w}_i Q$ exists in \mathcal{G}^i .*

Proof. Observe that an inherited run cannot be empty. We have $w = aw'$ and,

$$g_{(q_1,Q_2)}^{i+1} \xrightarrow{a}_{i+1} Q' \xrightarrow{w'}_i Q$$

Since the run is an inherited run, we have $g_{(q_1,Q_2)}^i \xrightarrow{a}_i Q'$ in \mathcal{G}_1^i and hence,

$$g_{(q_1,Q_2)}^i \xrightarrow{a}_i Q' \xrightarrow{w'}_i Q$$

in \mathcal{G}^i as required. \square

Lemma C.4. *Suppose the run $g_{(q_1,Q_2)}^{i+1} \xrightarrow{w}_{i+1} Q$ derived from S exists in \mathcal{G}^{i+1} and $\theta_1 \in S$. We have $q^{\theta_1} \xrightarrow{w}_i Q'$ in \mathcal{G}^i , where $Q' \subseteq Q$.*

Proof. Observe that, since the run is derived, we have $w \neq \varepsilon$. Let $w = aw'$. We have the following run in \mathcal{G}^{i+1} ,

$$g_{(q_1,Q_2)}^{i+1} \xrightarrow{a}_{i+1} Q^1 \xrightarrow{w'}_i Q$$

and by definition, since the run is derived from S and $\theta_1 \in S$, we have $q^{\theta_1} \xrightarrow{a}_i Q^2$ in \mathcal{G}^i where $Q^2 \subseteq Q^1$, and hence,

$$q^{\theta_1} \xrightarrow{a}_i Q^2 \xrightarrow{w'}_i Q'$$

with $Q' \subseteq Q$ as required. \square

Lemma C.5. *Suppose the run $g_{(q_1,Q_2)}^{i+1} \xrightarrow{w}_{i+1} Q$ derived from S exists in \mathcal{G}^{i+1} and there is some $(a, o, \theta_1) \in S$. If $[w'] = o([w])$, we have $q^{\theta_1} \xrightarrow{w'}_i Q'$ in \mathcal{G}_1^i , where $Q' \subseteq Q$.*

Proof. Since the run is derived, we have $w \neq \varepsilon$. We have $w = aw''$. There is only one value of o , $o = \text{push}_{w_p}$ and $[w'] = o([w]) = [w_p w'']$. We have the following run in \mathcal{G}_1^{i+1} ,

$$g_{(q_1,Q_2)}^{i+1} \xrightarrow{a}_{i+1} Q^1 \xrightarrow{w''}_i Q$$

and by definition, since the run is derived from S and $(a, o, \theta_1) \in S$, we have $q^{\theta_1} \xrightarrow{w_p}_i Q^2$ in \mathcal{G}_1^i where $Q^2 \subseteq Q^1$, and hence,

$$q^{\theta_1} \xrightarrow{w_p}_i Q^2 \xrightarrow{w''}_i Q'$$

with $Q'_f \subseteq Q$ as required. \square

Lemma C.6. *Let $S = \{\alpha_1, \dots, \alpha_m\} \in \tilde{G}_{(q,Q)}^{i+1}$. Given some γ with $\text{top}_1(\gamma) = a$ such that for each $e \in \{1, \dots, m\}$ we have,*

- *If $\alpha_e = \theta_e$ then $\gamma_e = \gamma$ and $\gamma_e \in \mathcal{L}(\theta_e)$*
 - *If $\alpha_e = (b, o_e, \theta_e)$ then $b = a$, $o_e(\gamma) = \gamma_e$ and $\gamma_e \in \mathcal{L}(\theta_e)$*
- we have $\gamma \in \mathcal{L}(G_{(q,Q)}^{i+1})$.*

Proof. Let $\gamma = [aw]$. We have $\alpha_e = \theta_e$ or $\alpha_e = (a, \text{push}_{w_e}, \theta_e)$. We have,

- When $\alpha_e = \theta_e$, the run,

$$q^{\theta_e} \xrightarrow{a}_{i} Q_e \xrightarrow{w}_{i} Q_f^e$$

with $Q_f^e \subseteq Q_f$ in \mathcal{G}^i . Furthermore, $\gamma_e = \gamma$.

- When $\alpha_e = (a, \text{push}_{w_e}, \theta_e)$, the run,

$$q^{\theta_e} \xrightarrow{w_e}_{i} Q_e \xrightarrow{w}_{i} Q_f^e$$

with $Q_f^e \subseteq Q_f$ in \mathcal{G}^i . Furthermore, we have $\gamma_e = [w_e w]$.

Hence, since $S \in \tilde{G}_{(q,Q)}^{i+1}$, we have from the definition of $G_{(q,Q)}^{i+1}$ the run,

$$q_{(q,Q)}^{i+1} \xrightarrow{a}_{i+1} Q_1 \cup \dots \cup Q_m \xrightarrow{w}_{i} Q_f^1 \cup \dots \cup Q_f^m$$

with $Q_f^1 \cup \dots \cup Q_f^m \subseteq Q_f$. Hence $\gamma \in \mathcal{L}(G_{(q,Q)}^{i+1})$ as required. \square

C.2. Soundness. We show that for any configuration $\langle p^j, \gamma \rangle$ such that $\gamma \in \mathcal{L}(A_i^{q^j})$, for some i , we have $\langle p^j, \gamma \rangle \xrightarrow{*} C$ with $C \subseteq C_{\text{Init}}$. Let $I = \{q^1, \dots, q^z\}$. The following lemma describes the relationship between added transitions and the evolution of the order-2 PDS.

In the following lemma, the restrictions on w' are technical requirements in the case of pop_2 operations. They may be justified by observing that only the empty store is accepted from the state q_f^ε , and that, since initial states are never accepting, the empty store cannot be accepted from an initial state.

