Symbolic Backwards-Reachability Analysis for Higher-Order Pushdown Systems

M. Hague   C.-H. L. Ong

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

October 20th 2008

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, the computation of winning regions of reachability games and determining whether the word language accepted by a higher-order pushdown automata is non-empty.

Contents

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 a order-one stack — that is, a stack of characters — a order-two stack is a stack of order-one stacks. Similarly, a order-three stack is a stack of order-two stacks, and so on. An order-n 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}. Regular sets of configurations are 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 problem[1] of extending the backwards reachability result of Bouajjani and Meyer to the general case of higher-order PDSs (by taking into account a set of control states). In fact, we consider (and solve) the backwards reachability problem for the more general case of higher-order *alternating* pushdown systems (APDSs). Though slightly unwieldy, an advantage of the alternating framework is that it conveniently lends itself to a number of higher-order PDS verification problems. Following the work of Cachat [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. Finally, we use the algorithm to provide a technique for calculating the

---

[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]

non-emptiness of the word language accepted by a higher-order pushdown automaton. This final application shows that the problem is n-EXPTIME-hard.

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. Hence, 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 a chain 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

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 [13]. 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 [16]. 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* [16] 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). In Section 4 we go on to give a description of the algorithm in the order-n case. This section includes some further examples to illustrate the construction and a complexity analysis. Section 5 discusses the applications of the main result to LTL model-checking, reachability games, alternation-free μ-calculus model checking over higher-order PDSs and determining whether the language accepted by a higher-order pushdown automata is empty. This final application provides a proof that the backwards reachability problem is n-EXPTIME-complete. Finally, we conclude in Section 6. 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 [14] 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 $\{Q_1, \ldots, Q_m\}$ with $Q_i \in 2^{\mathcal{Q}}$ for $i \in \{1, \ldots, m\}$. When the automaton is viewed as a game, Eloise — the existential player — chooses a set $Q \in \Delta(q, \gamma)$; Abelard — the universal player — then chooses a state $q \in Q$. 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 *mu*-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 $[a_1, \ldots, a_m]$ with $m \geq 0$ and $a_i \in \Sigma$ for all $i \in \{1, \ldots, m\}$, $[\notin \Sigma$ and $] \notin \Sigma$. For $n > 1$, $C_n^{\Sigma} = [w_1, \ldots, w_m]$ with $m \geq 1$ and $w_i \in C_{n-1}^{\Sigma}$ for all $i \in \{1, \ldots, m\}$.

There are three types of operations applicable to n-stores: *push*, *pop* and *top*. These are defined inductively. Over a 1-store, we have (for all $w \in \Sigma^*$),

$$\begin{aligned} push_w[a_1 \ldots a_m] &= [wa_2 \ldots a_m] \\ top_1[a_1 \ldots a_m] &= a_1 \end{aligned}$$

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

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

Note that we assume 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 $\mathcal{O}_n = \{\ push_w \mid w \in \Sigma^*\ \} \cup \{\ push_l, pop_l \mid 1 < l \leq n\ \}$. 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 $(p, a, o, p') \in \mathcal{D}$, $top_1(\gamma) = a$ and $\gamma' = o(\gamma)$.

We define $\overset{*}{\hookrightarrow}$ to be the transitive closure of $\hookrightarrow$. For a set of configurations C_{Init} we define $Pre^*(C_{Init})$ as the set of configurations $\langle p, \gamma \rangle$ such that, for some configuration $\langle p', \gamma' \rangle \in C_{Init}$, we have $\langle p, \gamma \rangle \overset{*}{\hookrightarrow} \langle p', \gamma' \rangle$.

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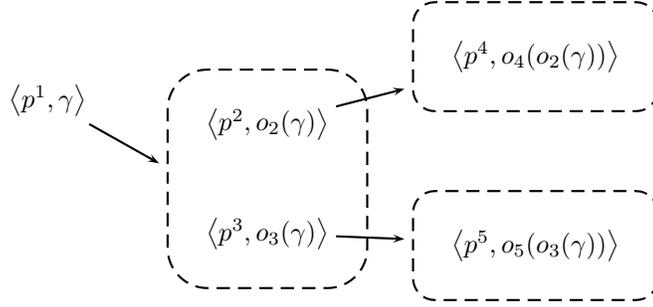
Figure 1: The configuration graph (excerpt) of an example higher-order APDS.

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 $(p, a, OP) \in \mathcal{D}$, $top_1(\gamma) = a$, 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 $\overset{*}{\hookrightarrow}$ to be the transitive closure of $\hookrightarrow$. For a set of configurations C_{Init} we define $Pre^*(C_{Init})$ as the set of configurations $\langle p, \gamma \rangle$ such that we have $\langle p, \gamma \rangle \overset{*}{\hookrightarrow} C$ and $C \subseteq C_{Init}$.

Example 2.1 *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 with the commands,*

$$(p^1, \_, \{(o_2, p^2), (o_3, p^3)\})$$
$$(p^2, \_, \{(o_4, p^4)\})$$
$$(p^3, \_, \{(o_5, p^5)\})$$

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 $C_{Init} = \{\langle p^5, \triangledown \rangle\}$, 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 $\langle p, \triangledown \rangle$ we can reduce the reachability problem for higher-order PDSs to the reachability problem over higher-order APDSs in a straightforward manner.

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

2.3 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 2.4

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

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

   where $\mathcal{Q}$ is a finite set of states, Σ is a finite alphabet, q^i for $i \in \{1, \ldots, z\}$ 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,

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

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

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

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

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

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

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

$\bigcup_{y \in \{1,\dots,m_x\}} \{B_y\} = \widetilde{B}_x$. Observe that $\widetilde{B}_0$ 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,

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

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

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

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 and X_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 $top_1(\gamma) = a$. The $(n-1)$-store automaton X_l^a accepts all $(n-1)$-stacks such that $top_1(\gamma) = a$ and $top_{l+1}(\gamma) = [[w']]$ for some w'. That is, $pop_l(\gamma)$ is undefined. If θ represents a store automaton, the state q^θ refers to the initial state of the automaton represented by θ.

3 The Order-Two Case

Before introducing the full algorithm, we give an example and a description of the algorithm for the order-2 case. This will provide an introduction to the important features of the solution whilst reducing the notational complexity.

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 $Pre^*(C_{Init})$. Thus, $Pre^*(C_{Init})$ is regular.*

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 $Pre^*(C_{Init})$; that is $\bigcup_{i \geq 0} \mathcal{L}(A_i) = Pre^*(C_{Init})$. To conclude the algorithm, we construct a single automaton A_* such that $\mathcal{L}(A_*) = \bigcup_{i \geq 0} \mathcal{L}(A_i)$.

We assume 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 $q_f^\varepsilon \in \mathcal{Q}_f$ which 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:

$$ \begin{aligned} d_1 &= (p^1, a, push_2, p^1) \\ d_2 &= (p^1, a, push_\varepsilon, p^1) \\ d_3 &= (p^2, a, push_w, p^1) \\ d_4 &= (p^2, a, pop_2, p^1) \end{aligned} $$

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 $d_1, \dots, d_4$.
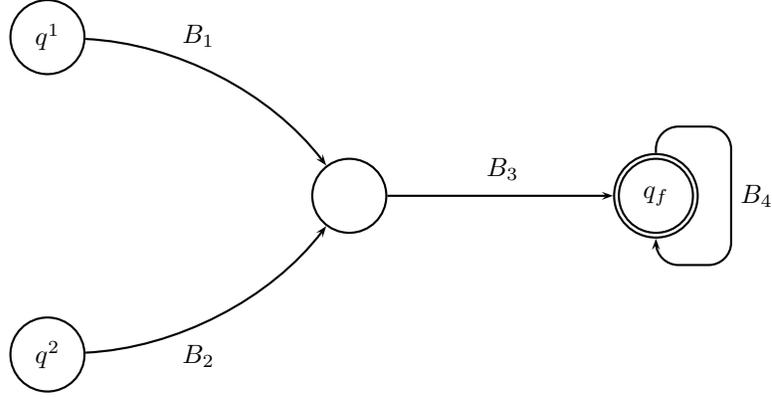
Figure 2: The initial 2-store multi-automaton
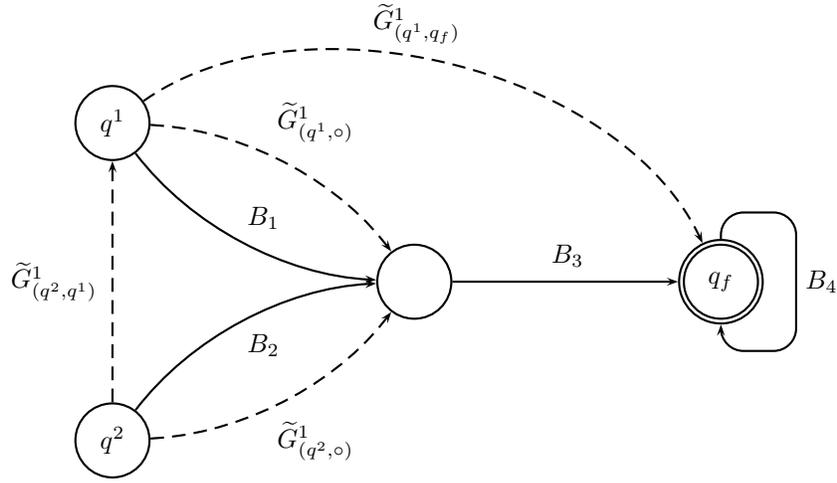
Figure 3: The automaton A_1

During the construction of A_1, 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 $\widetilde{G}^1_{(q_1,q_2)}$. This set is a recipe for the construction of an order-1 store automaton that will ultimately label the edge. The set $\widetilde{\mathcal{G}}^1$ is the set of all sets $\widetilde{G}^1_{(q_1,q_2)}$ introduced. After the first stage of the algorithm, the resulting A_1 is given in Figure 3 where the contents of

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

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 $\langle p_1, [\gamma_1 \ldots \gamma_m] \rangle$ with $top_1(\gamma_1) = a$ for each configuration $\langle p_1, [\gamma_1 \gamma_1 \ldots \gamma_m] \rangle$ accepted by A_0. This is because $push_2[\gamma_1 \ldots \gamma_m] = [\gamma_1 \gamma_1 \ldots \gamma_m]$. Hence we add the transition from q^1 to q_f. The contents of $\widetilde{G}^1_{(q^1,q_f)}$ indicate that this edge must accept the product of B_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 $\widetilde{G}^1_{(q^1,\circ)}$ and $\widetilde{G}^1_{(q^2,\circ)}$ specify that the automaton B_1 must be manipulated to produce the automaton that will label these new transitions. Finally, since $pop_2[\gamma_1 \ldots \gamma_m] = [\gamma_2 \ldots \gamma_m]$,

10

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

Table 1: The contents of the sets in $\widetilde{\mathcal{G}}^1$.
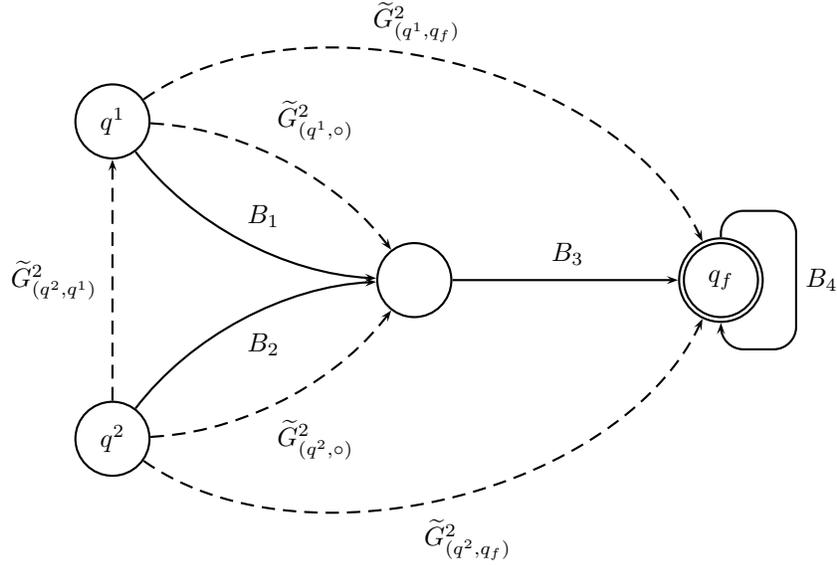
Figure 4: The automaton A_2.

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

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

are given in Table 2.

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

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 $G^1_{(q,q')}$ represented by the labels $\widetilde{G}^1_{(q,q')}$ for

11

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

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

the appropriate q, q'. Because each of these new automata will be constructed from $B_1, \ldots, B_4, B_1^a$, we build them simultaneously, constructing a single (1-store multi-)automaton $\mathcal{G}^1$ with an initial state $g^1_{(q,q')}$ for each $\widetilde{G}^1_{(q,q')}$. The automaton $\mathcal{G}^1$ is constructed through the addition of states and transitions to the disjoint union of $B_1, \ldots, B_4, B_1^a$. 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 $g^1_{(q^1,q_f)}$), a transition derived from $\{(a, push_\varepsilon, B_1)\}$ (from state $g^1_{(q^1,\circ)}$) and a transition derived from $\{B_1^a\}$ (from state $g^1_{(q^2,q^1)}$).

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 $G^1_{(q^1,\circ)}, G^1_{(q^1,q_f)}, G^1_{(q^2,\circ)}$ and $G^1_{(q^2,q^1)}$ by setting the initial state to $g^1_{(q^1,\circ)}, g^1_{(q^1,q_f)}, g^1_{(q^2,\circ)}$ and $g^1_{(q^2,q^1)}$ 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 $g^2_{(q^1,q_f)}$ is derived from the set $\{B_1^a, B_3, \widetilde{G}^1_{(q^1,\circ)}\}$. One of the transitions from $g^2_{(q^1,\circ)}$ and the only transition from $g^2_{(q^2,q^1)}$ 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 $g^2_{(q^1,\circ)}$ derives from $\{(a, push_\varepsilon, \widetilde{G}^1_{(q^1,\circ)})\}$. From this automaton we derive $G^1_{(q^1,\circ)}, G^1_{(q^1,q_f)}, G^1_{(q^2,\circ)}$ and $G^1_{(q^2,q^1)}$.

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

To construct A_* 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.

In particular, we are no longer required to add new states to $\mathcal{G}^2$ to construct $\mathcal{G}^i$ for $i > 2$. Instead, we fix the update instructions $\widetilde{G}^2_{(q,q')}[2/1]$ for all q, q' and manipulate $\mathcal{G}^2$ as we manipulated
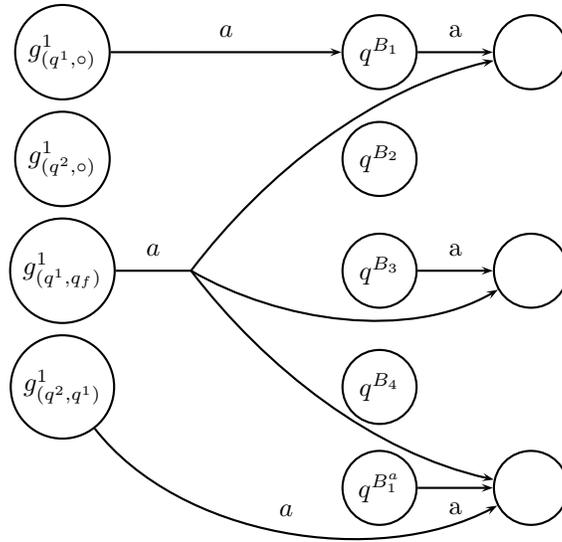
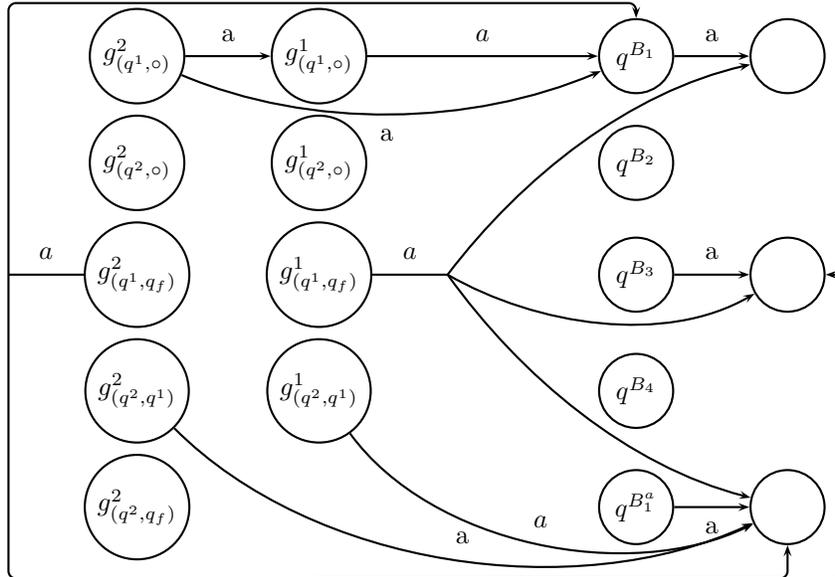Figure 5: A selective view of $\mathcal{G}^1$.

Figure 6: A selective view of $\mathcal{G}^2$.
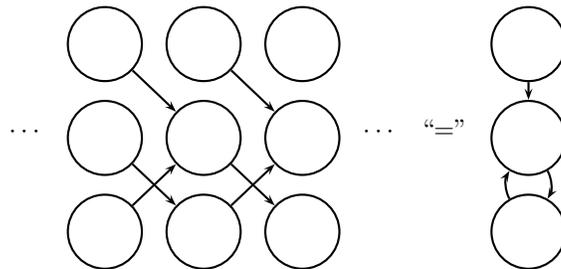
Figure 7: Collapsing a repetitive chain of new states.

13
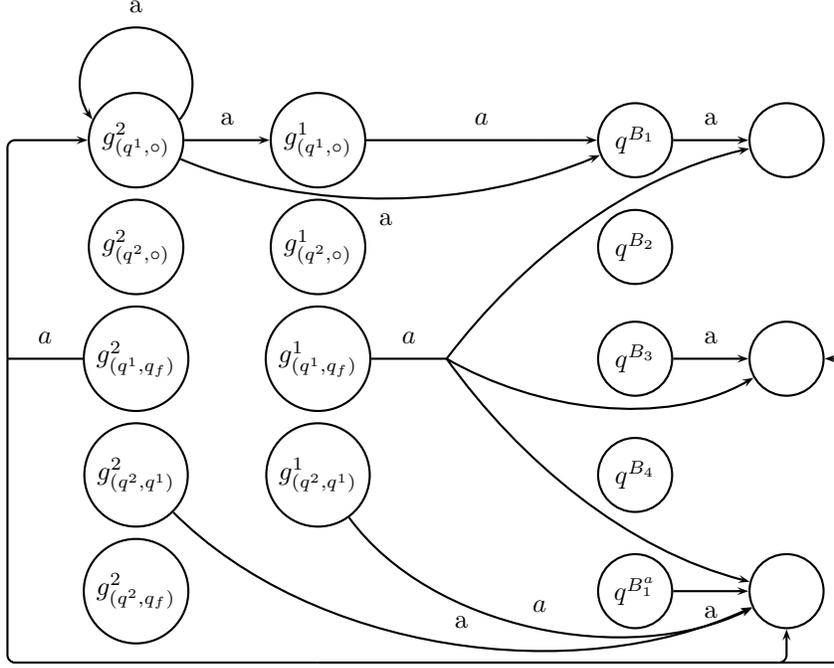
Figure 8: A selective view of $\hat{\mathcal{G}}^*$.

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 $\{B_l^a, B_3, \widetilde{G}_{(q^1,\circ)}^1\}$ in Figure 6. Instead, we show the transition introduced for the set $\{B_1^a, B_3, \widetilde{G}_{(q^1,\circ)}^1\}[2/1] = \{B_1^a, B_3, \widetilde{G}_{(q^1,\circ)}^2\}$ during the construction of $\hat{\mathcal{G}}^*$. We have also added the self-loop added by $\{(a, push_\varepsilon, G_{(q^1,\circ)}^1)\}[2/1] = \{(a, push_\varepsilon, G_{(q^1,\circ)}^2)\}$ 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 $A_0, A_1, \ldots$ 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 $\widetilde{G}_{(q_1,Q_2)}$. 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 $\widetilde{G}_{(q_1,Q_2)}^i$ is associated with a set that acts as a recipe for updating the 1-store automaton described by $\widetilde{G}_{(q_1,Q_2)}^{i-1}$ or creating a new automaton if $\widetilde{G}_{(q_1,Q_2)}^{i-1}$ 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 $\{S_1, \ldots, S_m\}$ represents an automaton

that accepts the union of the languages accepted by the automata described by $S_1, \ldots, S_m$. Each set $S \in \{S_1, \ldots, S_m\}$ corresponds to a possible effect of a command d at order-1 of the automaton. The automaton described by $S = \{\alpha_1, \ldots, \alpha_m\}$ accepts the intersection of languages described by its elements α_t ($t \in \{1, \ldots, m\}$).

