An Alternative Construction in Symbolic Reachability Analysis of Second Order Pushdown Systems

Anil Seth

CSE Department, I.I.T. Kanpur, Kanpur 208016, INDIA.
seth@cse.iitk.ac.in

Abstract. Recently, it has been shown that for any higher order pushdown system $\mathcal{H}$ and for any regular set $\mathcal{C}$ of configurations, the set $pre^*_{\mathcal{H}}(\mathcal{C})$, is regular. In this paper, we give an alternative proof of this result for second order automata. Our construction of automata for recognizing $pre^*_{\mathcal{H}}(\mathcal{C})$ is explicit. The termination of saturation procedure used is obvious. It gives a better bound on size of the automata recognizing $pre^*_{\mathcal{H}}(\mathcal{C})$ if there is no alternation present in $\mathcal{H}$ and in the automata recognizing $\mathcal{C}$. Using our techniques for two players reachability games on second order pushdown systems, we generalize some result of [2] concerning synthesis of strategies. Analogous to [2], we show two kinds of winning strategies for player 0 and give algorithms to compute them. The first is executable by second order pushdown automata and the second is linear time executable minimum cost strategy.

1 Introduction

Higher order pushdown automata (*hpda*) are a generalization of pushdown automata in that they can have nested stacks, such as stack of stacks. Higher order Push and Pop operations are provided to push a copy of the topmost stack of any order and to pop it. Order of a *hpda* depends on the depth of nested stacks allowed by a *hpda* .

These models were introduced in [7] and were further studied in [9, 8]. Higher order pushdown systems (*hpds*) are *hpda* without any input. A configuration of a *hpds* is a pair of its control state and its stack contents. A *hpds* can be viewed as a transition system on the infinite set of configurations. In the last few years, algorithmic properties of finitely presented infinite graphs have been extensively investigated by verification community, see [5, 11, 6]. Here we survey a few results which are of direct relevance to our work.

In [4], it was shown that for any pushdown system (order-1 *hpds*), and any regular set C of its configurations, $pre^*(C)$, the set of configurations from which an element of C is reachable, is also regular. [4] gave an iterative procedure, called saturation procedure, to construct an automata recognizing $pre^*(C)$, starting from the automata recognizing C. The iteration step adds new edges to the automata but no new states.

In [3], Bouajjani and Meyer, extended this result to higher order context free processes, which are higher order pushdown systems with a single control state. They have introduced a structured way to represent a regular set of configurations of a *hpds* . For example a regular set of order-2 PDS configurations is given by a automata with finitely many states but whose edge labels are ordinary finite automata recognising order-1 stack configurations. Authors of [3] have generalized the saturation procedure to these automata to prove their results. Recently, Hague and Ong [1] have extended results of [3] to higher order pushdown systems. This is a technically non-trivial result. Let A be a order-2 finite automata recognizing a set of order-2 PDS configurations C. The saturation procedure of [1] uses nested recursion involving automata appearing as edge labels of A and the structure of A. As new states are added during this construction termination of the recursion is not immediate and requires a proof. [1] gives details of all this.

In this paper, we give an alternative construction of an automata recognizing $pre^*(C)$ for second order *hpds* . The state set of our automata to compute $pre^*(C)$ is fixed beforehand and remains the same throughout saturation procedure, so termination of saturation procedure is immediate. Further each state has some intuitive meaning and saturation procedure is reduced essentially to order-1 saturation procedure. Our result was arrived at independently of [1].

Unlike, [3, 1], we represent a regular set of order-2 *hpda* configurations by ordinary automata. This is no restriction because as pointed out in [3] the two notions of regularity for representing stack configurations are in fact the same. In fact, given a (deterministic/nondeterministic) finite automata as in [3], one can convert it into a (deterministic/nondeterministic) ordinary automata in polynomial time.

In [1], size of the automata recognising $pre^*(C)$ (and the time to construct it) is double exponential in the size of *hpds* $\mathcal{H}$ (of order-2) and the size of automata $\mathcal{A}$, recognising C. The authors in [1], in fact solve the more general case of 2-player (or alternating) *hpds* with the same bounds. Further, in [1], $\mathcal{A}$, recognising C is also assumed to be alternating. Our bounds match theirs in the general case. We consider the special case when $\mathcal{H}$ is nondeterministic and $\mathcal{A}$ in given as a non-deterministic automata, in this case we show that an automata recognising $pre^*(C)$ can be constructed in time single exponential in the size of *hpds* $\mathcal{H}$ and size of automata $\mathcal{A}$. In case of 2-player (or alternating) *hpds* and nondeterministic $\mathcal{A}$ our construction takes double exponential time which matches the bound in [1], in fact in this case we construct an alternating automata with single exponential states to recognise attractor set of C.

In [2], results of [4] are used to give uniform winning strategies in two player reachability games on order-1 *hpds* . Two kinds of strategies are given in [2], the first is minimum cost positional strategy executable in linear time and the second is computable by a order-1 *hpds* . We generalize these results to order-2 *hpds* reachability games. We show a minimum cost positional strategy executable in linear time and a strategy executable by second order pushdown automata in order-2 *hpds* reachability games.

2 Preliminaries

A higher order pushdown system (*hpds*) $\mathcal{H}$ is a triple (Q, Γ, Δ), where Q is a finite set of control states, Γ is a (finite) stack alphabet and $\Delta \subseteq Q \times \Gamma \times Act$ is transition relation of $\mathcal{H}$. For an order-2 *hpds* , the set of actions is defined as $Act = \{push_1^{q,w}, push_2^q, pop_i^q \mid i \in \{1,2\}, q \in Q, w \in \Gamma^*\}$.

Configurations of a *hpds* are pairs (q, s), where q is a control state of the *hpds* and s is its stack configuration. The set of stack configurations of the ordinary pushdown automata (order-1 *hpds*) denoted by S_1 and the set of stack configurations of an order-2 *hpds* denoted by S_2 are defined as follows.

$S_1 = \{\ [_1\ w\ ]_1 \mid w \in \Gamma^*\}$, $\quad S_2 = \{\ [_2\ \alpha\ ]_2 \mid \alpha \in (S_1)^*\}$

Throughout this paper we use symbol $[_i$ to denote start of stack of order i and $]_i$ for the end of stack of order i. Let $push_1^w, push_2, pop_1, pop_2$ stand for stateless counterparts of actions $push_1^{q,w}, push_2^q, pop_1^q, pop_2^q$ respectively. These stateless counterparts act on stack configurations as follows.

$push_1^w([_1\ a_1 \cdots a_l]_1) = [_1\ wa_2 \cdots a_l]_1$,

$push_1^w([_2\ s_1 \cdots s_k\ ]_2) = [_2\ push_1^w(s_1)s_2 \cdots s_k\ ]_2$

$push_2([_2\ s_1 \cdots s_k]_2) = [_2\ s_1 s_1 s_2 \cdots s_k]_2$

$pop_1([_2\ s_1 \cdots s_k]_2) = [_2\ pop_1(s_1) \cdots s_k]_2$,

$pop_m([_m\ s_1 \cdots s_k]_m) = [_m\ s_2 \cdots s_k]_m$, if $k \geq 1$ else undefined, $m \in \{1,2\}$.