Lemma C.7. *For a given run $q^j \xrightarrow{w}_{i} Q$ of A_i there exists for any w' satisfying the conditions below, some C such that $\langle p^j, [ww'] \rangle \xrightarrow{*} C$, where C contains configurations of the form $\langle p^k, w''w' \rangle$ with $q^k \xrightarrow{w''}_{0} Q'$ or $\langle p^j, \nabla \rangle$ with $q^j \xrightarrow{\nabla}_{0} Q'$. Furthermore, the union of all such Q' is Q . We require $w' \neq \nabla$ and,*

- (1) *If $q_f^\varepsilon \in Q$ then $w' = \varepsilon$,*
- (2) *If $q^k \in Q$ for some q^k then $w' \neq \varepsilon$.*

Proof. The proof proceeds by induction on i . In the base case $i = 0$ and the property holds trivially. We now consider the case for $i + 1$. Since $T_{\mathcal{D}}$ does not add any ∇ -transitions, we can assume $w \neq \nabla$.

We perform a further induction over the length of the run. In the base case we have $w = \gamma$ (the case $w = \varepsilon$ is immediate with $C = \{\langle p^j, [w'] \rangle\}$) and consider the single transition $q^j \xrightarrow{\gamma}_{i+1} Q$. We assume that the transition is not inherited, else the property holds by Lemma C.3 and induction over i . If the transition is not inherited, then the run is derived

from some d and we have $\gamma \in \mathcal{L}(G_{(q^j, Q)}^{i+1})$ and the accepting run of $G_{(q^j, Q)}^{i+1}$ is derived from some $S \in \tilde{G}_{(q^j, Q)}^{i+1}$ introduced by during the processing of d .

Let $d = (p^j, a, \{(o_1, p^{k_1}), \dots, (o_m, p^{k_m})\})$. We have $\langle p^j, [\gamma w'] \rangle \hookrightarrow C'$ where,

$$C' = \{ \langle p^{k_t}, \gamma' \rangle \mid t \in \{1, \dots, m\} \wedge \gamma' = o_t([\gamma w']) \} \\ \cup \{ \langle p^j, \nabla \rangle \mid \text{if } o_t([\gamma w']) \text{ with } t \in \{1, \dots, m\} \text{ is not defined} \}$$

We can decompose the new transition as per the definition of $T_{\mathcal{D}}$. That is $Q = Q'_1 \cup \dots \cup Q'_m$. There are several cases:

- $o_t = \text{push}_2$.

By definition of $T_{\mathcal{D}}$, we have the run,

$$q^{k_t} \xrightarrow{\tilde{\theta}_1}_i Q' \xrightarrow{\tilde{\theta}_2}_i Q'_t$$

with $\{B_1^a\} \cup \tilde{\theta}_1 \cup \tilde{\theta}_2 \subseteq S$. By Lemma C.4 we have $\gamma \in \mathcal{L}(\{B_1^a\} \cup \tilde{\theta}_1 \cup \tilde{\theta}_2)$. Hence we have,

$$q^{k_t} \xrightarrow{\gamma}_i Q' \xrightarrow{\gamma}_i Q'_t$$

We have $\text{push}_2[\gamma w'] = [\gamma \gamma w']$ and $\langle p^{k_t}, [\gamma \gamma w'] \rangle \in C'$. Via induction over i we have the set C_t with $\langle p^{k_t}, o_t[\gamma w'] \rangle \xrightarrow{*} C_t$ which satisfies the lemma.

- $o_t = \text{pop}_2$.

We have $B_1^a \in S$. We have, by Lemma C.4, $\gamma \in \mathcal{L}(B_1^a)$.

If $Q_t = \{q^{k_t}\}$ then $\text{pop}_2[\gamma w'] = [w']$ since w' is non-empty and $C_t = \{\langle p^{k_t}, [w'] \rangle\}$. Note $q^{k_t} \xrightarrow{\varepsilon}_0 \{q^{k_t}\}$.

If $Q_t = \{q_f^\varepsilon\}$ then $w' = \varepsilon$ and $\text{pop}_2[\gamma w']$ is undefined. By definition of $T_{\mathcal{D}}$ we have $q^j \xrightarrow{\nabla}_0 \{q_f^\varepsilon\}$. Let $C_t = \{\langle p^j, \nabla \rangle\}$.

- $o_t = \text{push}_w$.

By definition, we have $q^{k_t} \xrightarrow{\theta}_i Q_t$ in A_i and $(a, o_t, \theta) \in S$. Hence, by Lemma C.5, we have $o_t[\gamma] \in \mathcal{L}(\theta)$ and the run $q^{k_t} \xrightarrow{o_t[\gamma]}_i Q_t$ in A_i . Furthermore, it is the case that $\langle p^{k_t}, o_t[\gamma w'] \rangle \in C'$ and via induction over i we have a set C' with $\langle p^{k_t}, o_t[\gamma w'] \rangle \xrightarrow{*} C_t$ which satisfies the lemma.

Hence, we have $\langle p^j, [w w'] \rangle \hookrightarrow C' \xrightarrow{*} C_1 \cup \dots \cup C_m = C$ where C satisfies the lemma.

This completes the proof of the single transition case. Let $w = \gamma_1 \dots \gamma_m$ and (for any Q) let $Q = Q^I \cup Q^{\setminus I}$ where Q^I contains all initial states in Q and $Q^{\setminus I} = Q \setminus Q^I$. We have the run,

$$q^j \xrightarrow{\gamma_1}_{i+1} Q_1 \xrightarrow{\gamma_2}_{i+1} \dots \xrightarrow{\gamma_m}_{i+1} Q_m$$

For each $q^k \in Q_1^I$ we have a run,

$$q^k \xrightarrow{\gamma_2}_{i+1} Q_2^k \xrightarrow{\gamma_3}_{i+1} \dots \xrightarrow{\gamma_m}_{i+1} Q_m^k$$

and by induction on the length of the run we have C_k such that $\langle p^k, [\gamma_2 \dots \gamma_m w'] \rangle \xrightarrow{*} C_k$ and C_k satisfies the lemma. Furthermore, since we only add new transitions to initial states, we have,

$$Q_1^{\setminus I} \xrightarrow{\gamma_2}_0 \dots \xrightarrow{\gamma_m}_0 Q_m'$$

and $Q_m = Q_m' \cup \bigcup_{q^k \in Q_1^I} Q_m^k$.