An element that is an automaton B refers directly to the automaton B. Similarly, an identifier $\widetilde{G}^i_{(q_1, Q_2)}$ refers to its corresponding automaton. Finally, an element of the form $(a, push_w, \theta)$ refers to an automaton capturing the effect of applying the inverse of the $push_w$ command to the stacks accepted by the automaton represented by θ; moreover, the top_1 character of the stacks accepted by the new automaton will be a. It is a consequence of 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 $\widetilde{G}^i_{(q_1, Q_2)}$ we attach a subset of

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

where $\mathcal{B}$ is the 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 $\widetilde{G}^i_{(q,Q)}$ in A_i as $\widetilde{\mathcal{G}}^i$. The sets $\mathcal{B}$ and $\mathcal{O}_1$ are finite by definition. The size of the set $\widetilde{\mathcal{G}}^i$ for any i is finitely bound by the (fixed) state-set of A_i.

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

The automaton $\mathcal{G}^i$ is built inductively. We set $\mathcal{G}^0$ to be the disjoint union of all automata in $\mathcal{B}$. We define $\mathcal{G}^{i+1} = T_{\widetilde{\mathcal{G}}^{i+1}}(\mathcal{G}^i)$ where $T_{\widetilde{\mathcal{G}}^j}(\mathcal{G}^i)$ is given in Definition 3.1. In Section 3.4 it will be seen that j is not always $(i+1)$.

Definition 3.1 Given an automaton $\mathcal{G}^i = (\mathcal{Q}^i, \Sigma, \Delta^i, \_, \mathcal{Q}_f)$ and a set of identifiers (with associated sets) $\widetilde{\mathcal{G}}^j_1$, we define,

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

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

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

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

- If $\alpha_t = \theta$, then $(q^\theta \xrightarrow{b} Q_t) \in \Delta^i$.

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

There are two key parts to Definition 3.1. 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 $\mathcal{L}(A_i)$.

There are two different forms for the elements $\alpha_t \in \{\alpha_1, \ldots, \alpha_r\}$. 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 $T_{\widetilde{\mathcal{G}}^j}(\mathcal{G}^i)$. If α_t is of the form $(a, push_w, \theta)$, then it corresponds to the effects of a command $(p, a, \{\ldots, (push_w, p'), \ldots\})$. The new store must have the character a as its top_1 character, and

15

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 $A_0, A_1, \ldots$

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

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

where Δ^{i+1} is given below.

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

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

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

- If $o_t = push_2$, then $S_t = \{B_1^a\} \cup \widetilde{\theta}_1 \cup \widetilde{\theta}_2$ and there exists a path $q^{k_t} \xrightarrow{\widetilde{\theta}_1}_i Q' \xrightarrow{\widetilde{\theta}_2}_i Q_t$ in A_i.

- If $o_t = pop_2$, then $S_t = \{B_1^a\}$ 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}.

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

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

The above construction is similar to Definition 3.1. 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 $\widetilde{\mathcal{G}}^i$. A command $(p^j, a, \{(o_1, p^{k_1}), \ldots, (o_m, p^{k_m})\})$ takes the place of a set $\{\alpha_1, \ldots, \alpha_m\}$.

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.

When 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 $pop_2(\gamma w') = w'$), Q_t is the initial state of the automaton represented by θ and S_t contains the automaton B_1^a. This ensures that the top_1 character of the new store is a. We also need to consider the undefined store $\triangledown$. 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 $\triangledown$ 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 $A_0, A_1, \ldots$ which is sound and complete with respect to $Pre^*(C_{Init})$.

Property 3.1 *For any configuration $\langle p^j, \gamma \rangle$ it is the case that $\gamma \in \mathcal{L}(A_i^{q^j})$ for some i iff $\langle p^j, \gamma \rangle \in Pre^*(C_{Init})$.*

3.4 Constructing the Automaton A_*

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

Definition 3.3

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

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

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 3.4

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

2. If $\widetilde{G}^i_{(q_1, Q_2)} \lesssim \widetilde{G}^j_{(q_1, Q_2)}$ and $\widetilde{G}^j_{(q_1, Q_2)} \lesssim \widetilde{G}^i_{(q_1, Q_2)}$, then we write $\widetilde{G}^i_{(q_1, Q_2)} \simeq \widetilde{G}^j_{(q_1, Q_2)}$.

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

We now show that a fixed point is reached at order-2. That we reach a fixed point is important, since, when $\widetilde{\mathcal{G}}^i \simeq \widetilde{\mathcal{G}}^{i+1}$ there are two key consequences. Firstly, for all q_1 and Q_2, we have $\widetilde{G}^i_{(q_1, Q_2)} \in \widetilde{\mathcal{G}}^i$ iff we also have $\widetilde{G}^{i+1}_{(q_1, Q_2)} \in \widetilde{\mathcal{G}}^{i+1}$. 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 $\widetilde{G}^i_{(q_1, Q_2)} \simeq \widetilde{G}^{i+1}_{(q_1, Q_2)}$ 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 3.2 *There exists $i_1 > 0$ such that $\widetilde{\mathcal{G}}^i \simeq \widetilde{\mathcal{G}}^{i_1}$ for all $i \geq i_1$.*

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

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

Lemma 3.1 *Suppose we have constructed a sequence of automata $\mathcal{G}^0, \mathcal{G}^1, \ldots$ with the associated sets $\widetilde{\mathcal{G}}^0, \widetilde{\mathcal{G}}^1, \ldots$. Further, suppose there exists an i_1 such that for all $i \geq i_1$ we have $\widetilde{\mathcal{G}}^i \simeq \widetilde{\mathcal{G}}^{i_1}$. We can define a sequence of automata $\hat{\mathcal{G}}^{i_1}, \hat{\mathcal{G}}^{i_1+1}, \ldots$ such that the state-set in $\hat{\mathcal{G}}^i$ remains constant 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 $g^{i_1}_{(q,Q')} \xrightarrow{w}_i Q_1$ with $Q_1 \subseteq \mathcal{Q}_f$ exists in $\hat{\mathcal{G}}^i$ for some i.*

2. *The run $g^{i_1}_{(q,Q')} \xrightarrow{w}_{i_0} Q_2$ with $Q_2 \subseteq \mathcal{Q}_f$ exists in $\hat{\mathcal{G}}^{i_0}$.*

3. *The run $g^{i'}_{(q,Q')} \xrightarrow{w}_{i'} Q_3$ with $Q_3 \subseteq \mathcal{Q}_f$ exists in $\mathcal{G}^{i'}$ for some i'.*

Proof. Follows from the definition of $\hat{\mathcal{G}}^{i+1} = T_{\widetilde{\mathcal{G}}^{i_1}[i_1/i_1-1]}(\hat{\mathcal{G}}^i_1)$, Lemma B.9, Lemma B.10 and Lemma B.11. $\qquad\square$

We use $\hat{\mathcal{G}}^{i+1} = T_{\widetilde{\mathcal{G}}^{i_1}[i_1/i_1-1]}(\hat{\mathcal{G}}^i)$ to construct the sequence $\hat{\mathcal{G}}^{i_1}, \hat{\mathcal{G}}^{i_1+1}, \ldots$ (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 $\widetilde{\mathcal{G}}^{i_1}[i_1/i_1 - 1]$ makes the sets in $\widetilde{\mathcal{G}}^{i_1}$ 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 $g^*_{(q_1,Q_2)} = g^{i_1}_{(q_1,Q_2)}$ for each $g^{i_1}_{(q_1,Q_2)}$. Finally, we show that we can construct the automaton A_*.

Property 3.3 *There exists an automaton A_* which is sound and complete with respect to $A_0, A_1, \ldots$ and hence computes the set $Pre^*(C_{Init})$.*

Proof. By Property 3.2 there is some i_1 with $\widetilde{\mathcal{G}}^i \simeq \widetilde{\mathcal{G}}^{i_1}$ for all $i \geq i_1$. By Lemma 3.1, we have $\hat{\mathcal{G}}^* = \hat{\mathcal{G}}^{i_0}$. We then define A_* from A_{i_1} with each transition $q \longrightarrow_* Q'$ in A_* labelled with the appropriate $B \in \mathcal{B}$ or automaton $G^*_{(q,Q')}$ from $\hat{\mathcal{G}}^* = \hat{\mathcal{G}}^{i_0}$. $\qquad\square$

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 $\hat{\mathcal{G}}^{i+1} = T_{\widetilde{\mathcal{G}}^{i_1}_l[i_1/i_1-1]}(\hat{\mathcal{G}}^i)$ 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 $\hat{\mathcal{G}}^*$.

4   The General Case

We now generalise the order-2 algorithm to APDSs of all orders.

Theorem 4.1 *Given an n-store multi-automaton A accepting the set of configurations C_{Init} of an order-n APDS, we can construct in n-EXPTIME (in the size of A_0) an n-store multi-automaton A_* accepting the set $Pre^*(C_{Init})$. Thus, $Pre^*(C_{Init})$ is regular.*

18

In the order-2 case, we added transitions to the 2-store multi-automaton A_0 depending on the commands $\mathcal{D}$ of the given APDS. The (order-1 store-)automata labelling the transitions were updated through the addition of new initial states until the transition structure at order-2 reached a fixed point. We were then able to stop adding new states at order-1, adding only new transitions until a second fixed point was reached. At this point the algorithm terminated.

In the order-n case, new states are added to the automata labelling the order-n edges until the order-n transition structure reaches a fixed point. We are then able to stop adding new states to the order-$(n-1)$ automata, although we continue to add new states to the automata labelling the transitions at order-$(n-1)$. Since the order-$(n-1)$ state-set is fixed, the order-$(n-1)$ transition structure will reach a second fixed point. At this stage we are no longer required to add new states to order-$(n-2)$ of the automaton, and so on. In this way we have a cascade of fixed points from order-n of the automaton down to order-1, at which point the algorithm terminates.

4.1 Preliminaries

We generalise the definition of $G^i_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^i$ to any order l of the store-automata. That is, to each $\widetilde{G}^i_{(q_1,Q_2)}$ at order-l we attach a subset of

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

where $\mathcal{B}_{l-1}$ is the union of the set of all $(l-1)$-store automata occurring in A_0 and in automata of the form $B^a_{l'}$ or $X^a_{l'}$ for some l' (defined in Section 2.3). Further, we denote the set of all order-$(l-1)$ identifiers $\widetilde{G}^i_{(q_1,Q_2)}$ in A_i as $\widetilde{\mathcal{G}}^i_{l-1}$. The sets $\mathcal{B}_{l-1}$ and $\mathcal{O}_l$ are finite by definition. If the state-set at order-l is fixed, there is a finite bound on the size of the set $\widetilde{\mathcal{G}}^i_{l-1}$ for any i.

Similarly, we generalise the definition of the operation $T_{\widetilde{\mathcal{G}}^j_l}$. When $l = 1$, the definition is identical to Definition 3.1. When $l > 1$, the definition is similar to the definition of $T_{\mathcal{D}}$ in the order-2 case, except we do not have to consider the undefined store $\triangledown$ which only occurs at order-n of an n-store multi-automaton.

Definition 4.1 Given an automaton $\mathcal{G}^i_l = (\mathcal{Q}^i, \Sigma, \Delta^i, \_, \mathcal{Q}_f)$ and a set of identifiers $\widetilde{\mathcal{G}}^j_l$, we define,

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

where $\mathcal{Q}^{i+1} = \mathcal{Q}^i \cup \{ g^j_{(q_1,Q_2)} \mid \mathcal{G}^j_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^j_l \}$ and Δ^{i+1} depends on l:

Case $l = 1$. $\Delta^{i+1} = \Delta^{inherited} \cup \Delta^{derived} \cup \Delta^i$ and,

$$
\begin{aligned}
\Delta^{inherited} &= \{ g^j_{(q_1,Q_2)} \xrightarrow{a} Q \mid (g^{j-1}_{(q_1,Q_2)} \xrightarrow{a} Q) \in \Delta^i \} \\
\Delta^{derived} &= \left\{ g^j_{(q_1,Q_2)} \xrightarrow{b} Q \mid \begin{array}{l} \{\alpha_1, \ldots, \alpha_r\} \in \mathcal{G}^j_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^j_l \text{ and} \\ b \in \Sigma \text{ and } Q = Q_1 \cup \cdots \cup Q_r \text{ and (1)} \end{array} \right\}
\end{aligned}
$$

where (1) requires, for each $t \in \{1, \ldots, r\}$ we have,

- If $\alpha_t = \theta$, then $(q^\theta \xrightarrow{b} Q_t) \in \Delta^i$.

- If $\alpha_t = (a, push_w, \theta)$, then $b = a$ and $q^\theta \xrightarrow{w} Q_t$ is a run of $\mathcal{G}^i_1$.

Case $l > 1$. We begin by defining the set of labels $\widetilde{\mathcal{G}}^{i+1}_{l-1}$. This set contains labels on transitions from $\mathcal{G}^i_l$, labels on transitions from the new states $g^j_{(q_1,Q_2)}$ inherited from $g^{j-1}_{(q_1,Q_2)}$ and labels on transitions derived from $\widetilde{\mathcal{G}}^j_l$. That is,

$$
\widetilde{\mathcal{G}}^{i+1}_{l-1} = \left\{ \begin{array}{l} \left\{ \widetilde{G}^{i+1}_{(q,Q)} \mid (q \xrightarrow{\widetilde{G}^i_{(q,Q)}} Q) \in \Delta^i \right\} \cup \\[2mm] \left\{ \widetilde{G}^{i+1}_{(q,Q)} \mid q = g^j_{(q_1,Q_2)} \text{ and } (g^{j-1}_{(q_1,Q_2)} \xrightarrow{B} Q) \in \Delta^i \right\} \cup \\[2mm] \left\{ \widetilde{G}^{i+1}_{(q,Q)} \mid q = g^j_{(q_1,Q_2)} \text{ and } \widetilde{G}^j_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^j_l \text{ and (2)} \right\} \end{array} \right.
$$

19

The contents of the associated sets $\widetilde{G}^{i+1}_{(q,Q)} \in \widetilde{\mathcal{G}}^{i+1}_{l-1}$ are defined as follows,

$$
\begin{aligned}
\widetilde{G}^{i+1}_{(q,Q)} &= \quad \emptyset \text{ if } q \neq g^j_{(q_1,Q_2)} \\
\widetilde{G}^{i+1}_{(g^j_{(q_1,Q_2)},Q)} &= \quad \{\, S \mid (2)\,\} \cup \left\{\, \{B\} \mid (g^{j-1}_{(q_1,Q_2)} \xrightarrow{B} Q) \in \Delta^i \,\right\}
\end{aligned}
$$

where (2) requires $\{\alpha_1, \ldots, \alpha_r\} \in \widetilde{G}^j_{(q_1,Q_2)}$, $Q = Q_1 \cup \cdots \cup Q_r$, $S = S_1 \cup \cdots \cup S_r$ and for each $t \in \{1, \ldots, r\}$ we have,

- If $\alpha_t = \theta$, then $(q^\theta \xrightarrow{B} Q_t) \in \Delta^i$ and $S_t = \{B\}$.

- If $\alpha_t = (a, push_l, \theta)$, then $q^\theta \xrightarrow{\widetilde{B}_1} Q^1 \xrightarrow{\widetilde{B}_2} Q_t$ is a path in $\mathcal{G}^i_l$ and $S_t = \{B^a_{l-1}\} \cup \widetilde{B}_1 \cup \widetilde{B}_2$.

- If $\alpha_t = (a, pop_l, \theta)$, then $Q_t = \{q^\theta\}$ and $S_t = \{B^a_{l-1}\}$.

- If $\theta = (a, o, \theta)$ when $\ell(o) < l$, then $q^\theta \xrightarrow{B} Q_t \in \Delta^i$ and $S_t = \{(a, o, B)\}$

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

$$
\Delta^{i+1} = \begin{cases}
\left\{\, q \xrightarrow{B} Q \mid (q \xrightarrow{B} Q) \in \Delta^i \text{ and } B \in \mathcal{B}_{l-1} \,\right\} \cup \\[2mm]
\left\{\, q \xrightarrow{\widetilde{G}^{i+1}_{(q,Q)}} Q \mid \widetilde{G}^{i+1}_{(q,Q)} \in \widetilde{\mathcal{G}}^{i+1}_{l-1} \,\right\}
\end{cases}
$$

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

4.2 Further Examples

We have given an outline of a fully worked example at order-two. However, there are many situations that this example does not illuminate. In this section we give a series of short examples showing the introduction of transitions to $\mathcal{G}^i_l$ from elements in $\widetilde{G}^i_{(q,Q')}$ not covered in the previous case.

4.2.1   Case $\{(a, push_{abcd}, \widetilde{G}^{i-1}_{(q_1,Q_1)})\} \in \widetilde{G}^i_{(q,Q)} \in \widetilde{\mathcal{G}}^i_1$

Suppose Figure 9 represents a sub-automaton of $\mathcal{G}^{i-1}_1$. There is one run from the state $g^{i-1}_{(q_1,Q_1)}$ over the word $abcd$. In particular we have,

$$
g^{i-1}_{(q_1,Q_1)} \xrightarrow{abcd}_{i-1} \{q_3, q_4\}
$$

Therefore we add the transition,

$$
g^i_{(q,Q)} \xrightarrow{a}_i \{q_3, q_4\}
$$

to $\mathcal{G}^i_1$.

4.2.2   Case $\{(a, pop_l, \widetilde{G}^{i-1}_{(q_1,Q_1)})\} \in \widetilde{G}^i_{(q,Q)} \in \widetilde{\mathcal{G}}^i_l$

Figure 10 shows the relevant subsection of $\mathcal{G}^i_l$ after the pop_l has been processed. Let $G = g^{i-1}_{(q_1,Q_1)}$. We have $\{B^a_{l-1}\}$ in $\widetilde{G}^i_{(g^i_{(q,Q)},G)}$.
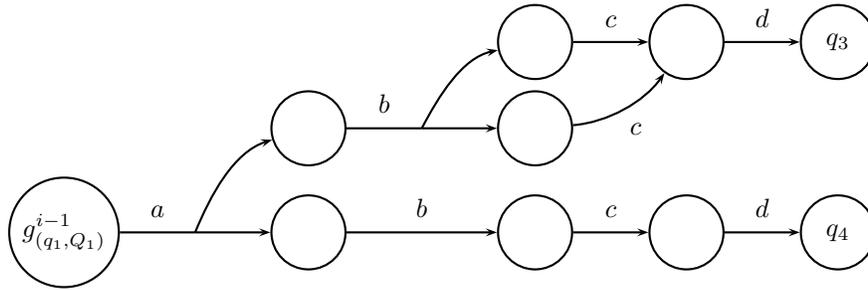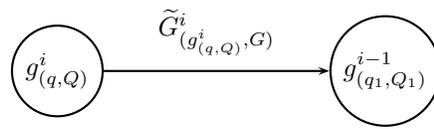
Figure 9: An example for $push_{abcd}$.

Figure 10: An example for pop_l.
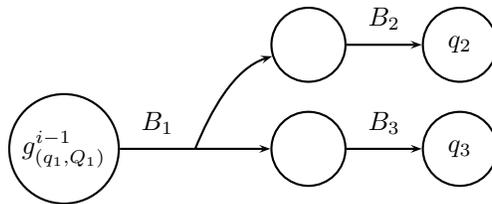
Figure 11: An example for $push_l$.
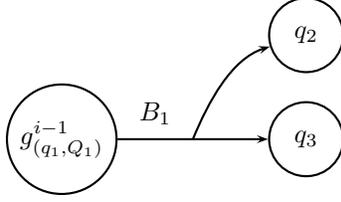
Figure 12: An example for $\ell(o) < l$.

4.2.3 Case $\{(a, push_l, \widetilde{G}^{i-1}_{(q_1,Q_1)})\} \in \widetilde{G}^{i}_{(q,Q)} \in \widetilde{\mathcal{G}}^{i}_{l}$

Suppose Figure 11 shows a sub-automaton of $\mathcal{G}^{i-1}_{l}$, where $B_1, B_2, B_3 \in \mathcal{B}_{l-1} \cup \widetilde{\mathcal{G}}^{i-1}_{l-1}$. There is one possible value for Q'' when enumerating over runs of the form,

$$\{g^{i-1}_{(q_1,Q_1)}\} \xrightarrow{\widetilde{B}_1}_{i-1} Q' \xrightarrow{\widetilde{B}_2}_{i-1} Q''$$

In particular $Q'' = \{q_2, q_3\}$. We have $\widetilde{B}_1 = \{B_1\}$ and $\widetilde{B}_2 = \{B_2, B_3\}$. Consequently, we ensure that the transition,

$$g^{i}_{(q,Q)} \xrightarrow{\widetilde{G}^{i}_{(g^{i}_{(q,Q)},\{q_2,q_3\})}} \{q_2, q_3\}$$

exists, and that $\{B^a_{l-1}, B_1, B_2, B_3\} \in \widetilde{G}^{i}_{(g^{i}_{(q,Q)},\{q_2,q_3\})}$.