We also define topmost element, $top(s)$, of a stack configuration s, as below.

$top([_1\ a_1 \cdots a_l]_1) = a_1$, $top([_2\ s_1 \cdots s_k]_2) = top(s_1)$

Transition relation $\hookrightarrow_{\mathcal{H}}$ is defined on configurations of a *hpds* of order 2 as follows.

$(q', s) \hookrightarrow_{\mathcal{H}} (q, Push_1^w(s))$ if $(q', top(s), Push_1^{q,w}) \in \Delta$

$(q', s) \hookrightarrow_{\mathcal{H}} (q, Push_2(s))$ if $(q', top(s), Push_2^q) \in \Delta$,

$(q', s) \hookrightarrow_{\mathcal{H}} (q, Pop_i(s))$ if $(q', top(s), Pop_i^q) \in \Delta$, $i \in \{1,2\}$.

Let S be a set of stack configurations of *hpds* $\mathcal{H}$, $pre^*_{\mathcal{H}}(S)$ is defined as $pre^*_{\mathcal{H}}(S) = \{s \mid \exists s' \in S[s \hookrightarrow_{\mathcal{H}} s']\}$. We omit the subscript $\mathcal{H}$ from $\hookrightarrow_{\mathcal{H}}$ whenever it is clear from the context. $\hookrightarrow^*$ stands for the reflexive, transitive closure of $\hookrightarrow$.

We consider regular subsets of *hpds* configurations. Let $\mathcal{H} = (Q, \Gamma, \Delta)$ be a *hpds* of order-2. We define $Q' = \{q' \mid q \in Q\}$. A finite (multi)automata for recognising configurations of $\mathcal{H}$ is given as $\mathcal{A} = (S_{\mathcal{A}}, \Gamma_2, Q', \delta, F)$, where $S_{\mathcal{A}}$ is the set of states of $\mathcal{A}$, $\Gamma_2 = \Gamma \cup \{\ [_i,]_i \mid 1 \leq i \leq 2\}$ is the input alphabet for $\mathcal{A}$. δ is the transition relation of $\mathcal{A}$, Q' is the set of its initial states and F is the set of final states of $\mathcal{A}$. Multi-automata were first introduced in [4] for order-1 PDS.

For any automata $\mathcal{B}$ and a state q of it, we use the notation $\mathcal{L}(\mathcal{B}, q)$ to denote the set of strings accepted by $\mathcal{B}$ when started in state q. The set of configurations, $C_{\mathcal{A}}$, accepted by multi-automata $\mathcal{A}$ above is given as
$C_{\mathcal{A}} = \mathcal{L}(\mathcal{A}) = \{(q, s) \mid s \in \mathcal{L}(\mathcal{A}, q')\}$.

2.1 Alternating Automata

An alternating automata R is given as (P, Σ, δ, F), where P is a finite set of states Σ is input alphabet and F is the set of final states. Transition function δ of R is given as $\delta : P \times \Sigma \to B^+(P)$, where $B^+(P)$ is the set of positive boolean

formulae (constructed using connectives $\wedge$ and $\vee$ only) over atoms in P. As any function in $B^+(P)$, can be written as a disjunction of conjunctions of elements in P, transition function can also be thought of as a relation on $P \times \Sigma \times 2^P$. In other words, it can be seen as a union of transitions of the form (q, a, S), where $q \in P$, $a \in \Sigma$ and $S \subseteq P$.

A run of an alternating automata on a input is a tree. More efficiently we can represent it as a dag as in [2, 10]. Since we allow ϵ transitions (and later transitions on a string of letters instead of a single letter), all paths through this dag may not be of equal length. We give a slightly modified definition of a run dag useful for our purposes.

A run-dag of automata $\mathcal{B}$ on input w is a rooted dag in which each node is labeled with a state of $\mathcal{B}$ and each edge is labeled with an input symbol or input string including empty string ϵ. If a node is labeled with q then all outgoing edges from q in the dag are labeled by the same string u. Further, if the set of labels of nodes to which these edges are connected is $\{q_{i_1}, \ldots, q_{i_k}\}$ then there must be a one step transition $(q, u, \{q_{i_1}, \ldots, q_{i_k}\}) \in \delta$ The concatenation of all labels in any maximal path (path which is not contained in a bigger path) through this dag should be a prefix of the string w, such that if it is a proper prefix of w then its terminal node is labeled with a constant *true* or *false*.

A run dag is *accepting* if all its terminal nodes are labeled with 'true' or with final states of $\mathcal{B}$.

We extend the function/relation δ to take its second argument as a string instead of a single letter as follows. $(p, w, S) \in \delta$ if there is a run dag on input w with root labeled as p and its leaves are labeled with either 'true' or with an element in S.

3 Computing pre^* for Hpds of order-2

Let $\mathcal{H} = (Q, \Gamma, \Delta)$ be an order-2 *hpds* . We define $Q' = \{q' \mid q \in Q\}$.
Let $\mathcal{A} = (S_{\mathcal{A}}, \Gamma_2, Q', \delta_{\mathcal{A}}, \mathcal{F})$ be a deterministic finite multi-automata with Q' as initial states. Let $\mathcal{A}$ accept a set $\mathcal{C}_{\mathcal{A}}$ of configurations of $\mathcal{H}$. It is assumed w.l.o.g. that sets Q and $S_{\mathcal{A}}$ are mutually disjoint.

We design an alternating finite automata $\mathcal{R}_{\mathcal{H},\mathcal{A}}$ to accept $pre^*_{\mathcal{H}}(\mathcal{C}_{\mathcal{A}})$.
$\mathcal{R}_{\mathcal{H},\mathcal{A}} = (S_{\mathcal{R}}, \Gamma_2, \delta_{\mathcal{R}}, \mathcal{F} \times \{\downarrow\})$
State set of $R_{\mathcal{H},\mathcal{A}}$, denoted $S_{\mathcal{R}}$, is a disjoint union of several sets as below.
Let $Q'' = \{q'' \mid q \in Q\}$, $\quad U_{Q,\mathcal{A},\downarrow} = Q \cup S_{\mathcal{A}} \cup \{\downarrow\}$, $\quad U_{\mathcal{A},\downarrow} = S_{\mathcal{A}} \cup \{\downarrow\}$
[To remember notation U can be thought of as union]
$R_1 = (Q \times U_{Q,\mathcal{A},\downarrow}) \cup (S_{\mathcal{A}} \times U_{\mathcal{A},\downarrow})$
$R_2 = \{s_w \mid s \in R_1, (w \in \Gamma) \text{ or } (Push_1^{q,w} \in \Delta, \text{ for some } q \in Q)\}$
$S_{\mathcal{R}} = Q'' \cup R_1 \cup R_2 \cup (Q \times \{2\})$

It will be shown that $w \in \mathcal{L}(\mathcal{R}_{\mathcal{H},\mathcal{A}}, q_i'')$ iff $(q_i'', w) \in pre^*_{\mathcal{H}}(\mathcal{C}_{\mathcal{A}})$. We use below $\mathcal{R}$ instead of $\mathcal{R}_{\mathcal{H},\mathcal{A}}$ when no confusion occurs.