From $q^j \xrightarrow{\gamma_1}_{i+1} Q_1$ we have C_1 with $\langle p^j, [\gamma_1 \dots \gamma_m w'] \rangle \xrightarrow{*} C_1$ satisfying the lemma. Let C_1^I be the set of all $\langle p^k, \gamma_2 \dots \gamma_m w' \rangle \in C_1$ and $C_1' = C_1 \setminus C_1^I$. For each $q^k \in Q_1^I$

we have $\langle p^k, [\gamma_2 \dots \gamma_m w'] \rangle \in C_1$ since there are no transitions to initial states in A_0 (and hence we must have $q^k \xrightarrow{\varepsilon} \{q^k\}$ to satisfy the conditions of the lemma for C_1). From $\langle p^k, [\gamma_2 \dots \gamma_m w'] \rangle \xrightarrow{*} C_k$ and since we have $Q_1 \xrightarrow{\gamma_2 \dots \gamma_m} Q'_m$, it is the case that the set $C = C'_1 \cup \bigcup_{q^k \in Q'_1} C_k$ which has $\langle p^j, [\gamma_1 \dots \gamma_m w'] \rangle \xrightarrow{*} C_1 \xrightarrow{*} C$ and satisfies the lemma as required. \square

Property C.8 (Soundness). *For any configuration $\langle p^j, \gamma \rangle$ such that $\gamma \in \mathcal{L}(A_i^{q^j})$ for some i , we have $\langle p^j, \gamma \rangle \xrightarrow{*} C$ such that $C \subseteq C_{Init}$. That is, $\langle p^j, \gamma \rangle \in Pre^*(C_{Init})$.*

Proof. Let $\gamma = [w_\gamma]$. Since $\gamma \in \mathcal{L}(A_i^{q^j})$ we have a run $q^j \xrightarrow{w_\gamma} Q_f$ with $Q_f \subseteq \mathcal{Q}_f$. Since \mathcal{Q}_f contains no initial states, we apply Lemma C.7 with $w' = \varepsilon$. Therefore, we have $\langle p^j, \gamma \rangle \xrightarrow{*} C \subseteq \mathcal{L}(A_0^{q^k})$. Since A_0 is defined to represent C_{Init} , soundness follows. \square

C.3. Completeness.

Property C.9 (Completeness). *For all $\langle p^j, \gamma \rangle \in Pre^*(C_{Init})$ there is some i such that $\gamma \in \mathcal{L}(A_i^{q^j})$.*

Proof. We take $\langle p^j, \gamma \rangle \in Pre^*(C_{Init})$ and reason by induction over the length of the shortest path $\langle p^j, \gamma \rangle \xrightarrow{*} C$ with $C \subseteq C_{Init}$.

In the base case the path length is zero and we have $\langle p^j, \gamma \rangle \in C_{Init}$ and hence $\gamma \in \mathcal{L}(A_0^{q^j})$.

For the inductive step we have $\langle p^j, \gamma \rangle \xrightarrow{*} C_1 \xrightarrow{*} C_2$ with $C_2 \subseteq C_{Init}$ and some i such that $C_1 \subseteq \mathcal{L}(A_i)$ by induction. We show $\gamma \in \mathcal{L}(A_{i+1}^{q^j})$ by analysis of the higher-order APDS command d used in the transition $\langle p^j, \gamma \rangle \xrightarrow{*} C_1$.

Let $d = (p^j, a, \{(o_1, p^{k_1}), \dots, (o_m, p^{k_m})\})$. We have

$$C_1 = \{ \langle p^{k_t}, \gamma' \rangle \mid t \in \{1, \dots, m\} \wedge \gamma' = o_t(\gamma) \} \\ \cup \{ \langle p^j, \nabla \rangle \mid \text{if } o_t(\gamma) \text{ with } t \in \{1, \dots, m\} \text{ is not defined} \}$$

By induction we have for each $e \in \{1, \dots, m\}$ that $q^{k_e} \xrightarrow{w_{o_e(\gamma)}} Q_f^e$ with $Q_f^e \subseteq \mathcal{Q}_f$ in A_i if $o_e(\gamma) = [w_{o_e(\gamma)}]$ is defined. Otherwise we have $q^j \xrightarrow{\nabla} \{q_f^\varepsilon\}$ in A_i .

Let $\gamma = [\gamma' w]$. We have $S' = S'_1 \cup \dots \cup S'_m$ and $Q' = Q_1 \cup \dots \cup Q_m$ where, for each $e \in \{1, \dots, m\}$,

- When $o_e = push_2$, $o_e(\gamma) = [\gamma' \gamma' w]$. Additionally, we have the transitions,

$$q^{k_e} \xrightarrow{\theta_e^1} Q' \xrightarrow{\tilde{\theta}_e^2} Q_e$$

in A_i where $\gamma' \in \mathcal{L}(\{B_1^a, \theta_e^1\} \cup \tilde{\theta}_e^2)$. Furthermore, we have the run $Q_e \xrightarrow{w} Q_f^e$ with $Q_f^e \subseteq \mathcal{Q}_f$ and $S'_e = \{B_1^a, \theta_e^1\} \cup \tilde{\theta}_e^2$.

- When $o_e = pop_2$. If $o_e(\gamma) = [w]$, we have the run,

$$q^{k_e} \xrightarrow{w} Q_f^e$$

in A_i with $Q_f^e \subseteq \mathcal{Q}_f$, $S'_e = \{B_1^a\}$, $\gamma' \in \mathcal{L}(B_1^a)$ and $Q_e = \{q^{k_e}\}$.

If $o_e(\gamma)$ is undefined we have $w = \varepsilon$ and the run,

$$q^j \xrightarrow{\nabla} \{q_f^\varepsilon\}$$

if A_i . Hence we have $S'_e = \{B_1^a\}$, $\gamma' \in \mathcal{L}(B_1^a)$ and $Q_e = Q_f^e = \{q_f^e\}$.