4.2.4 Case $\{(a, o, G^{i-1}_{(q_1,Q_1)})\} \in \widetilde{G}^{i}_{(q,Q)} \in \widetilde{\mathcal{G}}^{i}_{l}$ and $\ell(o) < l$

Suppose Figure 12 shows a sub-automaton of $\mathcal{G}^{i-1}_{l}$, where $B_1 \in \mathcal{B}_{l-1} \cup \widetilde{\mathcal{G}}^{i-1}_{l-1}$. There is one possible value for Q' when enumerating over runs of the form,

$$\{g^{i-1}_{(q_1,Q_1)}\} \xrightarrow{\widetilde{B}_1}_{i-1} Q'$$

In particular $Q' = \{q_2, q_3\}$. Consequently we ensure that the transition,

$$g^{i}_{(q,Q)} \xrightarrow{\widetilde{G}^{i}_{(g^{i}_{(q,Q)},\{q_2,q_3\})}} \{q_2, q_3\}$$

exists, and that $\{(a, o, B_1)\} \in \widetilde{G}^{i}_{(g^{i}_{(q,Q)},\{q_2,q_3\})}$.

4.2.5 Case $\{(a, push_l, \widetilde{G}^{i-1}_{(q_1,Q_1)}), \widetilde{G}^{i-1}_{(q_2,Q_2)}\} \in \widetilde{G}^{i}_{(q,Q)} \in \widetilde{\mathcal{G}}^{i}_{l}$

Suppose Figure 13 shows a sub-automaton of $\mathcal{G}^{i-1}_{l}$, where $B_1, B_2, B_3, B_4 \in \mathcal{B}_{l-1} \cup \widetilde{\mathcal{G}}^{i-1}_{l-1}$. There is one possible value for $S = S_1 \cup S_2$ and $Q' = Q'_1 \cup Q'_2$. That is $S_1 = \{B^a_{l-1}, B_1, B_2, B_3\}$ and $Q'_1 = \{q_3, q_4\}$ (as in the previous $push_l$ example); and $S_2 = \{B_4\}$ and $Q'_2 = \{q_5, q_6\}$.

Therefore, we ensure that the transition,

$$g^{i}_{(q,Q)} \xrightarrow{\widetilde{G}^{i}_{(g^{i}_{(q,Q)},Q')}} Q'$$

exists, and that $S \in \widetilde{G}^{i}_{(g^{i}_{(q,Q)},Q')}$.

22
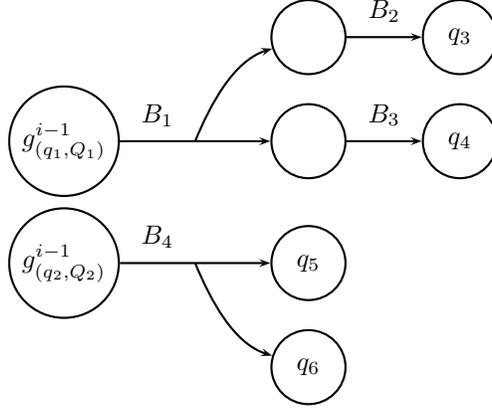
Figure 13: An example for a two-element S.

4.3 Constructing $A_0, A_1, \ldots$

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

Definition 4.2 Given an automaton $A_i = (\mathcal{Q}, \Sigma, \Delta^i, \{q^1, \ldots, q^z\}, \mathcal{Q}_f)$ and a set of commands $\mathcal{D}$, we define,

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

where Δ^{i+1} is given below.

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

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

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

- If $o_t = push_n$, then $S_t = \{B_{n-1}^a\} \cup \widetilde{\theta}_1 \cup \widetilde{\theta}_2$ and there exists a path,

$$q^{k_t} \xrightarrow{\widetilde{\theta}_1}_i Q' \xrightarrow{\widetilde{\theta}_2}_i Q_t$$

  in A_i.

- If $o_t = pop_n$, then $S_t = \{B_{n-1}^a\}$ and $Q_t = \{q^{k_t}\}$. Or, if $q^j \xrightarrow{\nabla}_i \{q_f^\varepsilon\}$ exists in A_i, we may have $S_t = \{B_{n-1}^a\}$ and $Q_t = \{q_f^\varepsilon\}$.

- If $o_t = push_w$ or $o_t = push_l$ for $l < n$, then $S_t = \{(a, o, \theta)\}$ and there exists a transition $q^{k_t} \xrightarrow{\theta}_i Q_t$ in A_i.

- If $o_t = pop_l$ for $l < n$, then $S_t = \{(a, o, \theta)\}$ and there exists a transition $q^{k_t} \xrightarrow{\theta}_i Q_t$ in A_i. Or, if $q^j \xrightarrow{\nabla}_i \{q_f^\varepsilon\}$ exists in A_i, we may have $S_t = \{X_l^a\}$ and $Q_t = \{q_f^*\}$.

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

$$\Delta^{i+1} = \left\{ q \xrightarrow{B} Q \;\middle|\; \begin{array}{c} (q \xrightarrow{B} Q) \in \Delta^i \\ \text{and } B \in \mathcal{B} \end{array} \right\} \cup \left\{ q \xrightarrow{\widetilde{G}_{(q,Q)}^{i+1}} Q \;\middle|\; \widetilde{G}_{(q,Q)}^{i+1} \in \widetilde{\mathcal{G}}_{n-1}^{i+1} \right\}$$

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

Our construction is sound and complete. The proofs for this result are given in Appendix B.4.

Property 4.1 *For any configuration $\langle p^j, \gamma \rangle$ it is the case that $\gamma \in \mathcal{L}(A^{q^j}_i)$ for some i iff $\langle p^j, \gamma \rangle \in Pre^*(C_{Init})$.*

Proof. The property follows from Property B.3 and Property B.4. □

4.4 Constructing A_*

We have shown how to construct a sequence of automata $A_0, A_1, \ldots$ which is sound and complete with respect to $Pre^*(\mathcal{L}(A_0))$. In this section we show how to compute an automaton A_* that is a finite representation of this sequence. That is, $\mathcal{L}(A_*) = Pre^*(\mathcal{L}(A_0))$. We begin by defining $\lesssim$, generalising the notion of subset modulo i for the sets $\widetilde{G}^i_{(q,Q')}$.

Definition 4.3

1. Given $\theta \in \mathcal{B}_l \cup \widetilde{\mathcal{G}}^i_l$ for some i and l, let

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

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 4.4

1. We write $\widetilde{G}^i_{(q,Q')} \lesssim \widetilde{G}^j_{(q,Q')}$ iff for each $S \in \widetilde{G}^i_{(q,Q')}$ we have $S[j-1/i-1] \in \widetilde{G}^j_{(q,Q')}$.

2. If $\widetilde{G}^i_{(q,Q')} \lesssim \widetilde{G}^j_{(q,Q')}$ and $\widetilde{G}^j_{(q,Q')} \lesssim \widetilde{G}^i_{(q,Q')}$, then we write $\widetilde{G}^i_{(q,Q')} \simeq \widetilde{G}^j_{(q,Q')}$.

3. Furthermore, we extend the notation to sets. That is, $\widetilde{\mathcal{G}}^i_l \lesssim \widetilde{\mathcal{G}}^j_l$ iff for all $\widetilde{G}^i_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^i_l$ we have $\widetilde{G}^j_{(q_1,Q_2)} \in \widetilde{\mathcal{G}}^j_l$ and $\widetilde{G}^i_{(q,Q')} \lesssim \widetilde{G}^j_{(q,Q')}$.

A fixed point is reached as in the order-2 case: eventually, no more transitions are added at order-n of the automaton, allowing us to fix the state set at order-$(n-1)$. This begins a cascade of fixed points which ends at order-1, resulting in termination.

Property 4.2 *There exists $i_{n-1} > 0$ such that $\widetilde{\mathcal{G}}^i_{n-1} \simeq \widetilde{\mathcal{G}}^{i_{n-1}}_{n-1}$ for all $i \geq i_{n-1}$.*

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

Once a fixed point has been reached at order-$(l+1)$, we can fix the state-set at order-l.

Lemma 4.1 *Suppose we have constructed a sequence of automata $\mathcal{G}_l^0, \mathcal{G}_l^1, \ldots$ and associated sets $\widetilde{\mathcal{G}}_l^0, \widetilde{\mathcal{G}}_l^1, \ldots$. Further, suppose there exists an i_l such that for all $i \geq i_l$ we have $\widetilde{\mathcal{G}}_l^i \simeq \widetilde{\mathcal{G}}_l^{i_l}$. We can define a sequence of automata $\hat{\mathcal{G}}_l^{i_l}, \hat{\mathcal{G}}_l^{i_l+1}, \ldots$ such that the state-set in $\hat{\mathcal{G}}_l^i$ remains constant (although the automata labelling the transitions may gain states when $l > 1$). The following are equivalent for all w,*

1. *The run $g_{(q,Q')}^{i_l} \xrightarrow{w}_i Q$ with $Q \subseteq \mathcal{Q}_f$ exists in $\hat{\mathcal{G}}_l^i$ for some i.*

2. *The run $g_{(q,Q')}^{i'} \xrightarrow{w}_{i'} Q''$ with $Q'' \subseteq \mathcal{Q}_f$ exists in $\mathcal{G}_l^{i'}$ for some i'.*

Again, we use $\hat{\mathcal{G}}^{i+1} = T_{\widetilde{\mathcal{G}}_l^{i_l}[i_l/i_1-l]}(\hat{\mathcal{G}}_l^i)$ to construct the sequence $\hat{\mathcal{G}}_l^{i_l}, \hat{\mathcal{G}}_l^{i_l+1}, \ldots$ with $\hat{\mathcal{G}}_l^{i_l} = \mathcal{G}_l^{i_l}$. Soundness and completeness are shown in Appendix B.5. Once the state-set has been fixed at order-l, we will eventually reach another fixed point. In this way the fixed points cascade.

Property 4.3 *For a sequence of automata $\mathcal{G}_l^0, \mathcal{G}_l^1, \ldots$ such that the state-set at order-l of $\mathcal{G}_l^i$ remains constant there exists $i_{l-1} > 0$ such that $\widetilde{\mathcal{G}}_{l-1}^i \simeq \widetilde{\mathcal{G}}_{l-1}^{i_{l-1}}$ for all $i \geq i_{l-1}$.*

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

When the state-set has been fixed at order-1, the finiteness of Σ dictates that only a finite number of transitions can be added before saturation is reached. At this point any updates to the automata at order-1 will not change the set of accepted 1-stores. Termination at order-1 implies termination at order-2, and so on down to order-n. That is, for any l, we are able to define an automaton $\mathcal{G}_l^*$ which finitely represents the infinite sequence $\mathcal{G}_l^0, \mathcal{G}_l^1, \ldots$ used to construct $A_0, A_1, \ldots$.

Lemma 4.2 *Suppose we have constructed a sequence of automata $\mathcal{G}_l^0, \mathcal{G}_l^1, \ldots$ and associated sets $\widetilde{\mathcal{G}}_l^0, \widetilde{\mathcal{G}}_l^1, \ldots$. Further, suppose there exists an i_l such that for all $i \geq i_l$ we have $\widetilde{\mathcal{G}}_l^i \simeq \widetilde{\mathcal{G}}_l^{i_l}$. We can define an automaton $\mathcal{G}_l^*$ such that the following are equivalent for all w,*

1. *The run $g_{(q,Q')}^* \xrightarrow{w}_* Q$ with $Q \subseteq \mathcal{Q}_f$ exists in $\mathcal{G}_l^*$.*

2. *The run $g_{(q,Q')}^i \xrightarrow{w}_i Q''$ with $Q'' \subseteq \mathcal{Q}_f$ exists in $\mathcal{G}_l^i$ for some i.*

Proof. The complete proof is given in Appendix B.5.3. We show here how to construct $\mathcal{G}_l^*$.

We proceed by induction over l. In the base case $l = 1$ and the result follows because the alphabet is finite: there is a finite bound on the number of new transitions that can be added. Let $\mathcal{G}^{i_0}$ denote this saturation point. We define $\mathcal{G}_1^* = \mathcal{G}_1^{i_0}$ letting $g_{(q,Q)}^* = g_{(q,Q)}^{i_1}$ for each $g_{(q,Q)}^{i_1}$.

When $l > 1$ we generate the sequence $\hat{\mathcal{G}}_l^{i_l}, \hat{\mathcal{G}}_l^{i_l+1}, \ldots$ by Lemma 4.1. Since the state-set remains constant, it follows from Property 4.3 that there is some i_{l-1} with $\widetilde{G}_{l-1}^i \simeq \widetilde{G}_{l-1}^{i_{l-1}}$ for all $i \geq i_{l-1}$.

By induction, we have $\mathcal{G}_{l-1}^*$. We then define $\mathcal{G}_l^*$ from $\hat{\mathcal{G}}_l^{i_{l-1}}$ with $g_{(q,Q')}^* = g_{(q,Q')}^{i_l}$ for all q, Q' and each transition $q \longrightarrow_* Q'$ in $\mathcal{G}_l^*$ labelled with the appropriate $B \in \mathcal{B}_{n-1}$ or automaton $G_{(q,Q')}^*$ from $\mathcal{G}_{l-1}^*$. $\qquad\square$

Finally, 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 $A_{i_{n-1}}$ is reached.

2. For $l = n - 1$ down to $l = 1$: iterate $\mathcal{G}_l^{i+1} = T_{\widetilde{\mathcal{G}}_l^{i_l}[i_l/i_l-1]}(\mathcal{G}_l^i)$ to generate the fixed point $\mathcal{G}_l^{i_{l-1}}$ from $\mathcal{G}_l^{i_l}$.

3. For $l = 1$ to $l = n - 1$: construct G_l^* as in Lemma 4.2.

4. Construct A_* as in Property 4.4.

Property 4.4 *There exists an automaton A_* which is sound and complete with respect to $A_0, A_1, \ldots$ and hence computes the set $Pre^*(C_{Init})$.*

Proof. By Property 4.2 there is some i_{n-1} with $\widetilde{\mathcal{G}}_{n-1}^i \simeq \widetilde{\mathcal{G}}_{n-1}^{i_{n-1}}$ for all $i \geq i_{n-1}$. By Lemma 4.2, we have $\mathcal{G}_{n-1}^*$. We then define A_* from $A_{i_{n-1}}$ with each transition $q \longrightarrow_* Q'$ in A_* labelled with the automaton $G_{(q,Q')}^*$ from $\mathcal{G}_{n-1}^*$. □

4.5 Complexity

We claim our algorithm runs in n-EXPTIME. We define,

$$exp_1(m) = 2^m \qquad \text{and} \qquad exp_l(m) = 2^{exp_{l-1}(m)}$$

Furthermore, let $\mathcal{B} = \bigcup_{1 \leq l < n} \mathcal{B}_l$.

The algorithm requires the construction of an automaton $\mathcal{G}_l^*$ from $\mathcal{G}_l^{i_l}$ for $1 \leq l \leq n-1$ and fixed point i_l. Because the construction of $\mathcal{G}_l^*$ follows the same pattern as the construction of $\mathcal{G}_l^{i+1}$ from $\mathcal{G}_l^i$ but with at least as many states, it follows that it dominates the complexity of the algorithm. We calculate the complexity inductively, beginning at order-one with a fixed state-set. Ultimately, the induction reaches order-n, where the state-set is fixed from the start of the algorithm.

Let $|\mathcal{Q}|$ be the number of states in $\mathcal{G}_l^{i_l}$. When $l = 1$ we can add at most $\mathcal{O}(|\mathcal{Q}| \times 2^{|\mathcal{Q}|} \times |\Sigma|)$ transitions to $\mathcal{G}_1^{i_1}$ to define $\mathcal{G}_1^*$, where $\mathcal{Q}$ is the state-set of $\mathcal{G}_1^{i_1}$. This means we need at most $\mathcal{O}(2^{|\mathcal{Q}|})$ iterations before a fixed point is reached. Each iteration requires the processing of up to $\mathcal{O}(|\mathcal{Q}|)$ sets $\widetilde{G}_{(q_1,Q_2)}$ (one for each state $g_{(q_1,Q_2)}^{i_1}$). To process each set we need to process up to $\mathcal{O}(2^{|\mathcal{Q}|+|\mathcal{B}|})$ sets S. This step in turn requires the analysis of $\mathcal{O}(|\mathcal{Q}| + |\mathcal{B}|)$ elements. The most expensive operation is to enumerate all runs over a word for a *push* command. To enumerate all runs of the form $Q_1 \xrightarrow{w} Q_2$ for some w (from a $push_w$ command) we require time $\mathcal{O}(2^{|\mathcal{Q}|})$ (Proposition B.1). Therefore, to construct $\mathcal{G}_1^*$ from $\mathcal{G}^{i_1}$ takes $\mathcal{O}(|\mathcal{Q}| \times (|\mathcal{B}| + |\mathcal{Q}|) \times 2^{|\mathcal{B}|+3\times|\mathcal{Q}|})$. That is, $\mathcal{O}(exp_1(|\mathcal{Q}|) \times 2^{|\mathcal{B}|})$.

When $l > 1$ and the state-set $\mathcal{Q}$ of $\mathcal{G}_l^{i_l}$ is frozen, we add at most $\mathcal{O}(|\mathcal{Q}| \times 2^{|\mathcal{Q}|})$ transitions to $\mathcal{G}_l^{i_l}$ during the construction of $\mathcal{G}_l^*$. This means we need at most $\mathcal{O}(2^{|\mathcal{Q}|})$ iterations before a fixed point is reached. Each iteration requires the processing of up to $\mathcal{O}(|\mathcal{Q}|)$ sets $\widetilde{G}_{(q_1,Q_2)}$ (one for each state $g_{(q_1,Q_2)}^{i_1}$). To process each set we need to process up to $\mathcal{O}(2^{|\mathcal{Q}|+|\mathcal{B}|})$ sets S. This step in turn requires the analysis of $\mathcal{O}(|\mathcal{Q}| + |\mathcal{B}|)$ elements. The most expensive operation is to enumerate all runs of two transitions for a *push* command. To enumerate all runs of the form $Q_1 \xrightarrow{\widetilde{B_1}} Q_2 \xrightarrow{\widetilde{B_2}} Q_3$ requires time $\mathcal{O}(2^{(|\Delta|+|\mathcal{Q}|)})$ (by Proposition B.2) Since $|\Delta|$ is $\mathcal{O}(2^{|\mathcal{Q}|})$, the run enumeration requires time $\mathcal{O}(exp_2(|\mathcal{Q}|))$.

Since the contents of the sets $\widetilde{G}_{(q,Q')}^i \in \widetilde{\mathcal{G}}_{l-1}^i$ are derived from the transition structure, we will reach a fixed point one step after we have reached a point where no more transitions are added. During this step we will add $\mathcal{O}(2^{|\mathcal{Q}|})$ states to $\mathcal{G}_{l-1}^{i_l}$ to construct $\mathcal{G}_{l-1}^{i_{l-1}}$ (since there are $\mathcal{O}(2^{|\mathcal{Q}|})$ new transitions added in $\mathcal{O}(2^{|\mathcal{Q}|})$ steps). By induction, the construction of $\mathcal{G}_{l-1}^{i+1}$ from $\mathcal{G}_{l-1}^i$ performed at the end of each iteration takes $\mathcal{O}(exp_{(l-1)}(2^{|\mathcal{Q}|}) \times 2^{|\mathcal{B}|})$ time.

Hence, to reach a fixed point we require time $\mathcal{O}(2^{|\mathcal{Q}|} \times ((|\mathcal{Q}| + |\mathcal{B}|) \times 2^{|\mathcal{Q}|+|\mathcal{B}|} \times exp_2(|\mathcal{Q}|) + (exp_l(|\mathcal{Q}|) \times 2^{|\mathcal{B}|})))$. That is $\mathcal{O}(exp_l(|\mathcal{Q}|) \times 2^{|\mathcal{B}|})$. By induction we can construct $\mathcal{G}_{l-1}^*$ in time $\mathcal{O}(exp_{(l-1)}(2^{|\mathcal{Q}|}) \times 2^{|\mathcal{B}|})$. Hence, we can construct $\mathcal{G}_l^*$ in time $\mathcal{O}(exp_l(|\mathcal{Q}|) \times 2^{|\mathcal{B}|})$.

Finally, at order-n we can add at most $\mathcal{O}(2^{|\mathcal{Q}|})$ new transitions before a fixed point is reached. Hence we must perform $\mathcal{O}(2^{|\mathcal{Q}|})$ iterations. Each iteration requires us to process $\mathcal{O}(2^{|\mathcal{Q}|})$ commands (since $|\mathcal{Q}|$ is larger than the number of control states in the higher-order APDS). In processing each command we must process $\mathcal{O}(|\mathcal{Q}|)$ pairs of the form (o, p). The most expensive of which requires run enumeration that takes $\mathcal{O}(exp_2(|\mathcal{Q}|))$ time. Hence we construct $\mathcal{G}_{n-1}^{i_{n-1}}$ with $\mathcal{O}(2^{|\mathcal{Q}|})$ states in $\mathcal{O}(2^{|\mathcal{Q}|} \times (2^{|\mathcal{Q}|} \times |\mathcal{Q}| \times exp_2(|\mathcal{Q}|) + (exp_{(n-1)}(2^{|\mathcal{Q}|}) \times 2^{|\mathcal{B}|})))$ time. Thus, $\mathcal{G}_{n-1}^*$ and hence A_* can be constructed in time $\mathcal{O}(exp_{(n-1)}(2^{|\mathcal{Q}|}) \times 2^{|\mathcal{B}|})$, that is $\mathcal{O}(exp_n(|\mathcal{Q}|) \times 2^{|\mathcal{B}|})$. The algorithm runs in n-EXPTIME.