We define the transition function $\delta_{\mathcal{R}}$ as $\bigcup_{i \geq 0} \delta_i$, where δ_i, $i = 0, 1, 2, \ldots$ are defined below iteratively. The sequence δ_i, $i = 0, 1, 2, \ldots$ is monotone when

viewed as a relation on $S_{\mathcal{R}} \times \Gamma_2 \times B^+(S_{\mathcal{R}})$. This part of the construction is called saturation procedure.

3.1 Intuitive Idea

The idea behind the construction is to start with the transition group (0). The automata now looks at the top of the stack symbol and sees which moves in *hpds* are possible. The $push_1$ and pop_1 move are handled as in order-1 case. Pop_2 move is simply handled by ignoring the input till the end of current order-1 stack. The $push_2$ operation is to be handled differently as it makes two copies of the current order-1 stack while there is only one copy in the input. To do so, we simultaneously verify the operation on the top stack of the configuration and the stack below. For this we need to know in which state is the top stack popped? This is done by guessing this state. So at a $push_2$ move the computation splits into two threads one thread verifies that the current stack can be popped in some state q and the other thread starts running on the current input with this state. The verifying thread will die on meeting the end of current order-1 stack either successfully or unsuccessfully.

The computation in the two threads is similar in many cases so we use unifying notation for states involving pairs. States $(p, \downarrow)$, through the use of $\downarrow$ indicates main thread which is to continue beyond the current order-1 stack. Other pairs denote threads which check constraints. Note both these type of threads can spawn further threads.

In this way we will be able to keep information about *hpds* configurations that can be reached. At each step, the automata has a choice to explore the next configuration reached or check if the current configuration is in the target set, whose pre^* is being computed. These are the main ideas, the construction below shows how these can be made to work.

3.2 Transitions in δ_0

The transitions of $\mathcal{R}$ below are grouped according to transitions in *hpds* $\mathcal{H}$. To improve readability of triples below, we show $S_{\mathcal{R}} \times \Gamma_2$ part of the triple in lightface and $B^+(S_{\mathcal{R}})$ component in boldface.

(0). For $q \in Q$, $(q^{''}, [_2[_1, (q, \downarrow)) \in \delta_0$.

(i). $(\mathbf{q}, \mathbf{a}, \mathbf{pop_1^p}) \in \mathbf{\Delta}$ then for $t \in Q \cup S_{\mathcal{A}} \cup \{\downarrow\}$, $((q, t), a, (\mathbf{p}, \mathbf{t})) \in \delta_0$.

(ii). $(\mathbf{q}, \mathbf{a}, \mathbf{push_2^p}) \in \mathbf{\Delta}$

(a)

$$\text{then for each } \; t \in Q, \quad ((q, t), a, \bigvee_{\mathbf{q_l} \in \mathbf{Q}} [(\mathbf{p}, \mathbf{q_l})_{\mathbf{a}} \wedge (\mathbf{q_l}, \mathbf{t})_{\mathbf{a}}]) \in \delta_0,$$

(b)

$$\text{and for each } t \in U_{\mathcal{A},\downarrow}, \quad ((q, t), a, \bigvee_{\mathbf{t_1} \in \mathbf{Q} \cup \mathbf{S_{\mathcal{A}}}} [(\mathbf{p}, \mathbf{t_1})_{\mathbf{a}} \wedge (\mathbf{t_1}, \mathbf{t})_{\mathbf{a}}]) \in \delta_0$$

/* See Lemma 1(5-7), for the intuitive meaning of states $(p, q_l)_a$ etc. */

(iii). $(\mathbf{q}, \mathbf{a}, \mathbf{pop_2^p}) \in \mathbf{\Delta}$ then we have the following triples in δ_0

(a) $((q, \downarrow), a, (\mathbf{p}, \mathbf{2}))$,

(b) $((r, 2), b, (\mathbf{r}, \mathbf{2}))$, $r \in Q, b \in \Gamma$ /* skip the current order-1 stack */

(c) $((r, 2),]_1[_1, (\mathbf{r}, \downarrow))$, /* beginning of the next order-1 stack reached */

(d) $((r, 2),]_1]_2, (\delta_{\mathcal{A}}(\mathbf{r'}, [_\mathbf{2}]_\mathbf{2}), \downarrow))$, /* this corresponds to the case when the popped stack was the last one */

(e) $((q, p), a, \mathbf{true})$ /* popping the stack in state q on 'a' satisfies the constraint (q, p) */

(iv). $(\mathbf{q}, \mathbf{a}, \mathbf{push_1^{p,u}}) \in \mathbf{\Delta}$ then for $t \in U_{Q,\mathcal{A},\downarrow}$, $((q, t), a, (\mathbf{p}, \mathbf{t})_\mathbf{u}) \in \delta_0$

(v). Transitions related to $\mathcal{A}$

/* We have the following transitions for simulating automata $\mathcal{A}$ on a guessed *hpds* configuration. */

(a) $(q'', [_2, (\delta_{\mathcal{A}}(\mathbf{q'}, [_\mathbf{2}), \downarrow))$, for $q \in Q$.
/* The guessed configuration is the initial configuration */

(b) $((q, t), \epsilon, (\delta_{\mathcal{A}}(\mathbf{q'}, [_\mathbf{2}[_\mathbf{1}), \mathbf{t}))$, for all $q \in Q, t \in U_{\mathcal{A},\downarrow}$.
/* The automata $\mathcal{R}$ guesses a configuration */

(c) $((t_1, t_2), a, (\delta_{\mathcal{A}}(\mathbf{t_1}, \mathbf{a}), \mathbf{t_2}))$, $\quad for\ (t_1 \in S_{\mathcal{A}}, t_2 = \downarrow, a \in \Gamma \cup \{[_1,]_1\})$
$OR \quad for\ (t_1 \in S_{\mathcal{A}}, t_2 \in S_{\mathcal{A}}, a \in \Gamma)$.
/* Simulates $\mathcal{A}$ in the first component */

(d) For $t_2 \in S_{\mathcal{A}}$,

$$((t_1, t_2),]_1, true) \in \delta_0 \quad \text{if } \delta_{\mathcal{A}}(t_1,]_1) = t_2$$
$$((t_1, t_2),]_1, false) \in \delta_0 \quad \text{otherwise}$$

/* accepts if the constraint is satisfied at the end of current order-1 stack */

(vi). Transitions from states in R_2

$((t_1, t_2)_a, \epsilon, (\delta_{\mathcal{A}}(\mathbf{t_1}, [_\mathbf{1}\mathbf{a}), \mathbf{t_2}))$, for all $t_1 \in S_{\mathcal{A}}, t_2 \in U_{\mathcal{A},\downarrow}, a \in \Gamma$.

3.3 Saturation Step

Saturation process is as follows.

$\delta_{k+1} := \delta_k \cup V$, where

$V = \{((p, t)_x, \epsilon, S) \mid p \in Q, t \in U_{Q,\mathcal{A},\downarrow}, ((p, t), x, S) \in \delta_k, (p, t)_x \in R_2\}$

For each k, $\delta_k \subseteq S_{\mathcal{R}} \times \Gamma_2 \times 2^{S_{\mathcal{R}}}$. As $(\delta_k)_{k \geq 0}$ is a monotonically increasing sequence, the saturation procedure terminates in at most $|S_{\mathcal{R}}| \times |\Gamma_2| \times 2^{|S_{\mathcal{R}}|}$ iterations.