- When $o_e = \text{push}_w$, and we have $o_e(\gamma) = [o_e(\gamma')w]$, and the transition $q^{k_e} \xrightarrow{\theta'_e} Q_e$ and run $Q_e \xrightarrow{w} Q_f^e$ with $Q_f^e \subseteq \mathcal{Q}_f$ in A_i . Additionally, $o_e(\gamma') \in \mathcal{L}(\theta'_e)$ and $S'_e = \{(a, o_e, \theta'_e)\}$. Hence, by definition of A_{i+1} , we have the transition,

$$q^j \xrightarrow{\tilde{G}}_{i+1} Q_1 \cup \dots \cup Q_m$$

with $S' \in \tilde{G}$ and by Lemma C.6 $\gamma' \in \mathcal{L}(G)$. Hence we have the run,

$$q^j \xrightarrow{\gamma'}_{i+1} Q_1 \cup \dots \cup Q_m \xrightarrow{w} Q_f^1 \cup \dots \cup Q_f^m$$

with $Q_f^1 \cup \dots \cup Q_f^m \subseteq \mathcal{Q}_f$ in A_{i+1} . That is, $\gamma \in \mathcal{L}(A_{i+1}^j)$ as required. \square

APPENDIX D. PROOFS FOR A_*

In this section we provide a proof of Lemma 3.8. The main idea of the proof is that the loops in $\hat{\mathcal{G}}^i$ can simulate, correctly, the prefix of any run in $\mathcal{G}^{i'}$ and vice-versa. That is, a run in $\hat{\mathcal{G}}^i$ begins by traversing its initial loops before progressing to its accepting states. If we unroll this looping we will construct a run of $\mathcal{G}^{i'}$ for a sufficiently large i' . In the other direction, the prefix of a run in $\mathcal{G}^{i'}$ can be simulated by the initial looping behaviour of $\hat{\mathcal{G}}^i$.

We begin by proving a small lemma that will ease the remaining proofs.

Lemma D.1. *Given $g_{(q_y, Q_y)}^{i_y} \xrightarrow{w}_{i_y} Q_y$ for all $y \in \{1, \dots, h\}$ for some h , let i_{max} be the maximum i_y . We have $\{g_{(q_1, Q_1)}^{i_{max}}, \dots, g_{(q_h, Q_h)}^{i_{max}}\} \xrightarrow{w} \bigcup_{y \in \{1, \dots, h\}} Q_y$.*

Proof. By Lemma C.2 we have $g_{(q_y, Q_y)}^{i_{max}} \xrightarrow{w}_{i_{max}} Q_y$ for each $y \in \{1, \dots, h\}$. Hence we have the run as required. \square

D.1. Proofs of Lemma 3.8.

Lemma D.2. *There exists some i_0 such that $\hat{\mathcal{G}}^i = \hat{\mathcal{G}}^{i_0}$ for all $i > i_0$. Furthermore, we have the run $g_{(q, Q')}^{i_1} \xrightarrow{w}_i Q_f$ with $Q_f \subseteq \mathcal{Q}_f$ for some i iff we have $g_{(q, Q')}^{i_1} \xrightarrow{w}_{i_0} Q_f$ in $\hat{\mathcal{G}}^{i_0}$.*

Proof. This is a simple consequence of the finiteness of Σ and that $T_{\hat{\mathcal{G}}^{i_1}[i_1/i_1-1]}$ only adds transitions and never states. The automaton will eventually become saturated and no new transitions will be added. \square

Lemma D.3. *For all w , if $g_{(q,Q')}^i \xrightarrow{w} Q_1$ with $Q_1 \subseteq \mathcal{Q}_f$ is a run in \mathcal{G}^i for some i , then we have $g_{(q,Q')}^{i_1} \xrightarrow{w} Q_2$ with $Q_2 \subseteq \mathcal{Q}_f$ in $\hat{\mathcal{G}}^{i_0}$.*

Proof. We prove the following property. For any path $g_{(q,Q')}^i \xrightarrow{w} \{q_1, \dots, q_h\}$ in \mathcal{G}^i , we have a path $g_{(q,Q')}^{i_1} \xrightarrow{w} \{q_1^!, \dots, q_h^!\}$ in $\hat{\mathcal{G}}^{i_0}$ with,

$$q_y^! = \begin{cases} g_{(q',Q'')}^{i'} & \text{if } q_y = g_{(q',Q'')}^{i'} \text{ and } i' \geq i_1 \\ q_y & \text{otherwise} \end{cases}$$

for all $y \in \{1, \dots, h\}$. Since $q_f^! = q_f$ for all $q_f \in \mathcal{Q}_f$, the lemma follows. When $Q = \{q_1, \dots, q_h\}$ we write $Q^!$ to denote the set $\{q_1^!, \dots, q_h^!\}$.

There are two cases. When $i \leq i_1$, then using that we have only added transitions to \mathcal{G}^{i_1} to define $\hat{\mathcal{G}}^{i_0}$ and that $q_y^! = q_y$ for all y , we have $g_{(q,Q')}^{i_1} \xrightarrow{w'} \{q_1^!, \dots, q_h^!\}$ in $\hat{\mathcal{G}}^{i_0}$.

We now consider the case $i > i_1$. We begin by proving that for a single transition,

$$g_{(q,Q')}^i \xrightarrow{b} \{q_1, \dots, q_h\}$$

in \mathcal{G}^i with $b \in \Sigma$, we have the following transition in $G_{(q,Q')}^{i_0}$,

$$g_{(q,Q')}^{i_1} \xrightarrow{b} \{q_1^!, \dots, q_h^!\}$$

We consider the source $S = \{\alpha_1, \dots, \alpha_m\} \in \tilde{G}_{(q,Q')}^i$ of the transition from $g_{(q,Q')}^i$. Since $\tilde{G}_{(q,Q')}^i \simeq \tilde{G}_{(q,Q')}^{i_1}$ we have $S[i_1/i-1] \in \tilde{G}_{(q,Q')}^{i_1}[i_1/i-1]$. Furthermore, we have $\{q_1, \dots, q_h\} = Q_1 \cup \dots \cup Q_m$. For $e \in \{1, \dots, m\}$ there are two cases,

- If $\alpha_e = \theta$, then let $g = q^\theta$. We have $g \xrightarrow{b} Q_e$ exists in \mathcal{G}^{i-1} . By induction over i we have $g^! \xrightarrow{b} Q_e^!$ in $\hat{\mathcal{G}}^{i_0}$.
- $\alpha_e = (a, \text{push}_{w_p}, \theta)$. Then $b = a$. Let $g = q^\theta$. By definition of $T_{\tilde{G}^{i_1}[i_1/i-1]}$, we have the path $g \xrightarrow{w_p} Q_e$ in \mathcal{G}^i . By induction on i we have the path $g^! \xrightarrow{w_p} Q_e^!$ in $\hat{\mathcal{G}}^{i_0}$.