5 Applications

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

5.1 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 $\Lambda : \mathcal{P} \to 2^{Prop}$ 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 μTL, 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 $[^1 a]^1$ 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 5.1 *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. $c \xhookrightarrow{*} \langle p^j, \gamma_3 \rangle$ for some γ_3 with $top_1(\gamma_3) = a$, and

2. $\langle p^j, [^n a]^n \rangle \xhookrightarrow{*} \langle p^f, \gamma_1 \rangle \xhookrightarrow{*} \langle p^j, \gamma_2 \rangle$

Proof. See Appendix C. □

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 5.1 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,

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

Lemma 5.1 *There exists a run $\langle (p, 0), [^n a]^n \rangle \overset{*}{\hookrightarrow} \langle (p, 1), w' \rangle$ with $w' \in \mathcal{L}(B_n^a)$ in BP' iff $\langle p, [^n a]^n \rangle$ satisfies (2).*

Proof. See Appendix C.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). $\qquad\square$

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 5.1 *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 5.1 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. $\qquad\square$

Corollary 5.1 *Given an order-n PDS $(\mathcal{P}, \mathcal{D}, \Sigma)$ with a labelling function $\Lambda : \mathcal{P} \to 2^{Prop}$ 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. $\qquad\square$

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.

5.2 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 5.2 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 5.3 Given an order-n PRG $(\mathcal{P}, \mathcal{D}, \Sigma, \mathcal{R})$ we define an order-n APDS $(\mathcal{P}, \mathcal{D}', \Sigma)$ where,

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

Furthermore, let R_{stuck} be the set of configurations $\langle p, \triangledown \rangle$ 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 $C_A^{\triangledown}$ be the set of order-n configurations with an undefined stack and a control state belonging to Abelard.

Theorem 5.2 *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 $Pre^*(\mathcal{R} \cup \mathcal{R}_{stuck}) \setminus C_A^{\triangledown}$. Hence, computing the winning regions in the order-n PRG is n-EXPTIME.*

5.3 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.

5.3.1 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 5.4 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 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 $\Lambda : \mathcal{P} \to 2^{Prop}$, and a valuation function $\mathcal{V}$ assigning a set of configurations to each variable $X \in \chi$, the set of configurations $[\![\phi]\!]_{\mathcal{V}}$ satisfying ϕ is defined,

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

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

5.3.2 Model-Checking the Alternation-Free μ-Calculus

Given an order-n PDS $(\mathcal{P}, \mathcal{D}, \Sigma)$ with a labelling function $\Lambda : \mathcal{P} \to 2^{Prop}$, 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 $[\![\phi]\!]_{\mathcal{V}}$.

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 the 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 $(p, \psi) \in \mathcal{P}_E^{(\mathcal{P}, \phi)}$ and $((p, \psi), a, push_a, (p, \psi_1)), ((p, \psi), a, push_a, (p, \psi_2)) \in \mathcal{D}_\mathcal{P}^\phi$,

- If $\psi = \psi_1 \wedge \psi_2$, then $(p, \psi) \in \mathcal{P}_A^{(\mathcal{P}, \phi)}$ and $((p, \psi), a, push_a, (p, \psi_1)), ((p, \psi), a, push_a, (p, \psi_2)) \in \mathcal{D}_\mathcal{P}^\phi$,

- 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 $(p, a, o, p') \in \mathcal{D}$ we have $((p, \psi), a, o, (p', \psi')) \in \mathcal{D}_\mathcal{P}^\phi$.

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 $\bot \in \Sigma$. This symbol marks the bottom of all order-one stacks and is never pushed or popped, except in the case of a command $(_-, \bot, push_\bot, _-)$. The use of such a symbol is common throughout the literature [13, 28, 25] etc..

Proposition 5.1 *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 of the alternation-free μ-calculus ϕ such that all σ-sub-formulae of ϕ are μ-sub-formulae, we have $\langle p, \gamma \rangle \in [\![\phi]\!]_\mathcal{V}$ iff $\langle (p, \phi), \gamma \rangle \in Attr_E(\mathcal{R})$.*

Proof. The result follows from the fundamental theorem of the propositional μ-calculus [23, 15]. 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 Section 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 $i \in \{1, \ldots, m\}$. The following proposition is taken directly from [2]:

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

$$[\![\phi]\!]_{\mathcal{V}} = [\![\phi[U/\Phi]]\!]_{\mathcal{V}'}$$

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

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

Theorem 5.3 *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 $\mathcal{L}(A) = [\![\phi]\!]_{\mathcal{V}}$.*

5.3.3 Complexity

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 $[\![\psi]\!]_{\mathcal{V}}$ 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 $[\![\psi]\!]_{\mathcal{V}'}$ for an internal node ψ with children $\phi_1, \ldots, \phi_m$ has $\mathcal{O}(exp_n(\Sigma_{i=1}^m n_i + n_\mathcal{R}) \times 2^{b_i})$ states, where n_i is the size of the automaton recognising $[\![\phi_i]\!]_{\mathcal{V}_i}$ for $i \in \{1, \ldots, m\}$ and b_i is the size of $\mathcal{B}$ for that automaton. Due to the final complementation step, $|\mathcal{B}|$ may be of size $\mathcal{O}(exp_{n+1}(\Sigma_{i=1}^m n_i + n_\mathcal{R}))$, which is also the total time required.

Subsequently, the automaton A recognising $[\![\phi]\!]_{\mathcal{V}'}$ 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.

5.4 Non-emptiness of Higher-Order Pushdown Automata

We show that the n-EXPTIME complexity of the algorithm is optimal. In fact, the backwards-reachability problem for order-n PDSs is n-EXPTIME-complete. This result is widely regarded to follow from the work of Engelfriet [10]. However, because Engelfriet considers a broad range of automata, it is not immediately clear that his work can be applied directly to our own. We provide another proof of the result which uses a clearly stated theorem of Engelfriet: the non-emptiness problem for (non-deterministic) order-n pushdown automata is $(n - 1)$-EXPTIME-complete [10][2]

Walukiewicz and Cachat have provided another proof of this property [26]. Initially this proof was not published due to Engelfriet's result. The following proof was constructed before their paper was made available. The proof strategy is due to Olivier Serre.

We show that the reachability problem for order-n APDS is n-EXPTIME-hard via a polynomial reduction from the non-emptiness problem over non-alternating order-$(n+1)$ pushdown automata. Let $\bot \in \Sigma$ be a dedicated "bottom of the stack" symbol that is neither popped from nor pushed onto the stack. We define $\bot_1 = [\bot]$ and $\bot_n = [\bot_{n-1}]$. In this case, the initial configuration of the automaton is of the form $\langle p_0, \bot_n \rangle$ for some p_0.

---

[2]Completeness follows from Theorem 2.6, which states that 2N(multi)-$P^k = (k - 1)$-EXPTIME, and Theorem 7.11, which can be instantiated to show that one-way non-deterministic order-k pushdown automata are log-space complete in 2N(multi)-P^k.

We define the order-n PRG P_G which can be used to determine whether the order-$(n + 1)$ pushdown automaton P is non-empty. Note that the construction is polynomial in size. Also, observe that commands of the form $(p, a, push_a, p')$ leave the stack contents unchanged.

Definition 5.5 Given an order-$(n + 1)$ pushdown automaton $P = (\mathcal{P}, \mathcal{D}, \Sigma, \Gamma, p_0, \mathcal{P}_f)$ we define the order-n PRG $P_G = (\mathcal{P}', \mathcal{D}', \Sigma, \mathcal{R})$ where,

$$
\begin{aligned}
\mathcal{P}' &= \mathcal{P}'_E \cup \mathcal{P}'_A \\
\mathcal{P}'_E &= \mathcal{P} \times (\mathcal{P} \cup \{\otimes\}) \ \cup \ \{\mathtt{ff}\} \\
\mathcal{P}'_A &= \mathcal{P} \times (\mathcal{P} \cup \{\otimes\}) \times (\mathcal{P} \cup \{\otimes\}) \ \cup \ \{\mathtt{tt}\}
\end{aligned}
$$

with $\mathtt{tt}, \mathtt{ff}, \otimes \notin \mathcal{P}$. Furthermore,

$$
\mathcal{R} = (\mathcal{P}_f \times \{\otimes\} \times C_n^\Sigma)
$$

Finally,

$$
\begin{aligned}
\mathcal{D}' = \ &\{ \ ((p, p_r), a, o, (p', p_r)) \mid (p, \_, a, o, p') \in \mathcal{D} \ \wedge \ o \in \mathcal{O}_n \ \} \ \cup \\
&\{ \ ((p, p_r), a, push_a, \mathtt{tt}) \mid (p, \_, a, pop_{n+1}, p_r) \in \mathcal{D} \ \} \ \cup \\
&\{ \ ((p, p_r), a, push_a, \mathtt{ff}) \mid (p, \_, a, pop_{n+1}, p') \in \mathcal{D} \ \wedge \ p \neq p_r \ \} \ \cup \\
&\{ \ ((p, p_r), a, push_a, (p', p'_r, p_r)) \mid (p, \_, a, push_{n+1}, p') \in \mathcal{D} \ \wedge \ p'_r \in (\mathcal{P} \cup \{\otimes\}) \ \} \ \cup \\
&\{ \ ((p', p'_r, p_r), a, push_a, (p', p'_r)) \mid p' \in \mathcal{P} \ \wedge \ p'_r, p_r \in (\mathcal{P} \cup \{\otimes\}) \ \wedge \ a \in \Sigma \ \} \ \cup \\
&\{ \ ((p', p'_r, p_r), a, push_a, (p'_r, p_r)) \mid p', p'_r \in \mathcal{P} \ \wedge \ p_r \in (\mathcal{P} \cup \{\otimes\}) \ \wedge \ a \in \Sigma \ \}
\end{aligned}
$$

Property 5.1 $\mathcal{L}(P) \neq \emptyset$ iff $\langle (p_0, \otimes), \perp_n \rangle \in Attr_E(\mathcal{R})$.

Proof. From Property C.1 and Property C.2 in Appendix C.4. $\quad\square$

Intuitively, P_G is defined to directly simulate any order-n moves of the pushdown automaton. When a $push_{n+1}$ move is to be played, Eloise is required to give a control state p_r which she claims play will be returned to when the top store added by the $push_{n+1}$ command is removed. If the top store will never be removed, she is able to play $p_r = \otimes$. Abelard then has a choice: either he can accept this assertion and let play continue from p_r with the current store contents, or he can challenge it. If he challenges Eloise's claim, play moves to the control state specified by the $push_{n+1}$ command. From this configuration, Eloise is required to move play to the state p_r as the current top store is pop_{n+1}-ed. If she succeeds, she wins the game, otherwise the play is a victory for Abelard.

Corollary 5.2 *The backwards reachability problem for order-n APDSs is n-EXPTIME-hard.*

Proof. The non-emptiness problem for an order-$(n + 1)$ pushdown automata is n-EXPTIME-hard [10]. By Definition 5.5 and Property 5.1 there is a polynomial reduction of this problem to computing the Eloise's winning region in an order-n pushdown game. By Theorem 5.2 there is a further polynomial reduction of this problem to backwards reachability over an order-n alternating pushdown system. Hence, the backwards reachability problem for order-n APDSs is n-EXPTIME-hard. $\quad\square$

6 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 PhD 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

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] M. Hague and A. Widjaja To. The complexity of model checking (collapsible) higher-order pushdown systems. In *FSTTCS*, pages 228–239, 2010.

[13] 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.

[14] J. A. Brzozowski and E. L. Leiss. On equations for regular languages, finite automata, and sequential networks. *Theor. Comput. Sci.*, 10:19–35, 1980.

[15] J. Bradfield and C. Stirling. Modal logics and mu-calculi: an introduction, 2001.

[16] 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.

[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.

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 $\triangledown$, 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 $n = 1$,

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

and if $n > 1$,

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

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

Property A.1 For any w, the run $\{q_1, \ldots, q_m\} \xrightarrow{w} Q'$ exists in the n-store automaton A iff the run $\{q_1, \ldots, q_m\} \xrightarrow{w} Q'$ 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,

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

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 $i \in \{1, \ldots, m\}$ with $Q' = Q_1^1 \cup \ldots \cup Q_1^m$. Hence, by definition of $\hat{A}$ we have the transition $\{q_1, \ldots, q_m\} \xrightarrow{a} Q_1^1 \cup \ldots \cup Q_1^m = Q_1$. Hence we have the run $\{q_1, \ldots, q_m\} \xrightarrow{w} Q_1$ in $\hat{A}$ as required.

In the other direction we have a run of the form

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

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 $i \in \{1, \ldots, m\}$ with $Q' = Q_1^1 \cup \ldots \cup Q_1^m$. Hence, we have the transition $\{q_1, \ldots, q_m\} \xrightarrow{a} Q_1^1 \cup \ldots \cup Q_1^m = Q_1$ in A. Thus, we have the run $\{q_1, \ldots, q_m\} \xrightarrow{w} Q_1$ in A as required.

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

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

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 $i \in \{1, \ldots, m\}$ with $Q' = Q_1^1 \cup \ldots \cup Q_1^m$. Consequently, we have $\gamma \in \mathcal{L}(B)$ where $B = B_1 \cap \ldots \cap B_m$. By induction over n we have $\gamma \in \mathcal{L}(\hat{B})$. Hence, by

definition of $\hat{A}$ we have the transition $\{q_1, \ldots, q_m\} \xrightarrow{\gamma} Q_1^1 \cup \ldots \cup Q_1^m = Q_1$. Hence we have the run $\{q_1, \ldots, q_m\} \xrightarrow{w} Q_1$ in $\hat{A}$ as required.

In the other direction we have a run of the form

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

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

Corollary A.1 *A set of n-stores is definable by an n-store automaton iff it is definable by a level n nested store automaton.*

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(Q_1)
>
>> **let** $Q_{next} = \emptyset$
>> **for each** $\{q_1, \ldots, q_m\} \in Q_1$
>>> **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 $\{q_1, \ldots, q_m\} \in Q_1$ and some $(q_i, a, Q_{next}^i) \in \Delta$ for each $i \in \{1, \ldots, m\}$ with $Q_{next} = Q_{next}^1 \cup \ldots \cup Q_{next}^m$.

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

> EXPANDWORD($a_1 \ldots a_m, Q$)
>
>> **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 $w = a_1 \ldots a_m$. We have all runs of the form $Q \xrightarrow{a_1} Q_1$ as before, and all runs over $a_2 \ldots a_m$ from all Q_1 by induction. We have all runs of the form $Q \xrightarrow{w} Q'$ therefrom. □

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 $Q \xrightarrow{\widetilde{B}_1} Q' \xrightarrow{\widetilde{B}_2} Q''$ 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 $\widetilde{B}$ with $Q \in Q_1$ and $Q \xrightarrow{\widetilde{B}} Q'$.

$\text{EXPAND}(Q_1)$

> **let** $Q_{next} = \emptyset$
> **for each** $\{q_1, \ldots, q_m\} \in Q_1$
>> **for each** set $\{(q_1, B_1, Q_{next}^1), \ldots, (q_m, B_m, Q_{next}^m)\} \subseteq \Delta$
>> $\quad Q_{next} = Q_{next} \cup \{(\{B_1, \ldots, B_m\}, Q_{next}^1 \cup \ldots \cup Q_{next}^m)\}$
> **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,

$\text{EXPANDETIMES}(e, Q)$

> **let** $Q_1 = \text{EXPAND}(\{Q\})$
> **for** h = 1 **to** e
>> **for each** $(\widetilde{B}_1, \ldots, \widetilde{B}_h, Q') \in Q_1$
>> $\quad Q_1 = Q_1 \cup (\{(\widetilde{B}_1, \ldots, \widetilde{B}_h)\} \times \text{EXPAND}(\{Q'\}))$
> **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 $(\widetilde{B}, Q') \in Q_1$ iff we have the path $Q \xrightarrow{\widetilde{B}} Q'$ in A. After execution of the loop we have, by correctness of EXPAND, $(\widetilde{B}_1, \ldots, \widetilde{B}_e, Q') \in Q_1$ iff we have the following path in A: $Q \xrightarrow{\widetilde{B}_1} \ldots \xrightarrow{\widetilde{B}_e} Q'$. □

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

> **let** $Q = \mathcal{Q}_f$
> **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 $Q' \subseteq \mathcal{Q}_f$. When $w = a_1 a_2 \ldots a_m$ 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 $a_2 \ldots a_m$ 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 $w = \gamma_1 \ldots \gamma_m$,

> **let** $Q = \mathcal{Q}_f$
> **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 $|w| = |\gamma_1| + \cdots + |\gamma_m|$, hence the algorithm requires $\mathcal{O}(|\gamma_1||\Delta_1||\mathcal{Q}| + \cdots + |\gamma_m||\Delta_1||\mathcal{Q}|) = \mathcal{O}(|w||\Delta_1||\mathcal{Q}|)$ 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 $Q' \subseteq \mathcal{Q}_f$. When $w = \gamma_1 \gamma_2 \ldots \gamma_m$ 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 $a_2 \ldots a_m$ is accepted from q'. Hence, we have $q \in Q$ iff there is an accepting run over w from q. $\qquad \square$

B.3   Boolean Operations

We define the boolean operations on n-store multi-automata.

Definition B.1 Given two n-store multi-automata $A^1 = (\mathcal{Q}_1, \Sigma, \Delta_1, \{q_1^1, \ldots, q_1^z\}, \mathcal{Q}_f^1)$ and $A^2 = (\mathcal{Q}_2, \Sigma, \Delta_2, \{q_2^1, \ldots, q_2^z\}, \mathcal{Q}_f^2)$, we define,

$$A^1 \cup A^2 = (\mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \{q^1, \ldots, q^z\}, \Sigma, \Delta_\cup, \{q^1, \ldots, q^z\}, \mathcal{Q}_f^\cup)$$

where $q^i \notin \mathcal{Q}_1 \cup \mathcal{Q}_2$ for $i \in \{1, \ldots, z\}$ and

$$
\begin{aligned}
\Delta_\cup \ =\ & \Delta_1 \cup \Delta_2 \\
& \cup\ \{\ (q^i, B, Q) \mid (q_1^i, B, Q) \in \Delta_1\ \wedge\ 1 \le i \le z\ \} \\
& \cup\ \{\ (q^i, B, Q) \mid (q_2^i, B, Q) \in \Delta_2\ \wedge\ 1 \le i \le z\ \}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{Q}_f^\cup \ =\ & \mathcal{Q}_f^1 \cup \mathcal{Q}_f^2 \\
& \cup\ \{\ q^i \mid q_j^i \in \mathcal{Q}_f^j\ \wedge\ 1 \le j \le 2\ \wedge 1 \le i \le z\ \}
\end{aligned}
$$

We define,

$$A^1 \cap A^2 = (\mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \{q^1, \ldots, q^z\}, \Sigma, \Delta_\cap, \{q^1, \ldots, q^z\}, \mathcal{Q}_f^\cap)$$

where $q^i \notin \mathcal{Q}_1 \cup \mathcal{Q}_2$ for $i \in \{1, \ldots, z\}$ and at order-1 we have,

$$\Delta_\cap \;=\; \Delta_1 \cup \Delta_2 \cup \{ \, (q^i, a, Q_1 \cup Q_2) \mid (q_1^i, a, Q_1) \in \Delta_1 \wedge (q_2^i, a, Q_2) \in \Delta_2 \ \wedge \ 1 \le i \le z \, \}$$

$$\begin{aligned}
\mathcal{Q}_f^\cap \;=\; & \mathcal{Q}_f^1 \cup \mathcal{Q}_f^2 \\
& \cup \{ \, q^i \mid q_1^i \in \mathcal{Q}_f^1 \ \wedge \ q_2^i \in \mathcal{Q}_f^2 \ \wedge \ 1 \le i \le z \, \}
\end{aligned}$$

Otherwise we have,

$$\begin{aligned}
\Delta_\cap \;=\; & \Delta_1 \cup \Delta_2 \\
& \cup \{ \, (q^i, B_1 \cap B_2, Q_1 \cup Q_2) \mid (q_1^i, B_1, Q_1) \in \Delta_1 \wedge (q_2^i, B_2, Q_2) \in \Delta_2 \wedge 1 \le i \le z \, \} \\
& \cup \{ \, (q^i, \triangledown, \{q_f^\varepsilon\}) \mid (q_1^i, \triangledown, \{q_f^\varepsilon\}) \in \Delta_1 \wedge (q_2^i, \triangledown, \{q_f^\varepsilon\}) \in \Delta_2 \wedge 1 \le i \le z \, \}
\end{aligned}$$

$$\mathcal{Q}_f^\cap \;=\; \mathcal{Q}_f^1 \cup \mathcal{Q}_f^2$$

Where $B_1 \cap B_2$ is defined recursively.

Property B.1 *Given two n-store multi-automata A_1 and A_2, we have $\mathcal{L}(A_1) \cap \mathcal{L}(A_2) = \mathcal{L}(A_1 \cap A_2)$ and $\mathcal{L}(A_1) \cup \mathcal{L}(A_2) = \mathcal{L}(A_1 \cup A_2)$.*

Proof. To prove the property for $A_1 \cup A_2$ in the case of an empty stack ε we have for all $i \in \{1, \ldots, z\}$ that $q^i \in \mathcal{Q}_f^\cup$ iff we have $q_j^i \in \mathcal{Q}_f^j$ for $j = 1$ or $j = 2$. Otherwise, we observe that we have for any $i \in \{1, \ldots, z\}$ an accepting run in $A_1 \cup A_2$ of a store γw,

$$q^i \xrightarrow{\ \gamma\ } Q \xrightarrow{\ w\ } Q_f$$

with $Q_f \subseteq \mathcal{Q}_f$ iff we have a run,

$$q_j^i \xrightarrow{\ \gamma\ } Q \xrightarrow{\ w\ } Q_f$$

in A_i for $j = 1$ or $j = 2$.