3.4 Correctness of the Construction

The following lemma easily follows from the construction.

Lemma 1. *1. For $t \in S_{\mathcal{A}}$, $w \in \mathcal{L}(\mathcal{R}, (t, \downarrow))$ iff $w \in \mathcal{L}(\mathcal{A}, t)$.*

2. For $t_1, t_2 \in S_{\mathcal{A}}$, $w \in \mathcal{L}(\mathcal{R}, (t_1, t_2))$ iff $w = x]_1 v$, $x \in \Gamma^*$ and $\delta_{\mathcal{A}}(t_1, x]_1) = t_2$.

3. $w \in \mathcal{L}(\mathcal{R},(q,2))$ *iff* $w = x]_1[_1 u$ *for* $x \in \Gamma^*$ *and* $u \in \mathcal{L}(\mathcal{R},(q,\downarrow))$ *or* $w = x]_1]_2$ *for* $x \in \Gamma^*$ *and* $[_2]_2 \in \mathcal{L}(\mathcal{A},q')$.
4. *For* $q \in Q$, $w \in \mathcal{L}(\mathcal{R},q'')$ *iff* $w = [_2[_1 v$ *for some* v, *and* $v \in \mathcal{L}(\mathcal{R},(q,\downarrow))$ *or* $w \in \mathcal{L}(\mathcal{A},q')$.
5. *For* $r \in Q \times (Q \cup S_\mathcal{A} \cup \{\downarrow\})$ *and* $r_u \in R_2$, $w \in \mathcal{L}(\mathcal{R},r_u)$ *iff* $uw \in \mathcal{L}(\mathcal{R},r)$.
6. *For* $t \in S_\mathcal{A}$, $a \in \Gamma$, $w \in \mathcal{L}(\mathcal{R},(t,\downarrow)_a)$ *iff* $[_1 aw \in \mathcal{L}(\mathcal{A},t)$.
7. *For* $(t_1,t_2) \in S_\mathcal{A} \times S_\mathcal{A}$, $a \in \Gamma$, $w \in \mathcal{L}(\mathcal{R},(t_1,t_2)_a)$ *iff* $w = x]_1 v$, $x \in \Gamma^*$ *and* $\delta_\mathcal{A}(t_1,[_1 ax]_1) = t_2$.

Proof. We omit easy details from this extended abstract. □

The theorem below is at the heart of the correctness proof.

Theorem 1. *Let* $\hookrightarrow_*$ *be the reachability relation on configurations of hpds* $\mathcal{H}$. *The following assertions hold for all* $p,q \in Q$, $t \in S_\mathcal{A}$, $v,w \in \Gamma_2^*$.

1. *For* $q \in Q$, $w \in \mathcal{L}(\mathcal{R},(q,\downarrow))$ *iff there is a* $\tau \in \mathcal{L}(\mathcal{A})$ *such that* $(q,[_2[_1 w) \hookrightarrow^* \tau$.
2. *For* $p,q \in Q$, $w \in \mathcal{L}(\mathcal{R},(p,q))$ *iff* $w = x]_1 u$, *for some* $x \in \Gamma^*$, *and* $(p,[_2[_1 x]_1]_2) \hookrightarrow^* (q,[_2]_2)$.
3. *For* $p \in Q$ *and* $t \in S_\mathcal{A}$, $w \in \mathcal{L}(\mathcal{R},(p,t))$ *iff* $w = x]_1 u$, *for some* $x \in \Gamma^*$, *and* $(p,[_2[_1 x]_1]_2) \hookrightarrow^* (r,[_2[_1 v]_1]_2)$ *and* $\delta_\mathcal{A}(r,[_2[_1 v]_1) = t$.

Proof. This is proved in two parts. We omit the details here which may be found in full version. □

Combining Lemma 1 part (4) and Theorem 1 part (1), we immediately have the following corollary.

Corollary 1. *For* $q \in Q$, $w \in \mathcal{L}(\mathcal{R},q'')$ *iff* $w \in pre^*(C_\mathcal{A})$.

We have presented the construction for deterministic $\mathcal{A}$, however essentially the same construction works for non-deterministic $\mathcal{A}$ also. The only change is that we replace $\delta_\mathcal{A}(r,a)$ by $\vee$ of states instead of a single state. More precisely, transitions (iii.d), (v) and (vi) only need to be modified. We omit easy details.

Note that the situation changes if $\mathcal{A}$ is alternating. In that case, the form of pairs (q,t) is to be replaced by (q,T), where $T \subseteq S_\mathcal{A}$. This is because a path in the simulation of $\mathcal{A}$ is given by a set of states. Consequently the number of states in the automata becomes exponential in $S_\mathcal{A}$.

Corollary 2. *Let* $\mathcal{H}$ *be a nondeterministic hpds with* $|Q|$ *states and* $|\Delta|$ *transitions and let* $\mathcal{A}$ *be a nondeterministic automaton with* $|S_\mathcal{A}|$ *states. Then one can effectively construct a finite alternating automata with* $O((|Q|+|S_\mathcal{A}|)^2 \cdot |\Delta|)$ *states to recognize* $pre^*(C_\mathcal{A})$.

From the above alternating automata recognizing $pre^*(C_\mathcal{A})$, we can get by standard construction a equivalent nondeterministic automata with 2^z states, where z is a polynomial in the size of $\mathcal{H}$ and $\mathcal{A}$. This is one exponential less than what the construction of [1] gives though the construction of [1] is optimal if $\mathcal{H}$ and $\mathcal{A}$ are alternating.

4 Two Player Reachability Game over Hpds

In rest of the paper, we extend our techniques to study two player reachability games over configuration graphs of second order *hpds* . Two player games in general are an important model of reactive computation and have been widely studied in the context of verification and synthesis of finite/infinite state systems over the last few years, see [12, 13].

We first recall some preliminary notions about these games. A game structure can be imposed on the configuration graph of a *hpds* by partitioning the states of the *hpds* into two parts. $\mathcal{H} = (Q_0 \oplus Q_1, \Gamma, \Delta)$ is a game structure where states in Q_m, $m \in \{0, 1\}$, correspond to player m.

A position in such a game is a configuration of $\mathcal{H}$. A position belongs to player m if control state in this position (configuration) $\in Q_m$. Starting from a initial position π_0 a play proceeds as follows, if π_0 belongs to player-m then player-m chooses a position π_1 such that $\pi_0 \hookrightarrow_{\mathcal{H}} \pi_1$. The play now enters position π_1 and continues this way, at stage i if position π_i belongs to player-m, $m \in \{0, 1\}$, then player-m chooses a position π_{i+1} such that $\pi_i \hookrightarrow_{\mathcal{H}} \pi_{i+1}$. A play is a sequence (possibly infinite) $\pi_0\pi_1 \cdots \pi_i \cdots$ of successive game positions.

In a reachability game for player k, $k \in \{0, 1\}$, a set of game positions (configurations of *hpds*) E is also given. Player-k wins a play in this game if a position $\in E$ is reached during this play otherwise player-$(1 - k)$ wins.