We have $Q_1^! \cup \dots \cup Q_m^! = \{q_1^!, \dots, q_h^!\}$. Since $\tilde{G}_{(q,Q')}^i \simeq \tilde{G}_{(q,Q')}^{i_1}$ and $S[i_1/i-1] \in \tilde{G}_{(q,Q')}^{i_1}[i_1/i-1]$, by definition of $\hat{\mathcal{G}}^{i_0}$, we have,

$$g_{(q,Q')}^{i_1} \xrightarrow{b} \{q_1^!, \dots, q_h^!\}$$

in $\hat{\mathcal{G}}^{i_0}$ as required.

We now prove the result for a run of more than one step by induction over the length of the run. In the base case we have a run of a single transition. The result in this case has already been shown.

In the inductive case we have a run of the form,

$$g_{(q,Q')}^i \xrightarrow{a_0} \{q_1^1, \dots, q_{h_1}^1\} \xrightarrow{a_1} \dots \xrightarrow{a_m} \{q_1^m, \dots, q_{h_m}^m\}$$

in \mathcal{G}^i . For each $y \in \{1, \dots, h_1\}$ we have a run $q_y^1 \xrightarrow{a_1 \dots a_m} Q_y$ such that $\bigcup_{y \in \{1, \dots, h_1\}} Q_y = \{q_1^m, \dots, q_{h_m}^m\}$. By induction over the length of the run we have $q_y^{1!} \xrightarrow{a_1 \dots a_m} Q_y^!$ for each y . Hence, since we have $g_{(q,Q')}^{i_1} \xrightarrow{a_0} \{q_1^{1!}, \dots, q_{h_1}^{1!}\}$ from the above proof for one transition, we have a run of the form,

$$g_{(q,Q')}^{i_1} \xrightarrow{a_0} \{q_1^{1!}, \dots, q_{h_1}^{1!}\} \xrightarrow{a_1} \dots \xrightarrow{a_m} \{q_1^{!m}, \dots, q_{h_m}^{!m}\}$$

in $\hat{\mathcal{G}}^{i_0}$ as required. \square

Lemma D.4. *For all w , if we have $g_{(q,Q')}^{i_1} \xrightarrow{w}_i Q_f$ with $Q_f \subseteq \mathcal{Q}_f$ in $\hat{\mathcal{G}}^i$ for some i , then there is some i' such that the run $g_{(q,Q')}^{i'} \xrightarrow{w}_{i'} Q_f$ exists in $\mathcal{G}^{i'}$.*

Proof. We take a run of $\hat{\mathcal{G}}_{(q,Q')}^i$,

$$g_{(q,Q')}^{i_1} \xrightarrow{w}_i \{q_1, \dots, q_h\}$$

We show that for all $i^1 \geq i_1$, there is some $i^2 > i^1$ such that,

$$g_{(q,Q')}^{i^2} \xrightarrow{w}_{i^2} \{q_1^?, \dots, q_h^?\}$$

in $G_{(q,Q')}^{i^2}$ where, for $y \in \{1, \dots, h\}$,

$$q_y^? = \begin{cases} g_{(q',Q'')}^{i_1} & \text{if } q_1 = g_{(q',Q'')}^{i_1} \\ q_y & \text{otherwise} \end{cases}$$

Since $q_f^? = q_f$ for all $q_f \in \mathcal{Q}_f$, the lemma follows. For a set $Q = \{q_1, \dots, q_h\}$ we write $Q^? = \{q_1^?, \dots, q_h^?\}$.

The proof proceeds by induction over i . In the base case $i \leq i_1$ and the property holds by Lemma C.2 and since $\hat{\mathcal{G}}^{i_1} = \mathcal{G}_1^{i_1}$ and there are no incoming transitions to any $g_{(q',Q'')}^{i_1}$ in \mathcal{G}^{i_1} .

In the inductive case, we begin by showing for a single transition,

$$g_{(q,Q')}^{i_1} \xrightarrow{b}_i \{q_1, \dots, q_h\}$$

in $\hat{\mathcal{G}}_{(q,Q')}^i$ with $b \in \Sigma$, we have, for all $i^1 \geq i_1$, there is some $i^2 > i^1$ such that,

$$g_{(q,Q')}^{i^2} \xrightarrow{b}_{i^2} \{q_1^?, \dots, q_h^?\}$$

in $G_{(q,Q')}^{i^2}$. We analyse the $S \in \tilde{G}_{(q,Q')}^{i_1}[i_1/i_1 - 1]$ that spawned the transition from $g_{(q,Q')}^{i_1}$ (we assume the transition is new, else the property holds by induction).

Let $S = \{\alpha_1, \dots, \alpha_m\}$. We have $\{q_1, \dots, q_h\} = Q_1 \cup \dots \cup Q_m$. For each $e \in \{1, \dots, m\}$, there are several cases,

- $\alpha_e = \theta$.

Let $g_e = q^\theta$. By definition of $\hat{\mathcal{G}}^i$ we have the transition $g_e \xrightarrow{b}_{i-1} Q_e$ in $\hat{\mathcal{G}}^{i-1}$.

If $\theta = \tilde{G}_{(q',Q'')}^{i_1}$ then by induction we have $i_e^2 > i^1$ such that $g_{(q',Q'')}^{i_e^2} \xrightarrow{b}_{i_e^2} Q_e^?$ in $\mathcal{G}^{i_e^2}$.

Otherwise g_e is initial in some $B \in \mathcal{B}$ and the transition $g_e \xrightarrow{b}_{i-1} Q_e$ also exists in \mathcal{G}^0 and is the same as $g_e \xrightarrow{b}_{i_0} Q_e^?$. Let $w_e = b$.