For $A_1 \cap A_2$ we proceed by induction over n. When $n = 1$, in the case of the empty stack ε we have for any $i \in \{1, \ldots, z\}$ that $q^i \in \mathcal{Q}_f^\cap$ iff $q_j^i \in \mathcal{Q}_f^j$ for $j = 1$ and $j = 2$. Otherwise, we have an accepting run of $A_1 \cap A_2$ over the word aw,

$$q^i \xrightarrow{\ a\ } Q_1 \cup Q_2 \xrightarrow{\ w\ } Q_f^1 \cup Q_f^2$$

with $Q_f^1 \cup Q_f^2 \subseteq \mathcal{Q}_f$ iff we have the accepting runs,

$$q_j^i \xrightarrow{\ a\ } Q_j \xrightarrow{\ w\ } Q_f^j$$

in A_j for $j = 1$ and $j = 2$.

When $n > 1$, we have an accepting run of $A_1 \cap A_2$ over the word γw,

$$q_0 \xrightarrow{\ \gamma\ } Q_1 \cup Q_2 \xrightarrow{\ w\ } Q_f^1 \cup Q_f^2$$

with $Q_f^1 \cup Q_f^2 \subseteq \mathcal{Q}_f$ iff (via the induction hypothesis) we have the accepting runs,

$$q^i \xrightarrow{\ \gamma\ } Q_i \xrightarrow{\ w\ } Q_f^i$$

in A_i for $i = 1$ and $i = 2$. $\qquad\square$

We now show how to complement n-store multi-automata. We begin by defining an operation on sets of sets.

Definition B.2 Given a set of sets $\{Q_1, \ldots, Q_m\}$ we define,

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

Definition B.3 Given an n-store multi-automaton $A = (\mathcal{Q}, \Sigma, \Delta, \{q^1, \ldots, q^z\}, \mathcal{Q}_f)$, we define $\bar{A}$ as follows.

- When $n = 1$ we assume A is *total* (this is a standard assumption that can easily be satisfied through the addition of a sink state). We define $\bar{A} = (\mathcal{Q}, \Sigma, \Delta', \{q^1, \ldots, q^z\}, \mathcal{Q} \setminus \mathcal{Q}_f)$ 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 $(q, a, Q_1), \ldots, (q, a, Q_m)$, and

  2. $Q_a = invert \left(\bigcup_{1 \in \{1, \ldots, m\}} \{Q_i\} \right)$, and

  3. $\Delta'(q, a) = Q_a$.

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

- When $n > 1$ we define $\bar{A} = (\mathcal{Q} \cup \{q_f^*, q_f^\varepsilon\}, \Sigma, \Delta', \{q^1, \ldots, q^z\}, (\mathcal{Q} \cup \{q_f^*, q_f^\varepsilon\}) \setminus \mathcal{Q}_f)$ 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 $(q, B_1, Q_1), \ldots, (q, B_m, Q_m)$ (we assume $m \geq 1$), and

  2. For all $\widetilde{B} \in 2^{B_1, \ldots, B_m}$ we have,

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

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

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

  3. $\Delta'(q, B_{\widetilde{B}}) = Q_{\widetilde{B}}$, and

  4. For all $j \in \{1, \ldots, z\}$ 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 $B_{\widetilde{B}}$ may take exponential time. The construction is therefore exponential.

We now show that the above definition is correct.

Property B.2 *Given an n-store multi-automaton A, we have $\mathcal{L}(\bar{A}^{q^j}) = \overline{\mathcal{L}(A^{q^j})}$ for all $q^j \in \{q^1, \ldots, q^z\}$.*

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 $Q' \subset \mathcal{Q}_f$. 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 $\{(q, a, Q_1^\top), \ldots, (q, a, Q_e^\top)\}$ be the set of all transitions in Δ from q over a. For each $i \in \{1, \ldots, e\}$, 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 $q \xrightarrow{a} \{q_1^\top, \ldots, q_e^\top\}$ exists in $\bar{A}$. Hence we have the accepting run,

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

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 $\mathcal{Q}_f \cap (\mathcal{Q} \cup \{q_0\}) \setminus \mathcal{Q}_f = \emptyset$. Furthermore, since we have an (accepting) $\triangledown$-transition from q^j for all $j \in \{1, \ldots, z\}$ in A iff we have an (accepting) $\triangledown$-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 $Q' \subset \mathcal{Q}_f$. 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, 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 $Q' \subseteq \mathcal{Q}_f$ 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 $\{(q, B_1^t, Q_1^t), \ldots, (q, B_e^t, Q_e^t), (q, B_1^f, Q^f), \ldots, (q, B_h^f, Q_h^f)\}$ be the set of all transitions in Δ from q such that $\gamma \in B_i^t$ for all $i \in \{1, \ldots, e\}$ and $\gamma \notin B_i^f$ for all $i \in \{1, \ldots, h\}$ (and consequently $\gamma \in \bar{B}_i^f$). For each $i \in \{1, \ldots, e\}$ 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 $q \xrightarrow{B} \{q_1^t, \ldots, q_e^t\}$ with $B = B_1^t \cap \ldots \cap B_e^t \cap \bar{B}_1^f \cap \ldots \cap \bar{B}_h^f$ exists in $\bar{A}$. Hence we have the accepting run,

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

  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. $\square$

B.4 Soundness and Completeness for $A_0, A_1, \ldots$

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

B.4.1 Preliminaries

We begin by proving some useful properties of the automaton construction. These properties assert that the automata constructed from the sets in $\widetilde{\mathcal{G}}_l^i$ are well-behaved. Once this has been established, we need only consider order-n of the automata $A_0, A_1, \ldots$ 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 B.3 shows that inherited runs are sound.

2. Lemma B.2 shows the completeness of inherited runs.

3. Lemma B.1, Lemma B.4 and Lemma B.5 show that derived runs are sound.

4. Lemma B.6 shows the completeness of derived runs.

Definition B.4 A non-empty run $g_{(q_1,Q_2)}^i \xrightarrow{w}_i Q$ of $\mathcal{G}_l^i$ or $q^j \xrightarrow{w}_i Q$ of A_i can be characterised by its initial transition $g_{(q_1,Q_2)}^i \xrightarrow{\gamma}_i Q'$ where $w = \gamma w'$. There are several cases,

- $l = 1$.

  Then 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 $S \in \widetilde{G}_{(q_1,Q_2)}^i$. We say that the run was **derived** from S.

- $l > 1$.

  Then we have $g_{(q_1,Q_2)}^i \xrightarrow{\widetilde{G}} Q'$ with $\gamma \in \mathcal{L}(G)$. There are three cases depending on the accepting run of G and the source of the transition from $g_{(q_1,Q_2)}^i$,

  - If the accepting run of G is an inherited run, then the run $g_{(q_1,Q_2)}^i \xrightarrow{w}_i Q$ is also **inherited**.

  - If the accepting run of G is derived from some $S' \in \widetilde{G}$ and S' was added to $\widetilde{G}$ when inheriting transitions from $g_{(q_1,Q_2)}^{i-1}$, then the run $g_{(q_1,Q_2)}^i \xrightarrow{w}_i Q$ is **inherited**.

  - If the accepting run of G is derived from some $S' \in \widetilde{G}$ and S' was added to $\widetilde{G}$ by some $S \in \widetilde{G}_{(q_1,Q_2)}^i$ or — if $l = n - 1$ — by $T_{\mathcal{D}}$ and the command d, then the run $g_{(q_1,Q_2)}^i \xrightarrow{w}_i Q$ is **derived** from S or d respectively.

The first lemma states that runs derived from a set S which is the conjunction of several automata behaves correctly:

Lemma B.1 The run $g_{(q_1,Q_2)}^{i+1} \xrightarrow{w}_{i+1} Q$ derived from $S = \{\theta_1, \ldots, \theta_m\} \subseteq \mathcal{B}_l \cup \widetilde{\mathcal{G}}_l^i$ exists in $\mathcal{G}_l^{i+1}$ iff the run,

$$\{q^{\theta_1}, \ldots, q^{\theta_m}\} \xrightarrow{w}_i Q$$

exists in $\mathcal{G}_l^i$.

Proof. We prove the lemma by induction over l. Observe that a derived run cannot be empty. When $l = 1$, we have $w = aw'$ for some character a and word w' and,

$$g_{(q_1,Q_2)}^{i+1} \xrightarrow{a}_{i+1} Q'_1 \cup \cdots \cup Q'_m \xrightarrow{w'}_i Q$$

derived from S exists iff the run $q^{\theta_t} \xrightarrow{a}_i Q'_t$ exists in $\mathcal{G}_1^i$ for all $t \in \{1, \ldots, m\}$ and the run,

$$\{q^{\theta_1}, \ldots, q^{\theta_m}\} \xrightarrow{a}_i Q'_1 \cup \cdots \cup Q'_m \xrightarrow{w'}_i Q$$

exists in $\mathcal{G}_1^i$ as required.

When $l > 1$, we have $w = \gamma w'$. The transition $g_{(q_1,Q_2)}^{i+1} \xrightarrow{\widetilde{G}}_{i+1} Q'$ derived from S and the run $Q' \xrightarrow{w'}_i Q$ exist in $\mathcal{G}_l^{i+1}$ where $\gamma \in \mathcal{L}(G)$ iff $Q' = Q'_1 \cup \cdots \cup Q'_m$ and the transition $q^{\theta_t} \xrightarrow{B_t}_i Q_t$ exists in $\mathcal{G}_l^i$ for all $t \in \{1, \ldots, m\}$. We have $\{B_1, \ldots, B_m\} = S' \in \widetilde{G}_{(g_{(q_1,Q_2)}^{i+1}, Q')}^{i+1} \in \widetilde{\mathcal{G}}_{l-1}^i$ and the accepting run of G is derived from S' iff we have $\gamma \in \mathcal{L}(B_t)$ for all $t \in \{1, \ldots, m\}$ (by induction) and the run,

$$\{q^{\theta_1}, \ldots, q^{\theta_m}\} \xrightarrow{\gamma}_{i+1} Q' \xrightarrow{w'}_i Q$$

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

The language accepted by the sequence $A_0, A_1, \ldots$ or any sequence $\mathcal{G}_l^0, \mathcal{G}_l^1, \ldots$ 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 B.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, 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 $B \in \mathcal{B}_l$ or $\triangledown$. 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,

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

where $w = \gamma w'$.

In case $\widetilde{G}_{(q_1,Q_2)}^i \in \widetilde{\mathcal{G}}_l^i$ and $l = 1$, 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, as required, we have the run,

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

When $\widetilde{G}_{(q_1,Q_2)}^i \in \widetilde{\mathcal{G}}_l^i$ and $l > 1$, by definition the automaton $G_{(q_1,Q_2)}^{i+1}$ has the transition $g_{(q_1,Q_2)}^{i+1} \xrightarrow{\widetilde{G}}_i Q^2$ with $\{B\} \in \widetilde{G}$ for every transition $g_{(q_1,Q_2)}^i \xrightarrow{B}_i Q^2$. Since we have $\gamma \in \mathcal{L}(B)$ (for

some B) and q^B is of the form $g^i_{(q,Q')}$ (and hence not an accepting state) we have $\gamma \neq [\varepsilon]$ and $\gamma \in \mathcal{L}(G)$ by Lemma B.1 and $\{B\} \in \widetilde{G}$. Hence we have the run,

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

as required. □

We now show that inheritance does not introduce spurious runs:

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

Proof. We proceed by induction over l. Observe that an inherited run cannot be empty.
   When $l = 1$ we have $w = aw'$ and,

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

Since the run is an inherited run, we have $g^i_{(q_1,Q_2)} \xrightarrow{a}_i Q'$ in $\mathcal{G}^i_1$ and hence,

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

in $\mathcal{G}^i_1$ as required.
   When $l > 1$ we have $w = \gamma w'$ and $g^{i+1}_{(q_1,Q_2)} \xrightarrow{\widetilde{G}^{i+1}_{(\dots)}}_{i+1} Q'$ with $Q' \xrightarrow{w'}_i Q$ in $\mathcal{G}^{i+1}_l$ and $\gamma \in \mathcal{L}(G^{i+1}_{(\dots)})$. There are two cases depending on the accepting run of $G^{i+1}_{(\dots)}$.
   If the accepting run of $G^{i+1}_{(\dots)}$ is inherited, then we have $\gamma \in \mathcal{L}(G^i_{(\dots)})$ by induction on l and hence the transition $g^i_{(q_1,Q_2)} \xrightarrow{\gamma}_i Q'$ in $\mathcal{G}^i_l$. Therefore, as required, we have the run,

$$g^i_{(q_1,Q_2)} \xrightarrow{\gamma}_i Q' \xrightarrow{w'}_i Q$$

If the accepting run of $G^{i+1}_{(\dots)}$ is derived from some S introduced to $\widetilde{G}^{i+1}_{(\dots)}$ when inheriting transitions from $g^i_{(q_1,Q_2)}$, then $S = \{B\}$ for some B and we have $g^i_{(q_1,Q_2)} \xrightarrow{B}_i Q'$ in $\mathcal{G}^i_l$. Furthermore, by Lemma B.1 we have $\gamma \in \mathcal{L}(B)$ and the run,

$$g^i_{(q_1,Q_2)} \xrightarrow{\gamma}_i Q' \xrightarrow{w'}_i Q$$

as required. □

The next two lemmata assert that derived runs are justified.

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

Proof. The proof is by induction over l. Observe that, since the run is derived, we have $w \neq \varepsilon$.
   In the base case $l = 1$. Let $w = aw'$. We have the following run in $\mathcal{G}^{i+1}_l$,

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

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 $\mathcal{G}^i_l$ 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.

When $l > 1$, let $w = \gamma w'$. We have the run,

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

in $\mathcal{G}^{i+1}_l$. In particular, we have $g^{i+1}_{(q_1,Q_2)} \xrightarrow{\widetilde{G}}_{i+1} Q^1$ with an accepting run of G over γ that is derived from $S' \in \widetilde{G}$ which was introduced by S. By definition, since $\theta_1 \in S$, we have $q^{\theta_1} \xrightarrow{\theta}_i Q^2$ with $Q^2 \subseteq Q^1$ and $\theta \in S'$. By induction, we have $\gamma \in \mathcal{L}(\theta)$ and hence the run,

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

with $Q' \subseteq Q$ as required. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ □

Lemma B.5 *Suppose the run $g^{i+1}_{(q_1,Q_2)} \xrightarrow{w}_{i+1} Q$ derived from S exists in $\mathcal{G}^{i+1}_l$ and $(a, o, \theta_1) \in S$. Let $[w'] = o([w])$, we have $q^{\theta_1} \xrightarrow{w'}_i Q'$ in $\mathcal{G}^i_l$, where $Q' \subseteq Q$.*

Proof. The proof is by induction over l. Since the run is derived, we have $w \neq \varepsilon$.

In the base case $l = 1$. We have $w = aw''$. There is only one value of o,

- $o = push_{w_p}$. Then $[w'] = o([w]) = [w_p w'']$. We have the following run in $\mathcal{G}^{i+1}_l$,

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

  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 $\mathcal{G}^i_l$ 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.

When $l > 1$, let $w = \gamma w''$ and we have the run,

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

in $\mathcal{G}^{i+1}_l$. In particular $g^{i+1}_{(q_1,Q_2)} \xrightarrow{\widetilde{G}}_{i+1} Q^1$ and there is an accepting run of G over γ derived from some $S' \in \widetilde{G}$. There are now three cases depending on o,

- $o = push_l$. By definition, since $(a, push_l, \theta_1) \in S$ we have in $\mathcal{G}^i_l$ the run

$$q^{\theta_1} \xrightarrow{\widetilde{\theta}_1}_i Q^2 \xrightarrow{\widetilde{\theta}_2}_i Q^3$$

  with $Q^3 \subseteq Q^1$ and $\{B^a_l\} \cup \widetilde{\theta}_1 \cup \widetilde{\theta}_2 \subseteq S'$. Hence, by Lemma B.4, we have $\gamma \in \mathcal{L}(\{B^a_l\} \cup \widetilde{\theta}_1 \cup \widetilde{\theta}_2)$, and hence $[w'] = o([w]) = [\gamma\gamma w'']$ and we have the following run of $\mathcal{G}^i_l$,

$$q^{\theta_1} \xrightarrow{\gamma}_i Q^2 \xrightarrow{\gamma}_i Q^3 \xrightarrow{w''}_i Q'$$

  with $Q' \subseteq Q$ as required.

- $o = pop_l$. Since $(a, pop_l, \theta_1) \in S$, we have $q^{\theta_1} \in Q^1$ and $B^a_l \in S'$. Furthermore w'' is non-empty since $q^{\theta_1} \notin \mathcal{Q}_f$ by definition of n-store automata. By Lemma B.4 $\gamma \in \mathcal{L}(B^a_l)$. Hence $[w'] = o([w]) = [w'']$ and we have $q^{\theta_1} \xrightarrow{w''}_i Q'$ with $Q' \subseteq Q$ as required.

- $\ell(o) < l$. By definition, $[w'] = o([w]) = [o(\gamma)w'']$. Since $(a, o, \theta_1) \in S$ we have $q^{\theta_1} \xrightarrow{\theta}_i Q^2$ with $Q^2 \subseteq Q^1$ and $(a, o, \theta) \in S'$. By induction over l we have $o(\gamma) \in \mathcal{L}(\theta)$ and hence,

$$q^{\theta_1} \xrightarrow{o(\gamma)}_i Q^2 \xrightarrow{w''}_i Q'$$

with $Q' \subseteq Q$ as required.

This completes the proof of the lemma. □

Finally, we show that the derived runs completely represent their source.

Lemma B.6 *Let* $S = \{\alpha_1, \ldots, \alpha_m\} \in \widetilde{G}^{i+1}_{(q,Q)}$. *Given some* γ *with* $top_1(\gamma) = a$ *such that for each* $e \in \{1, \ldots, m\}$ *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* $a = b$, $o_e(\gamma) = \gamma_e$ *and* $\gamma_e \in \mathcal{L}(\theta_e)$

we have $\gamma \in \mathcal{L}(G^{i+1}_{(q,Q)})$.

Proof. We have $\widetilde{G}^{i+1}_{(q,Q)} \in \widetilde{\mathcal{G}}^{i+1}_l$ for some l. The proof is by induction over l.

When $l = 1$ let $\gamma = [aw]$. We have $\alpha_e = \theta_e$ or $\alpha_e = (a, push_{w_e}, \theta_e)$. We have,

- When $\alpha_e = \theta_e$, the run,
$$q^{\theta_e} \xrightarrow{a}_i Q_e \xrightarrow{w}_i Q^e_f$$

with $Q^e_f \subseteq \mathcal{Q}_f$ in $\mathcal{G}^i_l$. Furthermore, $\gamma_e = \gamma$.

- When $\alpha_e = (a, push_{w_e}, \theta_e)$, the run,
$$q^{\theta_e} \xrightarrow{w_e}_i Q_e \xrightarrow{w}_i Q^e_f$$

with $Q^e_f \subseteq \mathcal{Q}_f$ in $\mathcal{G}^i_l$. Furthermore, we have $\gamma_e = [w_e w]$.

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

$$g^{i+1}_{(q,Q)} \xrightarrow{a}_{i+1} Q_1 \cup \cdots \cup Q_m \xrightarrow{w}_i Q^1_f \cup \cdots \cup Q^m_f$$

with $Q^1_f \cup \cdots \cup Q^m_f \subseteq \mathcal{Q}_f$. Hence $\gamma \in \mathcal{L}(G^{i+1}_{(q,Q)})$ as required.

When $l > 1$ let $\gamma = [\gamma' w]$. We have $S' = S'_1 \cup \cdots \cup S'_m$ and $Q' = Q_1 \cup \cdots \cup Q_m$ where,

- When $\alpha_e = \theta_e$, $\gamma = \gamma_e$ and we have the transition $q^{\theta_e} \xrightarrow{\theta'_e}_i Q_e$ in $\mathcal{G}^i_l$ with $\gamma' \in \mathcal{L}(\theta'_e)$ and the run $Q_e \xrightarrow{w}_i Q^e_f$ with $Q_f \subseteq \mathcal{Q}_f$. Furthermore $S'_e = \{\theta'_e\}$.

- When $\alpha_e = (a, push_l, \theta_e)$, $\gamma_e = [\gamma' \gamma' w]$. Additionally, we have the transitions,

$$q^{\theta_e} \xrightarrow{\theta^1_e}_i Q' \xrightarrow{\widetilde{\theta^2_e}}_i Q_e$$

in $\mathcal{G}^i_l$ where $\gamma' \in \mathcal{L}(\{B^a_{l-1}, \theta^1_e\} \cup \widetilde{\theta^2_e})$. Furthermore, we have the run $Q_e \xrightarrow{w}_i Q^e_f$ with $Q^e_f \subseteq \mathcal{Q}_f$ and $S'_e = \{B^a_{l-1}, \theta^1_e\} \cup \widetilde{\theta^2_e}$.