A strategy for player m, $m \in \{0, 1\}$, from position π_0, is a function which associates to each prefix $\pi_0\pi_1 \cdots \pi_i$ of a play, where π_i belongs to player m, a position π_{i+1} such that $\pi_i \hookrightarrow_{\mathcal{H}} \pi_{i+1}$. A strategy σ for player-m from position π is a winning strategy if in any play begining with π where player-m plays according to σ, player-m wins the play. A position π is winning for player-m if there is a winning strategy for player-m from position π. The set of all winning positions of player-m is called winning region or player-m.

It is well known that the winning region of player-k in the reachability game for player-k is is given by attractor of E with respect to k, denoted $Attr_k(E)$ and defined as below.

$$
\begin{aligned}
Attr_k^0(E) &= E \\
Attr_k^{j+1}(E) &= \{(q,s) \mid q \in Q_k, \exists \tau \in Attr_k^j(E)[(q,s) \hookrightarrow \tau]\} \\
&\quad \cup \{(q,s) \mid q \in Q_{1-k}, \forall \tau[((q,s) \hookrightarrow \tau) \rightarrow \tau \in Attr_k^j(E)]\} \\
Attr_k(E) &= \bigwedge_{j \geq 0} Attr_k^j(E)
\end{aligned}
$$

Note that the reachability problem of the previous section can be considered as a special case where all states belong to one player.

5 Regularity of the Attractor Set

Let reachability game for player k be given by game structure
$H = (Q_0 \oplus Q_1, \Gamma, \Delta)$ and a regular set $\mathcal{L}(\mathcal{A})$ specified by automata $\mathcal{A}$. In this section we show that winning region of player-k in the above game is regular.

We can modify the automata $\mathcal{R}$ of section 3 to $\mathcal{R}^k$ to compute attractor set with respect to player k. We need to replace the states such as (p, t) in section 3

by (p,T), $T \subseteq Q \cup S_{\mathcal{A}}$, as player k may not be able to guarantee to bring the game in specific configuration but only in a set of configurations.

The state set of $\mathcal{R}^k$, $St(R^k)$, is defined as follows.

Let $\mathcal{P}(Z)$ denote the powerset of Z, let $U_{\mathcal{P}(Q,\mathcal{A})} = \mathcal{P}(Q \cup S_{\mathcal{A}})$ and let $U_{\mathcal{P}(Q,\mathcal{A}),\downarrow} = U_{\mathcal{P}(Q,\mathcal{A})} \cup \{\downarrow\}$.

Let $R_1^{(k)} = (Q \cup S_{\mathcal{A}}) \times U_{\mathcal{P}(Q,\mathcal{A}),\downarrow}$

Let $R_2^{(k)} = \{s_w \mid s \in R_1^{(k)}, (w \in \Gamma) \text{ or } (Push_1^{q,w} \in \Delta, \text{ for some } q \in Q)\}$

$St(R^{(k)}) = Q'' \cup R_1^{(k)} \cup R_2^{(k)} \cup (Q \times \{2\})$

Intuitive meaning of state (p,T) is that $\mathcal{R}^k$ accepts a configuration C from (p,T) iff either player k can guarantee that current order-1 stack can be popped in a state $\in T$ (in fact, $T \cap Q$) or it guarantees that the game from configuration C can reach a configuration C' without popping topmost order-1 stack of C and in a run of $\mathcal{A}$ on input C', $\mathcal{A}$ reaches a state $\in T$ (in fact, $T \cap S_{\mathcal{A}}$) at the end of reading topmost order-1 stack of C. This is proved in Theorem 2(ii).

Transition function $\delta_{\mathcal{R}^k}$ is defined using the saturation procedure as in section 3.

5.1 Transitions in δ_0

(0). For $q \in Q$, $(q'', [_2[_1, (q,\downarrow)) \in \delta_0$.

(i). $q \in Q_k$, $(\mathbf{q}, \mathbf{a}, \mathbf{pop_1^p}) \in \mathbf{\Delta}$ then for all $T \in U_{\mathcal{P}(Q,\mathcal{A}),\downarrow}$, $((q,T), a, (\mathbf{p}, \mathbf{T}))$.

(ii). $q \in Q_k$, $(\mathbf{q}, \mathbf{a}, \mathbf{push_2^p}) \in \mathbf{\Delta}$ then for all $T \in U_{\mathcal{P}(Q,\mathcal{A}),\downarrow}$,

$$((q,T), a, \bigvee_{\mathbf{T_1} \subseteq \mathbf{Q} \cup \mathbf{S}_{\mathcal{A}}} [(\mathbf{p}, \mathbf{T_1})_{\mathbf{a}} \bigwedge \wedge_{\mathbf{t} \in \mathbf{T_1}} (\mathbf{t}, \mathbf{T})_{\mathbf{a}}])$$

(iii). $q \in Q_k$, $(\mathbf{q}, \mathbf{a}, \mathbf{pop_2^p}) \in \mathbf{\Delta}$ then we have the following triples in δ_0

(a) $((q,\downarrow), a, (\mathbf{p}, \mathbf{2}))$,
(b) $((r,2), b, (\mathbf{r}, \mathbf{2}))$, $r \in Q, b \in \Gamma$
(c) $((r,2),]_1[_1, (\mathbf{r}, \downarrow))$,
(d) $((r,2),]_1]_2, (\delta_{\mathcal{A}}(\mathbf{r'}, [_\mathbf{2}]_\mathbf{2}), \downarrow))$,
(e) $((q,T), a, \mathbf{true})$, if $p \in T$

(iv). $q \in Q_k$, and $(\mathbf{q}, \mathbf{a}, \mathbf{push_1^{p,u}}) \in \mathbf{\Delta}$ then for all $T \in U_{\mathcal{P}(Q,\mathcal{A}),\downarrow}$, $((q,T), a, (\mathbf{p}, \mathbf{T})_\mathbf{u})$

(v). Transitions related to $\mathcal{A}$

(a) $(q'', [_2, (\delta_{\mathcal{A}}(\mathbf{q'}, [_\mathbf{2}), \downarrow))$, for $q \in Q$.
(b) $((q,T), \epsilon, (\delta_{\mathcal{A}}(\mathbf{q'}, [_\mathbf{2}[_\mathbf{1}), \mathbf{T}))$, for all $T \in U_{\mathcal{P}(Q,\mathcal{A}),\downarrow}$.
(c) $((t_1,T), a, (\delta_{\mathcal{A}}(\mathbf{t_1}, \mathbf{a}), \mathbf{T}))$, $\quad for\ (t_1 \in S_{\mathcal{A}}, T = \downarrow, a \in \Gamma \cup \{[_1,]_1\})$
$\quad OR \quad for\ \ (t_1 \in S_{\mathcal{A}}, T \in U_{\mathcal{P}(Q,\mathcal{A})}, a \in \Gamma)$.
(d) For $T \in U_{\mathcal{P}(Q,\mathcal{A})}$,

$$\begin{array}{ll} ((t_1,T),]_1, true) \in \delta_0 & \text{if } \delta_{\mathcal{A}}(t_1,]_1) \in T \\ ((t_1,T),]_1, false) \in \delta_0 & \text{otherwise} \end{array}$$

(vi). Transitions from states in R_2
$((t_1, T)_a, \epsilon, (\delta_{\mathcal{A}}(\mathbf{t_1}, [_1\mathbf{a}), \mathbf{T}))$, for all $t_1 \in S_{\mathcal{A}}, T \in U_{\mathcal{P}(Q,\mathcal{A}),\downarrow}$.