- $\alpha_e = (a, \text{push}_{w_p}, \theta)$. Then $b = a$.

Let $g_e = q^\theta$. By definition of $\hat{\mathcal{G}}^i$ we have the run $g_e \xrightarrow{w_p}_{i-1} Q_e$ in $\hat{\mathcal{G}}^{i-1}$.

If $\theta = \tilde{G}_{(q',Q'')}^{i_1}$ then by induction we have $i_e^2 > i^1$ such that $g_{(q',Q'')}^{i_e^2} \xrightarrow{w_p}_{i_e^2} Q_e^?$ in $\mathcal{G}^{i_e^2}$.

Otherwise g_e is initial in some $B \in \mathcal{B}$ and the transition $g_e \xrightarrow{w_p}_{i-1} Q_e$ also exists in \mathcal{G}^0 and is the same as $g_e \xrightarrow{w_p}_{i_0} Q_e^?$. Let $w_e = w_p$.

Let i_{max} be the maximum i_e^2 . If $g_e = g_{(q', Q'')}^{i_1}$, we have, by Lemma C.2, $g_{(q', Q'')}^{i_{max}} \xrightarrow{w_e}_{i_{max}} Q_e^?$. Also, by Lemma C.2 we have $g_e \xrightarrow{w_e}_{i_{max}} Q_e^?$ when g_e is not of the form $g_{(q', Q'')}^{i_1}$. Since we have $\tilde{G}_{(q, Q')}^{i_{max}+1} \simeq \tilde{G}_{(q, Q')}^{i_1}$ we have $S[i_{max}/i_1 - 1] \in \tilde{G}_{(q, Q')}^{i_{max}+1}$ and since $Q_1^? \cup \dots \cup Q_m^? = \{q_1^?, \dots, q_h^?\}$ we have,

$$g_{(q, Q')}^{i_{max}+1} \xrightarrow{b}_{i_{max}+1} \{q_1^?, \dots, q_h^?\}$$

in $G_{(q, Q')}^{i_{max}+1}$. Let $i^2 = i_{max} + 1$ and we are done in the case of a single transition.

We now expand the result to a complete run by induction over the length of the run. That is, we take a run of $\tilde{G}_{(q, Q')}^i$,

$$g_{(q, Q')}^{i_1} \xrightarrow{w}_i \{q_1, \dots, q_h\}$$

and show that for all $i^1 \geq i_1$ there is some $i^2 > i^1$ such that,

$$g_{(q, Q')}^{i^2} \xrightarrow{w}_{i^2} \{q_1^?, \dots, q_h^?\}$$

in $G_{(q, Q')}^{i^2}$.

The base case has already been shown. We now consider the run,

$$g_{(q, Q')}^{i_1} \xrightarrow{a_0}_i \{q_1^1, \dots, q_{h_1}^1\} \xrightarrow{a_1}_i \dots \xrightarrow{a_m}_i \{q_1^m, \dots, q_{h_m}^m\}$$

We have $q_y^1 \xrightarrow{a_1 \dots a_m}_i Q_y$ for each $y \in \{1, \dots, h_1\}$ and $\bigcup_{y \in \{1, \dots, h_1\}} Q_y = \{q_1^m, \dots, q_{h_m}^m\}$. Then for all $y \in \{1, \dots, h_1\}$ via induction and Lemma D.1 we have for all $i^1 > i_1$ an i_{max} with

$$\{q_1^{?1}, \dots, q_{h_1}^{?1}\} \xrightarrow{a_1 \dots a_m}_{i_{max}} \{q_1^{?m}, \dots, q_{h_m}^{?m}\}$$

We then use the result for a single transition to obtain the result for the complete run. That is, we have for i_{max} an $i^2 > i_{max}$ such that,

$$g_{(q, Q')}^{i^2} \xrightarrow{a_0}_{i^2} \{q_1^{?1}, \dots, q_{h_1}^{?1}\} \xrightarrow{a_1 \dots a_m}_{i^2} \{q_1^{?m}, \dots, q_{h_m}^{?m}\}$$

exists in \mathcal{G}^{i^2} as required. \square

APPENDIX E. APPLICATIONS: PROOFS AND DEFINITIONS

E.1. Proof of Proposition 4.1.

Proof. We show a higher-order Büchi PDS has an accepting run iff the following condition holds: let c be a configuration of an order- n Büchi PDS BP . There is an accepting run in BP from c iff there exist distinct configurations $\langle p^j, [^n a]^n \rangle$ and $\langle p^j, \gamma_2 \rangle$ with $top_1(\gamma_2) = a$ and configuration $\langle p^f, \gamma_1 \rangle$ such that $p^f \in \mathcal{F}$ and,

- (1) $c \xrightarrow{*} \langle p^j, \gamma_3 \rangle$ for some γ_3 with $top_1(\gamma_3) = a$, and
- (2) $\langle p^j, [^n a]^n \rangle \xrightarrow{*} \langle p^f, \gamma_1 \rangle \xrightarrow{*} \langle p^j, \gamma_2 \rangle$

\Rightarrow : Every higher-order stack may be flattened into a well bracketed string, as per Definition 2.1. Given a suffix of an n -store w , let $comp(w)$ be a number of symbols “[” added to the beginning of w to form an n -store proper.