- When $\alpha_e = (a, pop_l, \theta_e)$, $\gamma_e = [w]$ and we have the run,

$$q^{\theta_e} \xrightarrow{w}_i Q^e_f$$

with $Q^e_f \subseteq \mathcal{Q}_f$, $S'_e = \{B^a_{l-1}\}$, $\gamma' \in \mathcal{L}(B^a_{l-1})$ and $Q_e = \{q^{\theta_e}\}$.

48

- $\alpha_e = (a, o_e, \theta_e)$ with $\ell(o_e) < l$, we have $\gamma_e = [o_e(\gamma')w]$, and the transition $q^{\theta_e} \xrightarrow{\theta'_e}_i Q_e$ and run $Q_e \xrightarrow{w}_i Q_f^e$ with $Q_f^e \subseteq \mathcal{Q}_f$ in $\mathcal{G}_l^i$. Additionally, $o_e(\gamma') \in \mathcal{L}(\theta'_e)$ and $S'_e = \{(a, o_e, \theta'_e)\}$.

Hence, by definition of $G_{(q,Q)}^{i+1}$, we have the transition,

$$g_{(q,Q)}^{i+1} \xrightarrow{\widetilde{G}}_{i+1} Q_1 \cup \cdots \cup Q_m$$

with $S' \in \widetilde{G}$ and by induction over l, $\gamma' \in \mathcal{L}(G)$. Hence we have the run,

$$g_{(q,Q)}^{i+1} \xrightarrow{\gamma'}_{i+1} Q_1 \cup \cdots \cup Q_m \xrightarrow{w}_i Q_f^1 \cup \cdots \cup Q_f^m$$

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

B.4.2 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 $\langle p^j, \gamma \rangle \xhookrightarrow{*} C$ with $C \subseteq C_{Init}$. Let $I = \{q^1, \ldots, q^z\}$. The following lemma describes the relationship between added transitions and the evolution of the order-n PDS. The restrictions on w' are technical requirements in the case of pop_n 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 B.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 $\langle p^j, [ww'] \rangle \xhookrightarrow{*} C$, 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 B.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 $S \in \widetilde{G}_{(q^j,Q)}^{i+1}$ introduced by during the processing of d.

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

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

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

- $o_t = push_n$.

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

$$q^{k_t} \xrightarrow{\widetilde{\theta_1}}_i Q' \xrightarrow{\widetilde{\theta_2}}_i Q'_t$$

49

with $\{B_{n-1}^a\} \cup \widetilde{\theta}_1 \cup \widetilde{\theta}_2 \subseteq S$. By Lemma B.4 we have $\gamma \in \mathcal{L}(\{B_{n-1}^a\} \cup \widetilde{\theta}_1 \cup \widetilde{\theta}_2)$. Hence we have,

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

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

- $o_t = pop_n$.

  We have $B_{n-1}^a \in S$. We have, by Lemma B.4, $\gamma \in \mathcal{L}(B_{n-1}^a)$.

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

- $\ell(o_t) < n$ and if $o_t = pop_l$ then $X_{\ell(o_t)}^a \notin S$.

  By definition, we have $q^{k_t} \xrightarrow{\theta}_i Q_t$ in A_i with $(a, o_t, \theta) \in S$. Therefore, by Lemma B.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, we have $\langle p^{k_t}, o_t[\gamma w']\rangle \in C'$ and via induction over i we have a set C_t with $\langle p^{k_t}, o_t[\gamma w']\rangle \overset{*}{\hookrightarrow} C_t$ which satisfies the lemma.

- $\ell(o_t) < n$, $o_t = pop_l$ and $X_{\ell(o_t)}^a \in S$.

  Since $X_l^a \in S$ by Lemma B.4 we have $\gamma \in \mathcal{L}(X_l^a)$. Hence $o_t[\gamma w']$ is undefined and we have $\langle p^j, \triangledown\rangle \in C'$. Because $X_l^a \in S$, by definition we have $q^j \xrightarrow{\triangledown}_i \{q_f^\varepsilon\}$. Since $\triangledown$ transitions are never added, it must be the case that $q^j \xrightarrow{\triangledown}_0 \{q_f^\varepsilon\}$. Let $C_t = \{\langle p^j, \triangledown\rangle\}$.

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

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

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

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

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

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

$$Q_1^{\backslash I} \xrightarrow{\gamma_2}_0 \ldots \xrightarrow{\gamma_m}_0 Q_m'$$

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 $\langle p^j, [\gamma_1 \ldots \gamma_m w']\rangle \overset{*}{\hookrightarrow} C_1$ satisfying the lemma for a single transition. Let C_1^I be the set of all $\langle p^k, \gamma_2 \ldots \gamma_m w'\rangle \in C_1$ and $C_1' = C_1 \setminus C_1^I$. For each $q^k \in Q_1^I$ we have $\langle p^k, [\gamma_2 \ldots \gamma_m w']\rangle \in C_1$ since there are no transitions to initial states in A_0 (and hence we must have $q^k \xrightarrow{\varepsilon}_0 \{q^k\}$ to satisfy the conditions of the lemma for C_1). From $\langle p^k, [\gamma_2 \ldots \gamma_m w']\rangle \overset{*}{\hookrightarrow} C_k$ and since we have $Q_1^{\backslash I} \xrightarrow{\gamma_2 \ldots \gamma_m}_0 Q_m'$, it is the case that the set $C = C_1' \cup \bigcup_{q^k \in Q_1^I} C_k$ which has $\langle p^j, [\gamma_1 \ldots \gamma_m w']\rangle \overset{*}{\hookrightarrow} C_1 \overset{*}{\hookrightarrow} C$ and satisfies the lemma as required. $\qquad\square$

Property B.3 (Soundness) *For any configuration $\langle p^j, \gamma\rangle$ such that $\gamma \in \mathcal{L}(A_i^{q^j})$ for some i, we have $\langle p^j, \gamma\rangle \overset{*}{\hookrightarrow} C$ 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 $q^j \xrightarrow{w_\gamma}_i Q_f$ with $Q_f \subseteq \mathcal{Q}_f$. Since $\mathcal{Q}_f$ contains no initial states, we apply Lemma B.7 with $w' = \varepsilon$. Therefore, we have $\langle p^j, \gamma \rangle \xhookrightarrow{*} C \subseteq \mathcal{L}(A_0^{q^k})$. Since A_0 is defined to represent C_{Init}, soundness follows. $\qquad\square$

B.4.3 Completeness

Property B.4 (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 $\langle p^j, \gamma \rangle \xhookrightarrow{*} C$ 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 $\langle p^j, \gamma \rangle \hookrightarrow C_1 \xhookrightarrow{*} C_2$ 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 $\langle p^j, \gamma \rangle \hookrightarrow C_1$.

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

$$
\begin{aligned}
C_1 \;=\; & \{ \langle p^{k_t}, \gamma' \rangle \mid t \in \{1, \ldots, m\} \wedge \gamma' = o_t(\gamma) \} \\
& \cup \{ \langle p^j, \triangledown \rangle \mid \text{if } o_t(\gamma) \text{ with } t \in \{1, \ldots, m\} \text{ is not defined} \}
\end{aligned}
$$

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

Let $\gamma = [\gamma' w]$. We have $S' = S_1' \cup \cdots \cup S_m'$ and $Q' = Q_1 \cup \cdots \cup Q_m$ where, for each $e \in \{1, \ldots, m\}$,

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

$$
q^{k_e} \xrightarrow{\theta_e^1}_i Q' \xrightarrow{\widetilde{\theta_e^2}}_i Q_e
$$

  in A_i where $\gamma' \in \mathcal{L}(\{B_{n-1}^a, \theta_e^1\} \cup \widetilde{\theta_e^2})$. Furthermore, we have the run $Q_e \xrightarrow{w}_i Q_f^e$ with $Q_f^e \subseteq \mathcal{Q}_f$ and $S_e' = \{B_{n-1}^a, \theta_e^1\} \cup \widetilde{\theta_e^2}$.

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

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

  in A_i with $Q_f^e \subseteq \mathcal{Q}_f$, $S_e' = \{B_{n-1}^a\}$, $\gamma' \in \mathcal{L}(B_{n-1}^a)$ and $Q_e = \{q^{k_e}\}$.

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

$$
q^j \xrightarrow{\triangledown}_i \{q_f^\varepsilon\}
$$

  if A_i. Hence we have $S_e' = \{B_{n-1}^a\}$, $\gamma' \in \mathcal{L}(B_{n-1}^a)$ and $Q_e = Q_f^e = \{q_f^\varepsilon\}$.

- When $\ell(o_e) < n$, and we have $o_e(\gamma) = [o_e(\gamma') w]$, we have the transition $q^{k_e} \xrightarrow{\theta_e'}_i Q_e$ and run $Q_e \xrightarrow{w}_i Q_f^e$ with $Q_f^e \subseteq \mathcal{Q}_f$ in A_i. Additionally, $o_e(\gamma') \in \mathcal{L}(\theta_e')$ and $S_e' = \{(a, o_e, \theta_e')\}$.

  If $o_e(\gamma)$ is not defined we have $\gamma = [\gamma' w]$, $o_e = pop_l$ for some $l < n$ and the run,

$$
q^j \xrightarrow{\triangledown}_i \{q_f^\varepsilon\}
$$

  in A_i. Hence we have $S_e' = \{X_l^a\}$, $\gamma' \in \mathcal{L}(X_l^a)$ and $Q_e = \{q_f^*\}$ with $q_f^* \xrightarrow{w}_i Q_f^e$ and $Q_f^e \subseteq \mathcal{Q}_f$.

51

Hence, by definition of A_{i+1}, we have the transition,

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

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

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

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

B.5   Proofs for A_*

In this section we provide proofs of Lemma 4.1 and Lemma 4.2. The proof of the first lemma is somewhat involved, hence we deal with the order-1 and the order-l for $l > 1$ cases individually. The main idea in both proofs is that the loops in $\hat{\mathcal{G}}_l^i$ can simulate, correctly, the prefix of any run in $\mathcal{G}_l^{i'}$ and vice-versa. That is, a run in $\hat{\mathcal{G}}_l^i$ begins by traversing it's initial loops before progressing to its accepting states. If we unroll this looping we will construct a run of $\mathcal{G}_l^{i'}$ for a sufficiently large i'. In the other direction, the prefix of a run in $\mathcal{G}_l^{i'}$ can be simulated by the initial looping behaviour of $\hat{\mathcal{G}}_l^i$.

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

Lemma B.8 *Given* $g_{(q_y,Q_y)}^{i_y} \xrightarrow{w}_{i_y} Q_y$ *for all* $y \in \{1,\ldots,h\}$ *for some* h, *let* i_{max} *be the maximum* i_y. *We have* $\{g_{(q_1,Q_1)}^{i_{max}},\ldots,g_{(q_h,Q_h)}^{i_{max}}\} \xrightarrow{w} \bigcup_{y\in\{1,\ldots,h\}} Q_y$.

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

B.5.1   Proof of Lemma 4.1 for $l = 1$

There are three parts to the proof. We first prove that a fixed point i_0 is reached. We then prove Lemma 4.1 in both directions.

Lemma B.9 *There exists some* i_0 *such that* $\hat{\mathcal{G}}_1^i = \hat{\mathcal{G}}_1^{i_0}$ *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* $\hat{\mathcal{G}}_1^{i_0}$.

Proof. This is a simple consequence of the finiteness of Σ and that $T_{\widetilde{\mathcal{G}}_1^{i_1}[i_1/i_1-1]}$ only adds transitions and never states. The automaton will eventually become saturated and no new transitions will be added. $\qquad\square$

Lemma B.10 *For all* w, *if* $g_{(q,Q')}^i \xrightarrow{w}_i Q_1$ *with* $Q_1 \subseteq \mathcal{Q}_f$ *is a run in* $\mathcal{G}_1^i$ *for some* i, *then we have* $g_{(q,Q')}^{i_1} \xrightarrow{w}_{i_0} Q_2$ *with* $Q_2 \subseteq \mathcal{Q}_f$ *in* $\hat{\mathcal{G}}_1^{i_0}$.

Proof. We prove the following property. For any path $g_{(q,Q')}^i \xrightarrow{w}_i \{q_1,\ldots,q_h\}$ in $\mathcal{G}_1^i$, we have a path $g_{(q,Q')}^{i_1} \xrightarrow{w}_{i_0} \{q_1^!,\ldots,q_h^!\}$ in $\hat{\mathcal{G}}_1^{i_0}$ with,

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

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

There are two cases. When $i \leq i_1$, then using that we have only added transitions to $\mathcal{G}_1^{i_1}$ to define $\hat{\mathcal{G}}_1^{i_0}$ and that $q_y^! = q_y$ for all y, we have $g_{(q,Q')}^{i_1} \xrightarrow{w'}_{i_0} \{q_1^!, \ldots, q_h^!\}$ in $\hat{\mathcal{G}}_1^{i_0}$.

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

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

in $\mathcal{G}_1^i$ with $b \in \Sigma$, we have the following transition in $G_{(q,Q')}^{i_0}$,

$$g_{(q,Q')}^{i_1} \xrightarrow{b}_{i_0} \{q_1^!, \ldots, q_h^!\}$$

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

- If $\alpha_e = \theta$, then let $g = q^\theta$. We have $g \xrightarrow{b}_{i-1} Q_e$ exists in $\mathcal{G}_1^{i-1}$. By induction over i we have $g^! \xrightarrow{b}_{i_0} Q_e^!$ in $\hat{\mathcal{G}}_1^{i_0}$.

- $\alpha_e = (a, push_{w_p}, \theta)$. Then $b = a$. Let $g = q^\theta$. By definition of $T_{\widetilde{\mathcal{G}}^{i_1}[i_1/i_1-1]}$, we have the path $g \xrightarrow{w_p}_{i-1} Q_e$ in $\mathcal{G}_1^i$. By induction on i we have the path $g^! \xrightarrow{w_p}_{i_0} Q_e^!$ in $\hat{\mathcal{G}}_1^{i_0}$.

We have $Q_1^! \cup \cdots \cup Q_m^! = \{q_1^!, \ldots, q_h^!\}$. Since $\widetilde{G}_{(q,Q')}^i \simeq \widetilde{G}_{(q,Q')}^{i_1}$ and $S[i_1/i-1] \in \widetilde{G}_{(q,Q')}^{i_1}[i_1/i_1-1]$, by definition of $\hat{\mathcal{G}}_1^{i_0}$, we have,

$$g_{(q,Q')}^{i_1} \xrightarrow{b}_{i_0} \{q_1^!, \ldots, q_h^!\}$$

in $\hat{\mathcal{G}}_1^{i_0}$ 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,

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

in $\mathcal{G}_1^i$. For each $y \in \{1, \ldots, h_1\}$ we have a run $q_y^1 \xrightarrow{a_1 \ldots a_m}_i Q_y$ such that $\bigcup_{y \in \{1, \ldots, h_1\}} Q_y = \{q_1^m, \ldots, q_{h_m}^m\}$. By induction over the length of the run we have $q_y^{!1} \xrightarrow{a_1 \ldots a_m}_{i_0} Q_y^!$ for each y. Hence, since we have $g_{(q,Q')}^{i_1} \xrightarrow{a_0}_{i_0} \{q_1^{!1}, \ldots, q_{h_1}^{!1}\}$ from the above proof for one transition, we have a run of the form,

$$g_{(q,Q')}^{i_1} \xrightarrow{a_0}_{i_0} \{q_1^{!1}, \ldots, q_{h_1}^{!1}\} \xrightarrow{a_1}_{i_0} \cdots \xrightarrow{a_m}_{i_0} \{q_1^{!m}, \ldots, q_{h_m}^{!m}\}$$

in $\hat{\mathcal{G}}_1^{i_0}$ as required. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

Lemma B.11 *For all w, if we have $g_{(q,Q')}^{i_1} \xrightarrow{w}_i Q_f$ with $Q_f \subseteq \mathcal{Q}_f$ in $\hat{\mathcal{G}}_1^i$ for some i, then there is some i' such that the run $g_{(q,Q')}^{i'} \xrightarrow{w}_{i'} Q_f$ exists in $\mathcal{G}_1^{i'}$.*

Proof.

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

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

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

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

53

in $G^{i^2}_{(q,Q')}$ where, for $y \in \{1, \ldots, h\}$,

$$q^?_y = \begin{cases} g^{i^1}_{(q',Q'')} & \text{if } q_1 = g^{i^1}_{(q',Q'')} \\ q_y & \text{otherwise} \end{cases}$$

Since $q^?_f = q_f$ for all $q_f \in \mathcal{Q}_f$, the lemma follows. For a set $Q = \{q_1, \ldots, q_h\}$ we write $Q^? = \{q^?_1, \ldots, q^?_h\}$.

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

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

$$g^{i_1}_{(q,Q')} \xrightarrow{b}_i \{q_1, \ldots, q_h\}$$

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

$$g^{i^2}_{(q,Q')} \xrightarrow{b}_{i^2} \{q^?_1, \ldots, q^?_h\}$$

in $G^{i^2}_{(q,Q')}$. We analyse the $S \in \widetilde{G}^{i_1}_{(q,Q')}[i_1/i_1 - 1]$ that spawned the transition from $g^{i_1}_{(q,Q')}$ (we assume the transition is new, else the property holds by induction). Let $S = \{\alpha_1, \ldots, \alpha_m\}$. We have $\{q_1, \ldots, q_h\} = Q_1 \cup \cdots \cup Q_m$. For each $e \in \{1, \ldots, m\}$, there are several cases,

- $\alpha_e = \theta$.

  Let $g_e = q^\theta$. By definition of $\hat{\mathcal{G}}^i_1$ we have the transition $g_e \xrightarrow{b}_{i-1} Q_e$ in $\hat{\mathcal{G}}^{i-1}_1$.

  If $\theta = \widetilde{G}^{i_1}_{(q',Q'')}$ then by induction we have $i^2_e > i^1$ such that $g^{i^2_e}_{(q',Q'')} \xrightarrow{b}_{i^2_e} Q^?_e$ in $\mathcal{G}^{i^2_e}_1$.

  Otherwise g_e is initial in some $B \in \mathcal{B}_1$ and the transition $g_e \xrightarrow{b}_{i-1} Q_e$ also exists in $\mathcal{G}^0_1$ and is the same as $g_e \xrightarrow{b}_0 Q^?_e$. Let $w_e = b$.

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

  Let $g_e = q^\theta$. By definition of $\hat{\mathcal{G}}^i_1$ we have the run $g_e \xrightarrow{w_p}_{i-1} Q_e$ in $\hat{\mathcal{G}}^{i-1}_1$.

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

  Otherwise g_e is initial in some $B \in \mathcal{B}_1$ and the transition $g_e \xrightarrow{w_p}_{i-1} Q_e$ also exists in $\mathcal{G}^0_1$ and is the same as $g_e \xrightarrow{w_p}_0 Q^?_e$. Let $w_e = w_p$.

Let i_{max} be the maximum i^2_e. If $g_e = g^{i_1}_{(q',Q'')}$, we have, by Lemma B.2, $g^{i_{max}}_{(q',Q'')} \xrightarrow{w_e}_{i_{max}} Q^?_e$. Also, by Lemma B.2 we have $g_e \xrightarrow{w_e}_{i_{max}} Q^?_e$ when g_e is not of the form $g^{i_1}_{(q',Q'')}$. Since we have $\widetilde{G}^{i_{max}+1}_{(q,Q')} \simeq \widetilde{G}^{i_1}_{(q,Q')}$ we have $S[i_{max}/i_1 - 1] \in \widetilde{G}^{i_{max}+1}_{(q,Q')}$ and since $Q^?_1 \cup \cdots \cup Q^?_m = \{q^?_1, \ldots, q^?_h\}$ we have,

$$g^{i_{max}+1}_{(q,Q')} \xrightarrow{b}_{i_{max}+1} \{q^?_1, \ldots, q^?_h\}$$

in $G^{i_{max}+1}_{(q,Q')}$. 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 $\hat{\mathcal{G}}^i_{(q,Q')}$,

$$g^{i_1}_{(q,Q')} \xrightarrow{w}_i \{q_1, \ldots, q_h\}$$

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

$$g^{i^2}_{(q,Q')} \xrightarrow{w}_{i^2} \{q^?_1, \ldots, q^?_h\}$$

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

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

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

We have $q_y^1 \xrightarrow{a_1 \ldots a_m}_i Q_y$ for each $y \in \{1, \ldots, h_1\}$ and $\bigcup_{y \in \{1, \ldots, h_1\}} Q_y = \{q_1^m, \ldots, q_{h_m}^m\}$. Then for all $y \in \{1, \ldots, h_1\}$ via induction and Lemma B.8 we have for all $i^1 > i_1$ an i_{max} with

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

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,

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

exists in $\mathcal{G}_1^{i^2}$ as required. □

We are now ready to prove the desired property.