(vii). $\mathbf{q \in Q_{1-k}}$
We define for $T \in U_{\mathcal{P}(Q,\mathcal{A}),\downarrow}$,
$I_1((q,T),a) = \bigwedge\{(p,T) \mid (q,a,pop_1^p) \in \Delta\}$
$I_2((q,T),a) = \bigwedge\{(q,T)_u \mid (q,a,push_1^{p,u}) \in \Delta\}$
$I_3((q,T),a) = \bigwedge\{\bigvee_{T_1 \in U_{\mathcal{P}(Q,\mathcal{A})}}[(q,T_1)_a \wedge_{t \in T_1} (t,T)_a] \mid (q,a,push_2^p) \in \Delta\}$
$I_4((q,\downarrow),a) = \bigwedge\{(p,2) \mid (q,a,pop_2^p) \in \Delta\}$
For $T \in U_{\mathcal{P}(Q,\mathcal{A})}$,

$$I_4((q,T),a) = \begin{cases} false & \text{if } (q,a,pop_2^p) \in \Delta, \text{ for some } p \notin T \\ true & \text{otherwise} \end{cases}$$

We add the following triples to δ_0, for $T \in U_{\mathcal{P}(Q,\mathcal{A}),\downarrow}$,

$$((q,T), a, \bigwedge_{j=1}^{4} I_j((q,T),a))$$

[Intuitively, I_1, I_2, I_3 and I_4 cover the cases when player $1-k$ chooses moves $pop_1, push_1, push_2$ and pop_2 respectively. In case of pop_2, if player $1-k$ can pop the current stack to reach a state $\notin T$, then constraint of reaching a state in T at the end of current order-1 stack can not be met and $\mathcal{R}^k$ rejects from this state.]

5.2 Saturation Step

Saturation process is as follows.
$\delta_{k+1} := \delta_k \cup \{((p,T)_x, \epsilon, \delta_{\mathbf{k}}((\mathbf{p},\mathbf{T}),\mathbf{x})) \mid p \in Q, (p,T)_x \in R_2\}$

5.3 Correctness of the Construction

Following is an easy Lemma, analogous to Lemma 1 of section 3.

Lemma 2. *1. For $t \in S_{\mathcal{A}}$, $w \in \mathcal{L}(\mathcal{R}^k, (t,\downarrow))$ iff $w \in \mathcal{L}(\mathcal{A}, t)$.*
2. For $t_1 \in S_{\mathcal{A}}$, $T \in U_{\mathcal{P}(Q,\mathcal{A})}$, $w \in \mathcal{L}(\mathcal{R}^k, (t_1,T))$ iff $w = x]_1 v$, $x \in \Gamma^*$ and $\delta_{\mathcal{A}}(t_1, x]_1) \in T$.
3. $w \in \mathcal{L}(\mathcal{R}^k, (q,2))$ iff $w = x]_1[_1 u$ for $x \in \Gamma^*$ and $u \in \mathcal{L}(\mathcal{R}^k, q)$ or $w = x]_1]_2$ for $x \in \Gamma^*$ and $[_2]_2 \in \mathcal{L}(\mathcal{A}, q')$.
4. For $q \in Q$, $w \in \mathcal{L}(\mathcal{R}^k, q'')$ iff $w = [_2[_1 v$ for some v, and $v \in \mathcal{L}(\mathcal{R}^k, (q,\downarrow))$ or $w \in \mathcal{L}(\mathcal{A}, q')$.
5. For $r \in Q \times (U_{\mathcal{P}(Q,\mathcal{A}),\downarrow})$ and $r_u \in R_2^k$, $w \in \mathcal{L}(\mathcal{R}^k, r_u)$ iff $uw \in \mathcal{L}(\mathcal{R}^k, r)$.
6. For $t \in S_{\mathcal{A}}$, $a \in \Gamma$, $w \in \mathcal{L}(\mathcal{R}^k, (t,\downarrow)_a)$ iff $[_1 aw \in \mathcal{L}(\mathcal{A}, t)$.
7. For $(t_1,T) \in S_{\mathcal{A}} \times U_{\mathcal{P}(Q,\mathcal{A})}$, $a \in \Gamma$, $w \in \mathcal{L}(\mathcal{R}^k, (t_1,T)_a)$ iff $w = x]_1 v$, $x \in \Gamma^*$ and $\delta_{\mathcal{A}}(t_1, [_1 ax]_1) \in T$.

Proof. Proof is similar to that of Lemma 1, we omit it. □

To state the main theorem in correctness proof, we first introduce a notation. Let $T \subseteq Q \cup S_{\mathcal{A}}$, we define $M(T) = \{(r, [_2 s]_2) \mid s \in S_1^*, \delta_{\mathcal{A}}(r', [_2 s) \in T\}$. $M(T)$ is the set of configurations of $\mathcal{H}$, on which automata $\mathcal{A}$ has a run which reaches a state in T after seeing the last order-1 stack. Using this notation, we have the following main theorem in the correctness proof which is analogous to Theorem 1 of section 3.

Theorem 2. *The following assertions hold for all $p, q \in Q$, and $T \subseteq Q \cup S_{\mathcal{A}}$.*

1. *$w \in \mathcal{L}(\mathcal{R}^k, (q, \downarrow))$ iff $(q, [_2 [_1 w) \in Attr_k(\mathcal{L}(\mathcal{A}))$.*
2. *$w \in \mathcal{L}(\mathcal{R}^k, (p, T))$ iff $w = x]_1 u$, for some $x \in \Gamma^*$, and $(p, [_2 [_1 x]_1]_2) \in Attr_k(\{(q, [_2]_2) \mid q \in T\} \cup M(T))$.*

Proof. In the full version. □

Corollary 3. *$w \in \mathcal{L}(\mathcal{R}^k, q'')$ iff $(q, w) \in Attr_k(\mathcal{L}(\mathcal{A}))$.*

Proof. Immediate, using Theorem 2(1.) and Lemma 2(4.). □

6 Strategy Extraction

From the results of section 3 and section 5, it follows that one can decide for an arbitrary configuration whether it is in $pre^*(C)$ or it is in the winning region of player-k by simply checking membership in the corresponding automata. It is a natural question to ask if a configuration u is in $pre^*(C)$, can we also determine the sequence of transitions of $\mathcal{H}$ which starting from u reach a configuration in C or if u is in winning region of player-k then can we determine what is the winning strategy of player-k from u to reach set E.

These questions have relevance to synthesis of transition systems and have been investigated in [2] in the context of PDS. It turns out that accepting runs of the corresponding automata on a configuration encode such strategies. In [2], two kinds of strategies are synthesized for a player to win a reachability game (in its winning region) in a PDS. One of these is executable by a pushdown automata and another is a minimum cost strategy executable in linear time. In this section we generalize these results of [2] to the setting of order-2 *hpds* .

6.1 Strategy Executable by Order-2 Hpda

An order-2 *hpda* executing a strategy is a deterministic order-2 pushdown automata with input and output tapes. It reads the moves of player $1-k$ from the input tape and and outputs the moves of player k on the output tape. A more formal definition can be given as in section 3.3 in [2].