Given an accepting run of BP $\rho = c_0 c_1 \dots$, there exists a sequence of suffixes w_1, w_2, \dots such that there exists an increasing sequence of natural numbers i_1, i_2, \dots and for all $j > 0$ and $i \geq i_j$ c_i has a stack with the suffix w_j . Additionally c_{i_j} has the n -store $comp(w_j)$ and

w_i is a suffix of w_j for all $i \leq j$ (it may be the case that $w_i = w_j$). Take the sequence $c_{i_1}c_{i_2}\dots$. Due to the finiteness of \mathcal{P} and Σ there must be p, a with an infinite number of c_{i_j} with control state p and a stack whose top_1 element is a . Furthermore, since ρ is accepting, we must have distinct c_{i_a} and c_{i_b} with p as their control states and a as the top_1 element, with a c_f whose control state is $p^f \in \mathcal{F}$, and,

$$c_0 \xrightarrow{*} c_{i_a} \xrightarrow{*} c_f \xrightarrow{*} c_{i_b}$$

We have (1) from $c_0 \xrightarrow{*} c_{i_a}$. By definition of $c_{i_1}, c_{i_2} \dots$ we have $c_{i_a} = \langle p, comp(w_{i_a}) \rangle$ and all configurations between c_{i_a} and c_{i_b} have the suffix w_{i_a} . This implies,

$$\langle p, [^n a]^n \rangle \xrightarrow{*} \langle p^f, u \rangle \xrightarrow{*} \langle p, v \rangle$$

with $top_1(v) = a$. Hence, (2) holds as required.

\Leftarrow : From (1) we have $c \xrightarrow{*} \langle p, \gamma_1 \rangle$ with $top_1(\gamma_1) = a$. From (2) we can construct a path,

$$\langle p, \gamma_2 \rangle \xrightarrow{*} \langle p^f, \gamma_3 \rangle \xrightarrow{*} \langle p, \gamma_4 \rangle$$

with $p^f \in \mathcal{F}$ and $top_1(\gamma_4) = a$ for any γ_2 with $top_1(\gamma_2) = a$. Thus, through infinite applications of (2), we can construct an accepting run of BP . \square

E.2. Proof of Lemma 4.3.

Proof. We begin by showing that if $\langle p, [^n a]^n \rangle$ satisfies (2), then a run $\langle (p, 0), [^n a]^n \rangle \xrightarrow{*} \langle (p, 1), \gamma \rangle$ with $\gamma \in \mathcal{L}(B_n^a)$ exists in BP' . The run over BP satisfying (2) can be split into two parts,

$$\langle p, [^n a]^n \rangle \xrightarrow{*} \langle p^f, \gamma_f \rangle \xrightarrow{*} \langle p, \gamma \rangle$$

with $\gamma \in \mathcal{L}(B_n^a)$ and p^f is the first accepting state seen in the run. We consider each part separately.

- Suppose we have a run,

$$\langle p_0, \gamma_0 \rangle \hookrightarrow \dots \hookrightarrow \langle p_m, \gamma_m \rangle$$

such that p_m is the only accepting control state in the run. This run is derived from a sequence of commands d_1, \dots, d_m . Let $d_i = (p_{i-1}, a_i, o_i, p_i)$ for all $i \in \{1, \dots, m\}$. We show the run,

$$\langle (p_0, 0), \gamma_0 \rangle \hookrightarrow \dots \hookrightarrow \langle (p_m, 0), \gamma_m \rangle$$

exists in BP' by induction over m . In the base case $m = 0$ and the result is trivial. Suppose we have,

$$\langle (p_1, 0), \gamma_1 \rangle \hookrightarrow \dots \hookrightarrow \langle (p_m, 0), \gamma_m \rangle$$

by the induction hypothesis. Since $d_1 = (p_0, a_1, o_1, p_1)$ and $p_0 \notin \mathcal{F}$, we have that $((p_0, 0), a_1, o_1, (p_1, 0))$ is in \mathcal{D}' . Hence we have the run,

$$\langle (p_0, 0), \gamma_0 \rangle \hookrightarrow \dots \hookrightarrow \langle (p_m, 0), \gamma_m \rangle$$

as required.

- We have $\langle p^f, \gamma_f \rangle \in (\mathcal{F} \times C_n^\Sigma) \cap \text{Pre}^+(\{p\} \times \mathcal{L}(B_n^a))$, we show there exists the run $\langle (p^f, 0), \gamma_f \rangle \xrightarrow{*} \langle (p, 1), \gamma \rangle$ in BP' with $\gamma \in \mathcal{L}(B_n^a)$.

We have the run $\langle p^f, \gamma_f \rangle \xrightarrow{*} \langle p, \gamma \rangle$ in BP with $\gamma \in \mathcal{L}(B_n^a)$. This run is of the form,

$$\langle p_0, \gamma_0 \rangle \hookrightarrow \langle p_1, \gamma_1 \rangle \hookrightarrow \dots \hookrightarrow \langle p_m, \gamma_m \rangle$$

with $m \geq 1$, $p_0 = p^f$, $\gamma_0 = \gamma_f$, $p_m = p$ and $\gamma_m = \gamma$. The run is the consequence of a sequence of commands d_1, \dots, d_m . Let $d_i = (p_{i-1}, a_i, o_i, p_i)$. Since $p_0 \in \mathcal{F}$ we have $((p_0, 0), a_1, o_1, (p_1, 1))$ in \mathcal{D}' by definition. Furthermore, for $i \in \{2, \dots, m\}$ we have $((p_{i-1}, 1), a_i, o_i, (p_i, 1))$ in \mathcal{D}' . We have the run

$$\langle (p_0, 0), \gamma_0 \rangle \hookrightarrow \langle (p_1, 1), \gamma_1 \rangle \hookrightarrow \dots \hookrightarrow \langle (p_m, 1), \gamma_m \rangle$$

in BP' therefrom.

The proof of this direction follows immediately.

We now consider the proof in the opposite direction. Suppose we have $\langle (p, 0), [^n a]^n \rangle \xrightarrow{*} \langle (p, 1), \gamma \rangle$ with $\gamma \in \mathcal{L}(B_n^a)$. From the definition of \mathcal{D}' it follows that the run is of the form,

$$\langle (p, 0), [^n a]^n \rangle \hookrightarrow \dots \hookrightarrow \langle (p^f, 0), \gamma_f \rangle \hookrightarrow \langle (p', 1), \gamma' \rangle \hookrightarrow \dots \hookrightarrow \langle (p, 1), \gamma \rangle$$

where the second element of each control state/flag pair changes only in the position shown. Furthermore, p^f is the first occurrence of an accepting control state in BP. This run is the result of a sequence of commands d_1, \dots, d_m where $m \geq 1$. From a simple projection on the first element of each control state/flag pair, we immediately derive a sequence commands d'_1, \dots, d'_m in \mathcal{D} and the following run of BP,

$$\langle p, [^n a]^n \rangle \hookrightarrow \dots \hookrightarrow \langle p^f, \gamma_f \rangle \hookrightarrow \langle p', \gamma' \rangle \hookrightarrow \dots \hookrightarrow \langle p, \gamma \rangle$$

Since $\langle p^f, \gamma_f \rangle$ and $\langle p', \gamma' \rangle$ must be distinct, the existence of this run implies $\langle p, [^n a]^n \rangle$ satisfies (2). \square

E.3. Proof of $\text{Attr}_E(\mathcal{R}) = \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$.