Corollary B.1 *Suppose we have constructed a sequence of automata $\mathcal{G}_1^0, \mathcal{G}_1^1, \ldots$ and associated sets $\widetilde{\mathcal{G}}_1^0, \widetilde{\mathcal{G}}_1^1, \ldots$. Further, suppose there exists an i_1 such that for all $i \geq i_1$ we have $\widetilde{\mathcal{G}}_1^i \simeq \widetilde{\mathcal{G}}_1^{i_1}$. We can define a sequence of automata $\hat{\mathcal{G}}_1^{i_1}, \hat{\mathcal{G}}_1^{i_1+1}, \ldots$ such that the state-set in $\hat{\mathcal{G}}_1^i$ remains constant and there exists i_0 such that $\hat{\mathcal{G}}_1^{i_0}$ 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 $\hat{\mathcal{G}}_1^i$ 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 $\hat{\mathcal{G}}_1^{i_0}$.*

3. *The run $g_{(q,Q')}^{i'} \xrightarrow{w}_{i'} Q_3$ with $Q_3 \subseteq \mathcal{Q}_f$ exists in $\mathcal{G}_1^{i'}$ for some i'.*

Proof. Follows from the definition of $\hat{\mathcal{G}}_1^{i+1} = T_{\widetilde{\mathcal{G}}_1^{i_1}[i_1/i_1-1]}(\hat{\mathcal{G}}_1^i)$, Lemma B.9, Lemma B.10 and Lemma B.11. □

B.5.2 Proof of Lemma 4.1 for $l > 1$

In this section we prove Lemma 4.1 for the case when $l > 1$. The structure of the proof is similar to the previous section.

Lemma B.12 *For all w, if $g_{(q,Q')}^i \xrightarrow{w}_i Q_f$ with $Q_f \subseteq \mathcal{Q}_f$ is a run in $\mathcal{G}_l^i$ for some i, then we have $g_{(q,Q')}^{i_l} \xrightarrow{w}_{i'} Q_f$ in $\hat{\mathcal{G}}_l^{i'}$ for some i'.*

Proof. We prove the following property. For any path $g_{(q,Q')}^i \xrightarrow{w}_{i'} \{q_1, \ldots, q_h\}$ we can construct a path $g_{(q,Q')}^{i_l} \xrightarrow{w}_{i'} \{q_1^!, \ldots, q_h^!\}$ with,

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

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

There are two cases. When $i \leq i_l$, using Lemma B.2, that $\mathcal{G}_l^{i_l} = \hat{\mathcal{G}}_l^{i_l}$ and that $q_y^! = q_y$ for all y, we have $g_{(q,Q')}^{i_l} \xrightarrow{w'}_{i_l} \{q_1^!, \ldots, q_h^!\}$ in $\hat{\mathcal{G}}_l^{i_l}$.

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

$$g^i_{(q,Q')} \xrightarrow{\gamma}_i \{q_1, \ldots, q_h\}$$

in $\mathcal{G}^i_l$, we have the following transition in $\hat{G}^{i'}_{(q,Q')}$ for some i',

$$g^{i_l}_{(q,Q')} \xrightarrow{\gamma}_{i'} \{q^!_1, \ldots, q^!_h\}$$

We consider the $S = \{\alpha_1, \ldots, \alpha_m\} \in \widetilde{G}^i_{(q,Q')}$ from which the transition from $g^i_{(q,Q')}$ was derived (if it was inherited, then the property holds by induction over i). We have $\{q_1, \ldots, q_h\} = Q_1 \cup \cdots \cup Q_m$. For each $e \in \{1, \ldots, m\}$ there are several cases,

- $\alpha_e = \theta$.

  Let $g_e = q^\theta$. We have $g_e \xrightarrow{\theta'}_{i-1} Q_e$ in $\mathcal{G}^{i-1}_l$ with $\theta' \in S' \in \widetilde{G}^i_{(g^i_{(q,Q')}, \{q_1, \ldots, q_h\})}$ and the accepting run of γ derived from S'. By Lemma B.4 we have $\gamma \in \mathcal{L}(\theta')$. Hence, we have $g_e \xrightarrow{\gamma}_{i-1} Q_e$ in $\mathcal{G}^{i-1}_l$. By induction over i, we have $g^!_e \xrightarrow{\gamma}_{i'_e} Q^!_e$ in $\hat{G}^{i'_e}_{(q,Q')}$ for some i'_e. Furthermore, let $w_e = \gamma$.

- $\alpha_e = (a, push_l, \theta)$.

  Let $g_e = q^\theta$. We have the path

  $$g_e \xrightarrow{\theta_1}_{i-1} Q \xrightarrow{\widetilde{\theta}_2}_{i-1} Q_e$$

  in $\mathcal{G}^{i-1}_l$ with $\{B^a_{l-1}, \theta_1\} \cup \widetilde{\theta}_2 \subseteq S' \in \widetilde{G}^i_{(g^i_{(q,Q')}, \{q_1, \ldots, q_h\})}$ and the accepting run of γ is derived from S'. By Lemma B.4 we have $\gamma \in \mathcal{L}(\{B^a_{l-1}, \theta_1\} \cup \widetilde{\theta}_2)$. Hence, we have the run,

  $$g_e \xrightarrow{\gamma}_{i-1} Q \xrightarrow{\gamma}_{i-1} Q_e$$

  By induction over i, we have,

  $$g^!_e \xrightarrow{\gamma\gamma}_{i'_e} Q^!_e$$

  for some i'_e. Furthermore, let $w_e = \gamma\gamma$.

- $\alpha_e = (a, pop_l, \theta)$.

  Let $g_e = q^\theta$. We have $Q_e = \{g_e\}$.

  Additionally, we have $B^a_{l-1} \in S' \in \widetilde{G}^i_{(g^i_{(q,Q')}, \{q_1, \ldots, q_h\})}$ and the accepting run of γ derived from S'. By Lemma B.4 we have $\gamma \in \mathcal{L}(B^a_{l-1})$. Furthermore, let $w_e = \varepsilon$.

- $\theta = (a, o, \theta)$ where $\ell(d) < l$.

  Let $g_e = q^{\theta_x}$. We have $g_e \xrightarrow{\theta'}_{i-1} Q_e$ in $\mathcal{G}^{i-1}_l$ with $(a, o, \theta') \in S' \in \widetilde{G}^i_{(g^i_{(q,Q')}, \{q_1, \ldots, q_h\})}$ and the accepting run of γ derived from S'. By Lemma B.5 we have $o(\gamma) \in \mathcal{L}(\theta')$. Hence, we have $g_e \xrightarrow{o(\gamma)}_{i-1} Q_e$ in $\mathcal{G}^{i-1}_l$. By induction over i, we have $g^!_e \xrightarrow{o(\gamma)}_{i'_e} Q^!_e$ in $\hat{G}^{i'_e}_{(q,Q')}$ for some i'_e. Furthermore, let $w_e = o(\gamma)$.

Let i_{max} be the maximum i'_e. By Lemma B.2 we have $g^!_e \xrightarrow{w_e}_{i_{max}} Q^!_e$ for all e. Since $S[i_{max}/i-1] \in \widetilde{G}^{i_{max}+1}_{(q,Q')}$ and $Q^!_1 \cup \cdots \cup Q^!_m = \{q^!_1, \ldots, q^!_h\}$ we have via Lemma B.6,

$$g^{i_1}_{(q,Q')} \xrightarrow{\gamma}_{i_{max}+1} \{q^!_1, \ldots, q^!_h\}$$

in $\hat{G}^{i_{max}+1}_{(q,Q')}$ 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,

$$g^i_{(q,Q')} \xrightarrow{\gamma_0}_i \{q^1_1, \ldots, q^1_{h_1}\} \xrightarrow{\gamma_1}_i \ldots \xrightarrow{\gamma_m}_i \{q^m_1, \ldots, q^m_{h_m}\}$$

For each $y \in \{1, \ldots, h_1\}$ we have a run $q^1_y \xrightarrow{\gamma_1 \ldots \gamma_m}_i Q_y$ such that $\bigcup_{y \in \{1, \ldots, h_1\}} Q_y = \{q^m_1, \ldots, q^m_{h_m}\}$. By induction on the length of the run we have $q^{!1}_y \xrightarrow{\gamma_1 \ldots \gamma_m}_{i'} Q^!_y$ for each y for some i'. Let i_{max} be the maximum i'. We also have $g^{i_1}_{(q,Q')} \xrightarrow{\gamma_0}_{i'} \{q^{!1}_1, \ldots, q^{!1}_{h_1}\}$ for some i' from the above proof for one transition. We have, via Lemma B.8, a run of the form,

$$g^{i_1}_{(q,Q')} \xrightarrow{\gamma_0}_{i_{max}} \{q^{!1}_1, \ldots, q^{!1}_{h_1}\} \xrightarrow{\gamma_1}_{i_{max}} \ldots \xrightarrow{\gamma_m}_{i_{max}} \{q^{!m}_1, \ldots, q^{!m}_{h_m}\}$$

in $\hat{\mathcal{G}}^{i_{max}}_l$ as required. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Lemma B.13 *For all w, if we have $g^{i_l}_{(q,Q')} \xrightarrow{w}_i Q_f$ with $Q_f \subseteq \mathcal{Q}_f$ in $\hat{\mathcal{G}}^i_l$ for some i, then there is some i' such that the run $g^{i'}_{(q,Q')} \xrightarrow{w}_{i'} Q_f$ exists in $\mathcal{G}^{i'}_l$.*

Proof.

We take a run of $\hat{\mathcal{G}}^i_{(q,Q')}$,

$$g^{i_l}_{(q,Q')} \xrightarrow{w}_i \{q_1, \ldots, q_h\}$$

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

$$g^{i^2}_{(q,Q')} \xrightarrow{w}_{i^2} \{q^?_1, \ldots, q^?_h\}$$

in $G^{i^2}_{(q,Q')}$ where, for $y \in \{1, \ldots, h\}$,

$$q^?_y = \begin{cases} g^{i^1}_{(q',Q'')} & \text{if } q_1 = g^{i_l}_{(q',Q'')} \\ q_y & \text{otherwise} \end{cases}$$

Since $q^?_f = q_f$ for all $q_f \in \mathcal{Q}_f$, the lemma follows. Given a set $Q = \{q_1, \ldots, q_m\}$, we write $Q^?$ to denote the set $\{q^?_1, \ldots, q^?_m\}$.

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

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

$$g^{i_l}_{(q,Q')} \xrightarrow{\gamma}_i \{q_1, \ldots, q_h\}$$

in $\hat{G}^i_{(q,Q')}$ we have, for all $i^1 \geq i_l$, there is some $i^2 > i^1$ such that,

$$g^{i^2}_{(q,Q')} \xrightarrow{\gamma}_{i^2} \{q^?_1, \ldots, q^?_h\}$$

in $G^{i^2}_{(q,Q')}$. We analyse the $S \in \widetilde{G}^{i_l}_{(q,Q')}[i_l/i_l - 1]$ from which the transition from $g^{i_l}_{(q,Q')}$ was derived (we assume the transition is not inherited, else the property holds by induction).

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

- $\alpha_e = \theta$.

  Let $g_e = q^\theta$. We have $g_e \xrightarrow{\theta'}_{i-1} Q_e$ in $\hat{\mathcal{G}}_l^{i-1}$ with $\theta' \in S' \in \widetilde{G}^i_{(g^{i_l}_{(q,Q')},\{q_1,\ldots,q_h\})}$ and the accepting run of γ derived from S'. By Lemma B.4 we have $\gamma \in \mathcal{L}(\theta')$. Hence, we have $g_e \xrightarrow{\gamma}_{i-1} Q_e$ in $\hat{\mathcal{G}}_l^{i-1}$.

  If $\theta = \widetilde{G}^{i_l}_{(q',Q'')}$, by induction over i, we have $i_e^2 > i^1$ such that $g^{i_e^2}_{(q',Q'')} \xrightarrow{\gamma}_{i_e^2} Q_e^?$ in $G^{i_e^2}_{(q,Q')}$. Otherwise g_e is initial in some $B \in \mathcal{B}_l$ and the transition $g_e \xrightarrow{\gamma}_{i-1} Q_e$ also exists in $\mathcal{G}_l^0$ and is the same as $g_e \xrightarrow{\gamma}_0 Q_e^?$. Let $w_e = \gamma$.

- $\alpha_e = (a, push_l, \theta)$.

  Let $g_e = q^\theta$. We have the path

  $$g_e \xrightarrow{\theta_1}_{i-1} Q \xrightarrow{\widetilde{\theta}_2}_{i-1} Q_e$$

  in $\hat{\mathcal{G}}_l^{i-1}$ with $\{B_{l-1}^a, \theta_1\} \cup \widetilde{\theta}_2 \subseteq S' \in \widetilde{G}^i_{(g^{i_l}_{(q,Q')},\{q_1,\ldots,q_h\})}$ and the accepting run of γ is derived from S'. By Lemma B.4 we have $\gamma \in \mathcal{L}(\{B_{l-1}^a, \theta_1\} \cup \widetilde{\theta}_2)$. Hence, we have the run,

  $$g_e \xrightarrow{\gamma}_{i-1} Q \xrightarrow{\gamma}_{i-1} Q_e$$

  If $\theta = \widetilde{G}^{i_l}_{(q',Q'')}$, by induction over i, we have $i_e^2 > i^1$ such that,

  $$g^{i_e^2}_{(q',Q'')} \xrightarrow{\gamma\gamma}_{i_e^2} Q_e^?$$

  Otherwise g_e is initial in some $B \in \mathcal{B}_l$ and the transition $g_e \xrightarrow{\gamma\gamma}_{i-1} Q_e$ also exists in $\mathcal{G}_l^0$ and is the same as $g_e \xrightarrow{\gamma\gamma}_0 Q_e^?$. Let $w_e = \gamma\gamma$.

- $\alpha_e = (a, pop_l, \theta)$.

  Let $g_e = q^\theta$. We have $Q_e = \{g_e\}$. Additionally, we have $B_{l-1}^a \in S' \in \widetilde{G}^i_{(g^{i_l}_{(q,Q')},\{q_1,\ldots,q_h\})}$ and the accepting run of γ derived from S'. By Lemma B.4 we have $\gamma \in \mathcal{L}(B_{l-1}^a)$. Furthermore, let $w_e = \varepsilon$.

- $\theta = (a, o, \theta)$ where $\ell(d) < l$.

  Let $g_e = q^\theta$. We have $g_e \xrightarrow{\theta'}_{i-1} Q_e$ in $\hat{\mathcal{G}}_l^{i-1}$ with $(a, o, \theta') \in S' \in \widetilde{G}^i_{(g^{i_l}_{(q,Q')},\{q_1,\ldots,q_h\})}$ and the accepting run of γ derived from S'. By Lemma B.5 we have $o(\gamma) \in \mathcal{L}(\theta')$. Hence, we have $g_e \xrightarrow{o(\gamma)}_{i-1} Q_e$ in $\hat{\mathcal{G}}_l^{i-1}$.

  When $\theta = \widetilde{G}^{i_l}_{(q',Q'')}$, we have by induction over i some $i_e^2 > i^1$ such that $g^{i_e^2}_{(q',Q'')} \xrightarrow{o(\gamma)}_{i_e^2} Q_e^?$ in $\hat{G}^{i_e^2}_{(q,Q')}$. Otherwise g_e is initial in some $B \in \mathcal{B}_l$ and the transition $g_e \xrightarrow{o(\gamma)}_{i-1} Q_e$ also exists in $\mathcal{G}_l^0$ and is the same as $g_e \xrightarrow{o(\gamma)}_0 Q_e^?$. Let $w_e = o(\gamma)$.

Let i_{max} be the maximum i_e^2. If $g_e = g^{i_l}_{(q',Q'')}$, we have, by Lemma B.2, $g^{i_{max}}_{(q',Q'')} \xrightarrow{w_e}_{i_{max}} Q_e^?$. Also, by Lemma B.2 we have $g_e \xrightarrow{w_e}_{i_{max}} Q_e^?$ when g_e is not of the form $g^{i_l}_{(q',Q'')}$. Since we have $\widetilde{G}^{i_{max}+1}_{(q,Q')} \simeq \widetilde{G}^{i_l}_{(q,Q')}$ we have $S[i_{max}/i_l] \in \widetilde{G}^{i_{max}+1}_{(q,Q')}$ and since $Q_1^? \cup \cdots \cup Q_m^? = \{q_1^?, \ldots, q_h^?\}$ we have via Lemma B.6,

$$g^{i_{max}+1}_{(q,Q')} \xrightarrow{\gamma}_{i_{max}+1} \{q_1^?, \ldots, q_h^?\}$$

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

58

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

$$g^{i_l}_{(q,Q')} \xrightarrow{w}_i \{q_1, \ldots, q_h\}$$

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

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

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

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

$$g^{i_l}_{(q,Q')} \xrightarrow{\gamma_0}_i \{q_1^1, \ldots, q_{h_1}^1\} \xrightarrow{\gamma_1}_i \ldots \xrightarrow{\gamma_m}_i \{q_1^m, \ldots, q_{h_m}^m\}$$

We have $q_y^1 \xrightarrow{\gamma_1 \ldots \gamma_m}_i Q_y$ for each $y \in \{1, \ldots, h_1\}$ and $\bigcup_{y \in \{1, \ldots, h_1\}} Q_y = \{q_1^m, \ldots, q_{h_m}^m\}$. Then for all $y \in \{1, \ldots, h_1\}$ we have via induction and Lemma B.8 we have for all $i^1 > i_l$ an i_{max} with

$$\{q_1^{?1}, \ldots, q_{h_1}^{?1}\} \xrightarrow{\gamma_1 \ldots \gamma_m}_{i_{max}} \{q_1^{?m}, \ldots, q_{h_m}^{?m}\}$$

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,

$$g^{i^2}_{(q,Q')} \xrightarrow{\gamma_0}_{i^2} \{q_1^{?1}, \ldots, q_{h_1}^{?1}\} \xrightarrow{a_1 \ldots a_m}_{i^2} \{q_1^{?m}, \ldots, q_{h_m}^{?m}\}$$

as required. □

Corollary B.2 *Suppose we have constructed a sequence of automata $\mathcal{G}_l^0, \mathcal{G}_l^1, \ldots$ and associated sets $\widetilde{\mathcal{G}}_l^0, \widetilde{\mathcal{G}}_l^1, \ldots$. Further, suppose there exists an i_l such that for all $i \geq i_l$ we have $\widetilde{\mathcal{G}}_l^i \simeq \widetilde{\mathcal{G}}_l^{i_l}$. We can define a sequence of automata $\hat{\mathcal{G}}_l^{i_l}, \hat{\mathcal{G}}^{i_1+1}, \ldots$ such that the state-set in $\hat{\mathcal{G}}_l^i$ remains constant (although the automata labelling the transitions may gain states). The following are equivalent for all w,*

1. *The run $g^{i_l}_{(q,Q')} \xrightarrow{w}_i Q_1$ with $Q_1 \subseteq \mathcal{Q}_f$ exists in $\hat{\mathcal{G}}_l^i$ for some i.*

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

Proof. From Lemma B.12 and Lemma B.13. □

B.5.3 Proof of Lemma 4.2

Finally, we show that the constructed automaton $\mathcal{G}_l^*$ is correct.

Lemma 4.2 1 *Suppose we have constructed a sequence of automata $\mathcal{G}_l^0, \mathcal{G}_l^1, \ldots$ and associated sets $\widetilde{\mathcal{G}}_l^0, \widetilde{\mathcal{G}}_l^1, \ldots$. Further, suppose there exists an i_l such that for all $i \geq i_l$ we have $\widetilde{\mathcal{G}}_l^i \simeq \widetilde{\mathcal{G}}_l^{i_l}$. We can define an automaton $\mathcal{G}_l^*$ such that the following are equivalent for all w,*

1. *The run $g^*_{(q,Q')} \xrightarrow{w}_* Q$ with $Q \subseteq \mathcal{Q}_f$ exists in $\mathcal{G}_l^*$.*

2. *The run $g^i_{(q,Q')} \xrightarrow{w}_i Q''$ with $Q'' \subseteq \mathcal{Q}_f$ exists in $\mathcal{G}_l^i$ for some i.*

Proof. We proceed by induction over l. In the base case $l = 1$ and the result follows from Corollary B.1. That is, $\mathcal{G}_1^* = \mathcal{G}_1^{i_0}$ letting $g^*_{(q,Q)} = g^{i_1}_{(q,Q)}$ for each $g^{i_1}_{(q,Q)}$. The equivalence of (1) and (2) is immediate.

When $l > 1$ we generate the sequence $\hat{\mathcal{G}}_l^{i_l}, \hat{\mathcal{G}}_l^{i_l+1}, \ldots$ by Corollary B.2. Since the state-set remains constant, it follows from Property 4.3 that there is some i_{l-1} with $\widetilde{G}_{l-1}^i \simeq \widetilde{G}_{l-1}^{i_{l-1}}$ for all $i \geq i_{l-1}$.

By induction, we have $\mathcal{G}_{l-1}^*$. We then define $\mathcal{G}_l^*$ from $\hat{\mathcal{G}}_l^{i_{l-1}}$ with $g_{(q,Q')}^* = g_{(q,Q')}^{i_l}$ for all q, Q' and each transition $q \longrightarrow_* Q'$ in $\mathcal{G}_l^*$ labelled with the automaton $G_{(q,Q')}^*$ from $\mathcal{G}_{l-1}^*$.