In this section, we show how to design a order-2 *hpda* to execute player-k's winning strategy in the second order *hpds* reachability game.

Theorem 3. *Let a game structure $\mathcal{H}$ for second order hpds and a regular set E be given. One can effectively design a second order hpda B such that for every configuration C in the winning region of player-k, after an appropriate initialization B's stack, B executes the winning strategy of player-k from position C. The initialization of B can be done in linear time in the length of C.*

Proof. We sketch the construction of B. The initialization of B's stack is by an accepting dag of automata computing $Attr_k(E)$ on input C. Let this dag be D. We assume that each node of D has more information than just a state of automata, for example it has information about transition taken at that node. We view D as layered dag, a sequence of set of nodes or transitions. This sequence is initially stored in the stack of B linearly, with root of D as top of B's stack. The control states of B include a subset of $\mathcal{R}^k$'s state and some auxiliary states. Initially B is in state q''. Transition function of B depends on its control state (say z) and topmost symbol (say ϕ) of B's stack, and also on the input tape if the control state of B corresponds to player $1-k$'s state in $\mathcal{H}$. We describe below the action of B in various cases which are grouped according to transitions in *hpds* .

Intuitively, D codes the winning strategy of player k from C. B walks through various paths in D (depending on player $1-k$'s choices). B's state represents state of the node B is currently at, while ϕ represents all nodes at that level of D.

(0) If $z = q''$ and $(q'', [_2[_1, q) \in \phi$, then B pops the topmost element of the stack and enters state q.

(i) If $z = (q, T)$, $q \in Q_k$ and $((q, T), a, (\mathbf{p}, \mathbf{T})) \in \phi$ then B outputs pop_1^p, pops topmost element of its stack and moves to state (p, T).

(ii) $z = (q, T)$, $q \in Q_k$ and $((q, T), a, (\mathbf{p}, \mathbf{T})_\mathbf{u}) \in \phi$ then B outputs $push_1^{p,u}$, pops topmost element of its stack and moves to state $(p, T)_u$.

(iii) $z = (q, T)$, $q \in Q_k$ and $((\mathbf{q}, \mathbf{T}), \mathbf{a}, (\mathbf{p}, \mathbf{T_1})_\mathbf{a} \wedge \bigwedge_{\mathbf{t} \in \mathbf{T_1}} (\mathbf{t}, \mathbf{T})_\mathbf{a}) \in \phi$ for some $T_1 \subseteq Q \cup S_\mathcal{A}$ then B outputs $push_2^p$ and pops the top element from the stack. (The current top of the stack has transitions with source $(p, T_1)_a$ and $(t, T)_a$ for each $t \in T_1$). B remembers the transition α begining with $(p, T_1)_a$ and modifies the current top element of stack to contain transitions with source $(t, T)_a$ only, for $t \in T_1$. (That is B pops the top element and pushes a new one which contains only transitions with source $(t, T)_a$ of the old top element).
Now B does a $push_2$ operation. Deletes the topmost element of the stack and pushes a dag expansion of transition α on the stack and changes its control state to (p, T_1).

(iv) $(\mathbf{q}, \mathbf{a}, \mathbf{pop_2^p}) \in \mathbf{\Delta}$ If $z = q$, $q \in Q_k$ and $(q, \downarrow), a, (\mathbf{p}, \mathbf{2}))\phi$ then B outputs pop_2^p, pops the top element from the stack and changes its control state to $(p, 2)$.
If $z = (r, 2)$ and $((r, 2), b, (\mathbf{r}, \mathbf{2})) \in \phi$ then B does pop_1 and remains in state $(r, 2)$.
If $z = (r, 2)$ and $((r, 2),]_1[_1, (\mathbf{r}, \downarrow)) \in \phi$ then B does pop_1 and enters state $(r, \downarrow)$.

If $z = (r, 2)$ and $((r, 2),]_1]_2, (\delta_{\mathcal{A}}(\mathbf{r}', [_\mathbf{2}]_\mathbf{2}), \downarrow)) \in \phi$ then B enters the Halt state.
If $z = (r, 2)$ and $((q, T), a, \mathbf{true}) \in \phi$ (corresponding to $(\mathbf{q}, \mathbf{a}, \mathbf{pop_2^p}) \in \mathbf{\Delta}$) then B does a pop_2^p operation. In the topmost element of the stack now it examines a transition with source $(p, T)_b$ for $b \in \Gamma$ and enters state $(p, T)_b$. [such a unique $(p, T)_b$ is guaranteed to be there, by action of B in step (ii)].

(v),(vi) Transitions related to $\mathcal{A}$:
If $z = q''$ or $z = (q, t)$ and the first or second transitions of group (v) $\in \phi$ then B enters a Halt state. /* A configuration in the target set is reached */.

(vii) If $z = q$ and $q \in Q_{1-k}$ then B reads the input to find transition say, β. B now finds transitions begining with the target set of β in the top stack symbol and proceeds to take same actions as in one of the earlier cases corresponding to β.

(viii) Transition is added in the saturation step:
Let $z = (p, T)_x$ and $((p, T)_x, \epsilon, S) \in \phi$. S is a conjunction of states. Further let l be the least number such that this transition is in δ_{l+1}.
B pops the topmost element from the stack and pushes a dag rooted at (p, T) for input x whose leaves are states S and has transitions in δ_l only. B changes its control state to (p, T).

This finishes the description of the *hpda* B. □

Note that unlike in [2], the strategy executed by this *hpda* is not constant time. This is because for a single pop_2 operation in *hpds* there are as many transitions (of group (iii).b) in the accepting dag as the number of elements in the order-1 stack. It seems possible to make the strategy constant time by doing a initialization of B's stack in several order-1 stacks each containing the portion of D corresponding to an order-1 stack in C. We have not worked out the details yet.

6.2 Computing Min-Cost Strategy

Let C be a configuration in the winning region of player k in a *hpds* reachability game for player-k. The number of steps (also called cost) in which player k is guaranteed to reach the target set E assuming the worst case behaviour from the opponent is easily seen to be the least j such that $C \in Attr_k^j(E)$. A winning strategy of player k in which he plays at any configuration a move which guarantees him to win the game in *minimum cost from that configuration* is called minimum cost winning strategy of player k.

Analogous to a result of [2], we show that min-cost value for player k from a configuration C can be characterized using accepting dags of $\mathcal{R}^k$ on input C. Further, there is an accepting dag of $\mathcal{R}^k$ on input C which codes a min-cost strategy of player k.

We define a cost labeled dag as a dag D alongwith a labeling by natural number of each node of D. The labeling value of node n, denoted $L(n)$, is defined using the rules below.

(0) Nodes with states in set $S_{\mathcal{A}} \times U_{\mathcal{A},\downarrow}$ or leaves marked 'true' have cost $L(n) = 0$.

(i) Transition rule used at n is $((q,T), a, (\mathbf{p}, \mathbf{T}))$, $q \in Q_k$
[Corresponding to $(q, a, pop_1^p) \in \Delta$]
Let n_1 be the child of n corresponding to $(\mathbf{p}, \mathbf{T})$ then $L(n) = 1 + L(n_1)$.