Proof. We show $\text{Attr}_E(\mathcal{R}) = \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$. We begin by proving $\text{Attr}_E(\mathcal{R}) \supseteq \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$.

Take a configuration $\langle p, \gamma \rangle \in \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$. We show $\langle p, \gamma \rangle \in \text{Attr}_E(\mathcal{R})$ by induction over the shortest path $\langle p, \gamma \rangle \xrightarrow{*} C$ of the order- n APDS with $C \subseteq \mathcal{R}'$.

For the base case, we have $\langle p, \gamma \rangle \in \mathcal{R}' \setminus \mathcal{C}_A^\nabla$. Hence, $\langle p, \gamma \rangle \in \text{Attr}_E(\mathcal{R})$ since $\mathcal{R} \subseteq \text{Attr}_E(\mathcal{R})$.

Now, suppose we have $\langle p, \gamma \rangle \hookrightarrow C$ via the command $d = (p, a, OP)$ in the higher-order APDS with $C \in \text{Pre}^*(\mathcal{R}) \setminus \mathcal{C}_A^\nabla$ and by induction $C \subseteq \text{Attr}_E^i(\mathcal{R})$ for some i . There are two cases,

- If $p \in \mathcal{P}_A$ then for each $(o, p') \in OP$ and hence each move (p, a, o, p') in the higher-order PDS we have a corresponding $\langle p', \gamma' \rangle \in C$. We have either $\langle p', \gamma' \rangle \in \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$ or we have $\langle p', \gamma' \rangle = \langle p, \nabla \rangle$.

If we have $\langle p', \gamma' \rangle \in \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$ then $\langle p', \gamma' \rangle \in \text{Attr}_E^i(\mathcal{R})$ for some i by induction.

If we have $\langle p', \gamma' \rangle = \langle p, \nabla \rangle$ then $o(\gamma)$ is undefined. Hence (p, a, o, p') is not a valid move for Abeland.

Hence we have $\langle p, \gamma \rangle \in \mathcal{C}_A$ and $\forall c'. \langle p, \gamma \rangle \hookrightarrow c' \Rightarrow c' \in \text{Attr}_E^i(\mathcal{R})$ which implies $\langle p, \gamma \rangle \in \text{Attr}_E^{i+1}(\mathcal{R}) \subseteq \text{Attr}_E(\mathcal{R})$.

- If $p \in \mathcal{P}_E$ then $C = \{\langle p', o(\gamma) \rangle\}$ and $(p, a, o, p') \in \mathcal{D}$. Thus, we have $\exists c'. \langle p, \gamma \rangle \hookrightarrow c' \wedge c' \in \text{Attr}_E^i(\mathcal{R})$ and $\langle p, \gamma \rangle \in \mathcal{C}_E$. Therefore $\langle p, \gamma \rangle \in \text{Attr}_E^{i+1}(\mathcal{R}) \subseteq \text{Attr}_E(\mathcal{R})$.

Thus, we have $\text{Attr}_E(\mathcal{R}) \supseteq \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$ as required.

To show $\text{Attr}_E(\mathcal{R}) \subseteq \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$ we induct over i in $\text{Attr}_E(\mathcal{R}) = \bigcup_{i \leq 0} \text{Attr}_E^i(\mathcal{R})$. When $i = 0$ we have $\text{Attr}_E^0(\mathcal{R}) = \mathcal{R} \subseteq \mathcal{R}' \setminus \mathcal{C}_A^\nabla \subseteq \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$. For $i > 1$ there are two cases for all c such that $c \notin \text{Attr}_E^{i-1}(\mathcal{R})$ and $c \in \text{Attr}_E^i(\mathcal{R})$,

- $c \in \{c \in \mathcal{C}_E \mid \exists c'. c \hookrightarrow c' \wedge c' \in \text{Attr}_E^{i-1}(\mathcal{R})\}$.

Hence there is some command $d = (p, a, o, p')$ in the higher-order PDS and command $(p, a, \{(o, p')\})$ in the higher-order APDS. By induction $c' \in \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$ and $c = \langle p, \gamma \rangle$ and $c' = \langle p', o(\gamma) \rangle$. Hence $c \in \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$.

- $c \in \{c \in \mathcal{C}_A \mid \forall c'. c \hookrightarrow c' \Rightarrow c' \in \text{Attr}_E^{i-1}(\mathcal{R})\}$.

Let $c = \langle p, \gamma \rangle$. We have $d = (p, a, OP)$ in the higher-order APDS such that for all moves (p, a, o, p') we have $(o, p') \in OP$. If $o(\gamma)$ is defined, we have $\langle p, \gamma \rangle \hookrightarrow \langle p', o(\gamma) \rangle$ and $\langle p', o(\gamma) \rangle \in \text{Pre}^*(\mathcal{R}')$ by induction. If $o(\gamma)$ is undefined, then since we have $\langle p, \nabla \rangle \in \mathcal{R}'$ we have $\langle p, \nabla \rangle \in \text{Pre}^*(\mathcal{R}')$.

Thus, we have $\langle p, \gamma \rangle \hookrightarrow C$ via an application of the command d such that $C \subseteq \text{Pre}^*(\mathcal{R}')$. Hence $\langle p, \gamma \rangle \in \text{Pre}^*(\mathcal{R}')$ and since $\gamma \neq \nabla$, we have $\langle p, \gamma \rangle \in \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$ as required.

Thus, we have $\text{Attr}_E(\mathcal{R}) = \text{Pre}^*(\mathcal{R}') \setminus \mathcal{C}_A^\nabla$. □