We show (1) and (2) are equivalent. By Lemma 4.1 (2) is equivalent to a run $g_{(q,Q')}^{i_l} \xrightarrow{w}_{i'} Q_1$ with $Q_1 \subseteq \mathcal{Q}_f$ in $\hat{\mathcal{G}}_l^{i'}$ for some i'.

We proceed by induction over the length of w. Note that $g_{(q,Q')}^{i_l} = g_{(q,Q')}^*$. We prove that if $w \neq \varepsilon$ a run $g_{(q,Q')}^i \xrightarrow{w}_* Q$ exists in $\mathcal{G}_l^*$ iff a run $g_{(q,Q')}^i \xrightarrow{w}_{i'} Q$ exists in $\hat{\mathcal{G}}_l^{i'}$ for some i'. It is necessarily the case that $i \leq i_l$. The Lemma follows because a state of the form $g_{(q',Q'')}^i$ is never accepting and therefore $w \neq \varepsilon$.

In the base case, let $w = \gamma$. Let $g = g_{(q,Q')}^i$. To prove (1) implies (2) we assume the transition $g \xrightarrow{G_{(g,Q)}^*}_* Q$ in $\mathcal{G}_l^*$ with $\gamma \in G_{(g,Q)}^*$. By induction over l we have $\gamma \in \hat{G}_{(g,Q)}^{i'}$ for some i'. Hence, we have $g \xrightarrow{\gamma}_{i'} Q$ in $\hat{\mathcal{G}}_l^{i'}$ as required.

To show (2) implies (1) we assume the transition $g \xrightarrow{\hat{G}_{(g,Q_1)}^{i'}}_{i'} Q_1$ in $\hat{\mathcal{G}}_l^{i'}$ for some i' with $\gamma \in \hat{G}_{(g,Q_1)}^{i'}$. By induction over l we have $\gamma \in G_{(g,Q_1)}^*$. Hence, we have $g \xrightarrow{\gamma}_* Q_1$ in $\mathcal{G}_l^*$ as required. This completes the proof of the base case.

For the induction, let $w = \gamma w'$ and $w' \neq \varepsilon$. To show (2) follows from (1) assume we have a transition $g \xrightarrow{G_{(g,Q)}^*}_* Q = \{q_1, \ldots, q_m\}$ with $\gamma \in G_{(g,Q)}^*$ in $\mathcal{G}_l^*$ and the run,

$$\{q_1, \ldots, q_m\} \xrightarrow{w'}_* Q_f$$

in $\mathcal{G}_l^*$. As before, we have $g \xrightarrow{\gamma}_{i'} Q$ in $\hat{\mathcal{G}}_l^{i'}$ for some i'. For all $e \in \{1, \ldots, m\}$, we have a run $q_e \xrightarrow{w'}_* Q_f^e$ with $Q_f = Q_f^1 \cup \cdots \cup Q_f^m$. If q_e is of the form $g_{(q',Q'')}^j$ for some j, then by induction we have $g_{(q',Q'')}^j \xrightarrow{w'}_{i'} Q_f^e$ in $\hat{\mathcal{G}}_l^{i'}$ for some i' by induction over the length of the word. Otherwise the run over w' uses only states and transitions left unchanged by the algorithm. Hence we have $q_e \xrightarrow{w'}_{i_l} Q_f^e$ in $\hat{\mathcal{G}}_l^{i_l}$. As required, by Lemma B.8 we have in $\hat{\mathcal{G}}_l^{i_{max}}$ for some i_{max},

$$g \xrightarrow{\gamma}_{i_{max}} \{q_1, \ldots, q_m\} \xrightarrow{w'}_{i_{max}} Q_f^1 \cup \cdots \cup Q_f^m$$

To show (1) follows from (2) assume we have a transition $g \xrightarrow{\hat{G}_{(g,Q)}^{i'}}_{i'} Q = \{q_1, \ldots, q_m\}$ with $\gamma \in \hat{G}_{(g,Q)}^{i'}$ in $\hat{\mathcal{G}}_l^{i'}$ and the run,

$$\{q_1, \ldots, q_m\} \xrightarrow{w'}_{i'} Q_f$$

in $\hat{\mathcal{G}}_l^{i'}$ for some i'. As in the base case, we have $g \xrightarrow{\gamma}_* Q$ in $\mathcal{G}_l^*$. For all $e \in \{1, \ldots, m\}$, we have a run $q_e \xrightarrow{w'}_{i'} Q_f^e$ with $Q_f = Q_f^1 \cup \cdots \cup Q_f^m$. If q_e is of the form $g_{(q',Q'')}^j$ for some j, then by induction we have $g_{(q',Q'')}^j \xrightarrow{w'}_* Q_f^e$ in $\mathcal{G}_l^*$ by induction over the length of the word. Otherwise the run over w' uses only states and transitions left unchanged by the algorithm. Hence, since $\mathcal{G}_l^*$ is derived from $\hat{\mathcal{G}}_l^{i_{l-1}}$, we have $q_e \xrightarrow{w'}_* Q_f^e$ in $\mathcal{G}_l^0$. Subsequently, we have,

$$g \xrightarrow{\gamma}_* \{q_1, \ldots, q_m\} \xrightarrow{w'}_* Q_f^1 \cup \cdots \cup Q_f^m$$

in $\mathcal{G}_l^*$ as required. □

C Applications: Proofs and Definitions

C.1 Proof of Proposition 5.1

Proof. $\Rightarrow$: 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 $\rho = c_0 c_1 \ldots$, there exists a sequence of suffixes $w_1, w_2, \ldots$ such that there exists an increasing sequence of natural numbers $i_1, i_2, \ldots$ 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 $c_{i_1} c_{i_2} \ldots$. 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 $p^f \in \mathcal{F}$,

$$c_0 \overset{*}{\hookrightarrow} c_{i_a} \overset{*}{\hookrightarrow} c_f \overset{*}{\hookrightarrow} c_{i_b}$$

We have (1) from $c_0 \overset{*}{\hookrightarrow} c_{i_a}$. By definition of $c_{i_1}, c_{i_2} \ldots$ 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,

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

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

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

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

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. $\square$

C.2 Proof of Lemma 5.1

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

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

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

  such that p_m is the only accepting control state in the run. This run is derived from a sequence of commands $d_1, \ldots, d_m$. Let $d_i = (p_{i-1}, a_i, o_i, p_i)$ for all $i \in \{1, \ldots, m\}$. We show the run,
$$\langle (p_0, 0), \gamma_0 \rangle \hookrightarrow \ldots \hookrightarrow \langle (p_m, 0), \gamma_m \rangle$$

  exists in BP' by induction over m. In the base case $m = 0$ and the result is trivial. Suppose we have,
$$\langle (p_1, 0), \gamma_1 \rangle \hookrightarrow \ldots \hookrightarrow \langle (p_m, 0), \gamma_m \rangle$$

  by the induction hypothesis. Since $d_1 = (p_0, a_1, o_1, p_1)$ and $p_0 \notin \mathcal{F}$, we have $((p_0, 0), a_1, o_1, (p_1, 0))$ in $\mathcal{D}'$. Hence we have the run,
$$\langle (p_0, 0), \gamma_0 \rangle \hookrightarrow \ldots \hookrightarrow \langle (p_m, 0), \gamma_m \rangle$$

  as required.

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

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

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

  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 $d_1, \ldots, d_m$. 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 $i \in \{2, \ldots, m\}$ we have $((p_{i-1}, 1), a_i, o_i, (p_i, 1))$ in $\mathcal{D}'$. We have the run

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

  in BP' therefrom.

The proof of this direction follows immediately.

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

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

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 $d_1, \ldots, d_m$ where $m \geq 1$. From a simple projection on the first element of each control state/flag pair, we immediately derive a sequence commands $d'_1, \ldots, d'_m$ in $\mathcal{D}$ and the following run of BP,

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

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).

Since BP' is twice as large as BP, $Pre^*(\{p^j\} \times \mathcal{L}(B_n^a))$ can be calculated in n-EXPTIME. $\square$

C.3 Proof of $Attr_E(\mathcal{R}) = Pre^*(\mathcal{R}') \setminus \mathcal{C}_A^\triangledown$

Proof. Let $\mathcal{R}' = \mathcal{R} \cup \mathcal{R}_{stuck}$. Since the size of an n-store multi-automaton recognising $\mathcal{R}_{stuck}$ is linear, the complexity follows from the complexity of computing $Pre^*(\mathcal{R}')$.

We show $Attr_E(\mathcal{R}) = Pre^*(\mathcal{R}') \setminus \mathcal{C}_A^\triangledown$. We begin by proving $Attr_E(\mathcal{R}) \supseteq Pre^*(\mathcal{R}') \setminus \mathcal{C}_A^\triangledown$.

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

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

Now, suppose we have $\langle p, \gamma \rangle \hookrightarrow C$ via the command $d = (p, a, OP)$ in the higher-order APDS with $C \in Pre^*(\mathcal{R}) \setminus \mathcal{C}_A^\triangledown$ and by induction $C \subseteq Attr_E^i(\mathcal{R})$ 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 $\langle p', \gamma' \rangle \in Pre^*(\mathcal{R}') \setminus \mathcal{C}_A^\triangledown$ or we have $\langle p', \gamma' \rangle = \langle p, \triangledown \rangle$.

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

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

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

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

Thus, we have $Attr_E(\mathcal{R}) \supseteq Pre^*(\mathcal{R}') \setminus \mathcal{C}_A^\triangledown$ as required.

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

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

  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 $c' \in Pre^*(\mathcal{R}') \setminus \mathcal{C}_A^\triangledown$ and $c = \langle p, \gamma \rangle$ and $c' = \langle p', o(\gamma) \rangle$. Hence $c \in Pre^*(\mathcal{R}') \setminus \mathcal{C}_A^\triangledown$.

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

  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 $\langle p', o(\gamma) \rangle \in Pre^*(\mathcal{R}')$ by induction. If $o(\gamma)$ is undefined, then since we have $\langle p, \triangledown \rangle \in \mathcal{R}'$ we have $\langle p, \triangledown \rangle \in Pre^*(\mathcal{R}')$.

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

Thus, we have $Attr_E(\mathcal{R}) = Pre^*(\mathcal{R}') \setminus \mathcal{C}_A^\triangledown$. □

C.4 Non-emptiness of Order-$(n+1)$ Pushdown Automata

Property C.1 *If $\mathcal{L}(P) \neq \emptyset$ then $\langle (p_0, \otimes), \perp_n \rangle \in Attr_E(\mathcal{R})$.*

Proof. Assume there exists $w = \alpha_1 \ldots \alpha_m \in \mathcal{L}(P)$. Since $w \in \mathcal{L}(P)$ there exists a run,

$$\langle p_0, \gamma_0 \rangle \overset{\alpha_1}{\hookrightarrow} \ldots \overset{\alpha_m}{\hookrightarrow} \langle p_m, \gamma_m \rangle$$

with $\gamma_0 = \perp_{n+1}$, $p_m \in \mathcal{P}_f$ and the corresponding sequence of commands $d_1, \ldots, d_m$ such that for each $1 \leq i \leq m$ we have $d_i = (p_{i-1}, \alpha_i, a_i, o_i, p_i)$ with $top_i(\gamma_{i-1}) = a_i$ and $\gamma_i = o_i(\gamma_{i-1})$.

We show by induction over the number of order-$(n+1)$ commands in the sequence $d_1, \ldots, d_m$ that Eloise has a winning strategy in the game P_G. In particular we show that if there exists a run,

$$\langle p_0, \gamma_0 \rangle \overset{\alpha_1}{\hookrightarrow} \ldots \overset{\alpha_m}{\hookrightarrow} \langle p_m, \gamma_m \rangle$$

for arbitrary p_0 and γ_0 — with the further condition that if $o_i = pop_{n+1}$ for some i, then there exists $i' < i$ with $o_{i'} = push_{n+1}$ — then from a configuration $\langle (p_0, p_r), top_{n+1}(\gamma_0) \rangle$ of P_G Eloise has a strategy to reach the configuration $\langle (p_m, p_r), top_{n+1}(\gamma_m) \rangle$ or win the game. For convenience we define the abbreviation $\gamma^n = top_{n+1}(\gamma)$ for all γ.

In the base case, no order-$(n+1)$ commands appear in the sequence $d_1, \ldots, d_m$. That is $o_i \in \mathcal{O}_n$ for all $1 \leq i \leq m$. Therefore, from the configuration $\langle (p_0, p_r), \gamma_0^n \rangle$ Eloise can play the sequence of moves $d'_1, \ldots, d'_m$ where $d'_i = ((p_{i-1}, p_r), a_i, o_i, (p_i, p_r))$ and reach the configuration $\langle (p_m, p_r), \gamma_m^n \rangle$.

In the inductive case we have $o_i = push_{n+1}$ for some i such that for all $i' < i$, $o_{i'} \in \mathcal{O}_n$. By induction Eloise can force play into the configuration $\langle (p_{i-1}, p_r), \gamma_{i-1}^n \rangle$ or win the game. In the former case there are two further cases to consider. Note that $\gamma_{i-1}^n = \gamma_i^n$.

- There is no $i' \geq i$ such that $\gamma_{i-1} = \gamma_{i'}$. That is, the top stack added by the push command is never removed. In this case Eloise moves play to the configuration $\langle (p_i, \otimes, p_r), \gamma_i^n \rangle$ via the command $d'_i = ((p_{i-1}, p_r), a, push_a, (p_i, \otimes, p_r))$ where $a = top_1(\gamma_{i-1})$. From this configuration Abelard may only move play to the configuration $\langle (p_i, \otimes), \gamma_i^n \rangle$. By induction, Eloise has a strategy as required from this configuration.

- There is some $i' \geq i$ such that $\gamma_{i-1} = \gamma_{i'}$. Take the least such i'. We have $o_{i'} = pop_{n+1}$. Eloise moves play to the configuration $\langle (p_i, p_{i'}, p_r), \gamma_i^n \rangle$ via the command $d'_i = ((p_{i-1}, p_r), a, push_a, (p_i, p_{i'}, p_r))$ where $a = top_1(\gamma_{i-1})$. There are two further subcases:

63

- Abelard moves play to the configuration $\langle(p_i, p_{i'}), \gamma_i^n\rangle$. By induction, Eloise can either win from this configuration, or force play to the configuration $\langle(p_{i'-1}, p_{i'}), \gamma_{i'-1}^n\rangle$. In the latter case, since $o_{i'} = pop_{n+1}$, Eloise moves play to $\langle\texttt{tt}, \gamma_{i'-1}^n\rangle$ and wins the game.

- Abelard moves play to the configuration $\langle(p_{i'}, p_r), \gamma_i^n\rangle = \langle(p_{i'}, p_r), \gamma_{i'}^n\rangle$. By induction Eloise can either win from this configuration or force play to the configuration $\langle(p_m, p_r), \gamma_m^n\rangle$ as required.

Since any run from $\langle p_0, \perp_{n+1}\rangle$ must perform a $push_{n+1}$ before a pop_{n+1} can occur, it is the case that from $\langle(p_0, \otimes), \perp_{n+1}\rangle$ Eloise can either win the game or force play to the configuration $\langle(p_m, \otimes), \gamma_m^n\rangle$. Since $p_m \in \mathcal{P}_f$, this is a victory for Eloise. $\qquad\square$

Property C.2 *If* $\langle(p_0, \otimes), \perp_n\rangle \in Attr_E(\mathcal{R})$ *then* $\mathcal{L}(P) \neq \emptyset$.

Proof. Assume that Eloise has a winning strategy from the configuration $\langle(p_0, \otimes), \perp_n\rangle$ in the game P_G. We begin by proving that there is some bound k on the length of any play of P_G played according to Eloise's strategy. We then show that a word w can be constructed such that $w \in \mathcal{L}(P)$.

To show there is some bound k on the length of any play of P_G we observe that if $\langle(p_0, \otimes), \perp_n\rangle \in Attr_E(\mathcal{R})$ then $\langle(p_0, \otimes), \perp_n\rangle \in Attr_E^i(\mathcal{R})$ for some i. We proceed by induction over i. We show for any $c \in Attr_E^i(\mathcal{R})$, there is a bound on the length of any play according to Eloise's winning strategy.

When $i = 0$ then $c \in \mathcal{R}$ and $k = 0$. For $i + 1$ then in both cases $c \in \mathcal{P}'_E$ or $c \in \mathcal{P}'_A$ play moves to a configuration $c' \in Attr_E^i(\mathcal{R})$. By induction there is some bound k on the length of any play according to Eloise's strategy from c'. Hence, we have a bound $k + 1$ on the length of any play from c. Thus we have a bound on the length of any play from $\langle(p_0, \otimes), \perp_n\rangle$ as required.

We now show that there exists a word w such that $w \in \mathcal{L}(P)$. We proceed by induction over k. In particular, we prove the following result: if Eloise has a winning strategy from a configuration $\langle(p, p_r), \gamma\rangle$, then there exists some $w \in \Gamma^*$ with, for all γ' such that $top_{n+1}(\gamma') = \gamma$ and — if $p_r \neq \otimes$ — $pop_{n+1}(\gamma')$ is defined, a run of P of the form $\langle p, \gamma'\rangle \xrightarrow{w} \langle p', \gamma''\rangle$ where either $p' = p_r$, $\gamma'' = pop_{n+1}(\gamma')$ or $p' \in \mathcal{P}_f$ and $p_r = \otimes$.

In the base case $k = 0$. That is $p \in \mathcal{P}_f$ and $p_r = \otimes$. The property holds trivially. For $k + 1$ there are several cases depending on Eloise's next move d.

- $d = ((p, p_r), a, o, (p', p_r))$ and $o \in \mathcal{O}_n$.

  By definition of $\mathcal{D}'$, there exists some command $(p, \alpha, a, o, p') \in \mathcal{D}$. Hence, the transition $\langle p, \gamma'\rangle \xrightarrow{\alpha} \langle p', o(\gamma')\rangle$ exists in P.

  After Eloise moves, play continues from the configuration $\langle(p', p_r), o(\gamma)\rangle$ and $top_{n+1}(o(\gamma')) = o(\gamma)$. By induction over k there exists $w' \in \Gamma^*$ such that P has the run $\langle p', o(\gamma')\rangle \xrightarrow{w'} \langle p'', \gamma''\rangle$ satisfying the induction hypothesis. Hence, the run $\langle p, \gamma'\rangle \xrightarrow{\alpha w'} \langle p'', \gamma''\rangle$ exists in P and satisfies the induction hypothesis.

- $d = ((p, p_r), a, push_a, \texttt{tt})$.

  By definition of $\mathcal{D}'$, there exists some command $(p, \alpha, a, pop_{n+1}, p_r) \in \mathcal{D}$. Hence, because $p_r \in \mathcal{P}$ and thus $p_r \neq \otimes$, the transition $\langle p, \gamma'\rangle \xrightarrow{\alpha} \langle p_r, \gamma''\rangle$ exists in P where $top_{n+1}(\gamma') = \gamma$ and $\gamma'' = pop_{n+1}(\gamma')$.

- $d = ((p, p_r), a, push_a, \texttt{ff})$.

  In this case, play moves to the position $\langle\texttt{ff}, \gamma\rangle$, which is a loss for Eloise. Since Eloise's strategy is winning, this case cannot occur.

- $d = ((p, p_r), a, push_a, (p', p'_r, p_r))$.

64

By definition of $\mathcal{D}'$, there exists some command $(p, \alpha, a, push_{n+1}, p') \in \mathcal{D}$. Hence, the transition $\langle p, \gamma' \rangle \xrightarrow{\alpha} \langle p', push_{n+1}(\gamma') \rangle$ exists in P. Note that $top_{n+1}(\gamma') = top_{n+1}(push_{n+1}(\gamma')) = \gamma$.

Play moves to the configuration $\langle (p', p'_r, p_r), \gamma \rangle$. Eloise's strategy must accommodate both of Abelard's possible replies. In the case where Abelard moves play to the configuration $\langle (p', p_r), \gamma \rangle$, since Eloise's strategy is winning we have by induction $w' \in \Gamma'$ such that $\langle p', push_{n+1}(\gamma') \rangle \xrightarrow{w'} \langle p'', \gamma'' \rangle$ exists and satisfies the induction hypothesis.

If $p'' \in \mathcal{P}_f$ and $p_r = \otimes$, then we have the run $\langle p, \gamma' \rangle \xrightarrow{\alpha w'} \langle p'', \gamma'' \rangle$ and the induction hypothesis is satisfied.

Otherwise we have a run $\langle p, \gamma' \rangle \xrightarrow{\alpha w'} \langle p_r, \gamma' \rangle$. In the case where Abelard moves play to the configuration $\langle (p'_r, p_r), \gamma \rangle$, we have by induction a run $\langle p_r, \gamma' \rangle \xrightarrow{w''} \langle p''', \gamma''' \rangle$ for some w'' which satisfies the induction hypothesis. We consequently have a run $\langle p, \gamma' \rangle \xrightarrow{\alpha w' w''} \langle p''', \gamma''' \rangle$ as required.

Thus, since Eloise has a winning strategy from the configuration $\langle (p_0, \otimes), \perp_n \rangle$, we have a run $\langle p_0, \perp_{n+1} \rangle \xrightarrow{w} \langle p, \gamma \rangle$ for some w, p and γ. Since $\otimes \notin \mathcal{P}$, we must have $p \in \mathcal{P}_f$. Thus, $\mathcal{L}(P) \neq \emptyset$ as required. $\qquad\square$