(ii) Transition rule used at n is $((q,T), a, (\mathbf{p}, \mathbf{T_1})_{\mathbf{a}} \bigwedge \wedge_{\mathbf{t} \in \mathbf{T_1}} (\mathbf{t}, \mathbf{T})_{\mathbf{a}})$, for some $T_1 \subseteq Q \cup S_{\mathcal{A}}$.
[Corresponding to $(q, a, push_2^p) \in \Delta$]
Let $n_0, n_1, \cdots n_r$ be children of n corresponding to $(p, T_1)_a, \{(t,T)_a \mid t \in T_1\}$ respectively. We have $L(n) = 1 + L(n_0) + max\{L(n_i) \mid 1 \leq i \leq r\}$

(iii) Transition rule used at n corresponds to $(\mathbf{q}, \mathbf{a}, \mathbf{pop_2^p}) \in \mathbf{\Delta}$ for $q \in Q_k$

(a) If the rule used is $((q,\downarrow), a, (\mathbf{p}, \mathbf{2}))$ let n_1 be the child of n corresponding to $(\mathbf{p}, \mathbf{2})$. then we have $L(n) = 1 + L(n_1)$.

(b) If the rule used is $((q,T), a, \mathbf{true})$ then we define $L(n) = 1$.

(c) If the rule used is $((r,2), b, (\mathbf{r}, \mathbf{2}))$ let n_1 be the child of n corresponding to $(\mathbf{r}, \mathbf{2})$. then we define $L(n) = L(n_1)$.

(d) If the rule used is $((r,2),]_1[_1, (\mathbf{r}, \downarrow))$, let n_1 be the child of n corresponding to $(\mathbf{r}, \downarrow)$ then we define $L(n) = L(n_1)$.
If the rule used is $((r,2),]_1]_2, (\delta_{\mathcal{A}}(\mathbf{r'}, [_{\mathbf{2}}]_{\mathbf{2}}), \downarrow)$ then we define $L(n) = 0$

(iv) Transition rule used at n is $(\mathbf{q}, \mathbf{a}, \mathbf{push_1^{p,u}}) \in \mathbf{\Delta}$, $q \in Q_k$
[Corresponding to $(\mathbf{q}, \mathbf{a}, \mathbf{push_1^{p,u}}) \in \mathbf{\Delta}$]
Let n_1 be the child of n corresponding to $(\mathbf{p}, \mathbf{T})_{\mathbf{u}}$. We define $L(n) = 1 + L(n_1)$.

(v),(vi) Transitions related to $\mathcal{A}$: If rule used at n is from groups (v) or (vi) then we define $L(n) = 0$.

(vii) Let n correspond to a state $q \in Q_{1-k}$. (the rule used is from group (vii))
In this case n has children corresponding to each rule of Δ applicable at n. Let the cost of choosing these moves be $\{c_1, \cdots, c_r\}$. then $L(n) = max\{c_1, \cdots, c_r\}$.

(viii) Transition used at n is $((p,T)_x, \epsilon, S)$ (introduced in saturation step)
[Which is added during saturation procedure and is in δ_{l+1}.]
Let D_1 be a cost labeled dag on input x for this transition, and using transitions of δ_l only. We define $L(n) =$ value at the root of D_1.
(Note the use of recursion here as labeling inside D_l must satisfy the cost labeling rules. Also note that the choice of dag D_1, is not unique.)

The following theorem is analogous to Theorem 2 of section 5 and is proved in the similar way.

Theorem 4. *Let D be a cost labeled dag for input w with root of D marked with state m and cost with j.*

1. *$(q, [_2[_1 w) \in Attr_k^j(\mathcal{L}(\mathcal{A}))$ iff there is an accepting dag D of $\mathcal{R}^k$ for input w, rooted at $(q, \downarrow)$ and with cost labeled j at its root.*
2. *Let $p \in Q, T \subseteq Q \cup S_{\mathcal{A}}$ and $x \in \Gamma^*$.*
$(p, [_2[_1 x]_1]_2) \in Attr_k^j(\{(q, [_2]_2) \mid q \in T\} \cup M(T))$ iff there is an accepting dag D of $\mathcal{R}^k$ for input $x]_1$, rooted at (p,T) and with cost labeled j at its root.

Proof. It follows by considering cost labeling in the proof of Theorem 2. □

We construct a minimum cost dag of $\mathcal{R}^k$ on C in bottom up manner, considering at each step all possible moves of $\mathcal{R}^k$ and choosing the one leading to minimum cost and satisfying the rules of cost labeled dag. For transitions of the kind $((p,T)_x, \epsilon, S)$, we use the algorithm recursively constructing the dag with transitions of lower levels only. For a fixed $\mathcal{H}, \mathcal{A}$ this can be done in time linear in the length of the configuration. This leads to the following theorem.

Theorem 5. *For a fixed $\mathcal{H}$, $\mathcal{A}$, k, there is a linear time algorithm which takes as its input a configuration of player k and outputs the minimum cost move of player k in the reachability game above if the input configuration is in the winning region of player k.*

7 Conclusion

We have given an alternative construction of finite automata recognizing $pre^*_{\mathcal{H}}(C)$, where C is a regular set of configurations of a second order pushdown system $\mathcal{H}$. Our construction is explicit and simple to understand. It also yields results about strategy synthesis which are a generalization from pushdown reachability games to second order pushdown reachability games. Our approach can be extended to pushdown systems of higher than second order also though some new phenomenon arise there which make the construction more involved. This will be future work.

References

1. Hague, M., Ong, L.: Symbolic backwards-reachability analysis for higher-order pushdown systems. In proc. FoSSaCS, 2007 (Also downloadable from www.comlab.ox.ac.uk/oucl/work/matthew.hague)
2. Cachat, T.: Symbolic strategy synthesis for games on pushdown graphs. In proc. ICALP (2003) 315–333
3. Bouajjani, A., Meyer, A.: Symbolic reachability analysis of higher-order context-free processes. In proc. FSTTCS (2004) LNCS 3328.
4. Bouajjani, A., Esparza, J., Maler O.: Reachability analysis of pushdown automata: Applications to model checking. In proc. Concur (1997) 135-150
5. Walukiewicz, I.: Pushdown processes: games and model checking. Information and computation **164** (2001) 234–263
6. Carayol, A., Wohrle, S.: The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In proc. FSTTCS (2003) 112-123
7. Maslov, A. N.: Multilevel stack automata. In Problems of Information Transmission, 15:1170-1174, 1976.
8. Engelfriet, J.: Iterated stack automata and complexity classes. Information and computation **95** (1991) 21–75
9. Damm, W.: The IO and OI hierarchies. Theoretical Computer Science **20** (1982) 95–208
10. Loding, C., Thomas, W.: Alternating Automata and Logics over Infinite Words, IFIP TCS'00, LNCS 1872, pp. 521-535, 2000.

11. Caucal, D.: On infinite terms having a decidable monadic theory. In proc. MFCS (2002) 165-176.
12. Gradel, E., Thomas, W., Wilke, Th.: Automata, logics, and infinite games, LNCS 2500, Springer, 2002.
13. Kupferman, O., Vardi, M. Y.: An Automata-Theoretic Approach to Reasoning about Infinite-state Systems, In Proc. CAV 2000, LNCS 1885, 2000.