International Journal of Foundations of Computer Science
Vol. 19, No. 4 (2008) 983–998
© World Scientific Publishing Company

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

Received 24 December 2007
Accepted 3 June 2008
Communicated by V. Halava and I. Potapov

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 automaton for recognizing $pre^*_{\mathcal{H}}(\mathcal{C})$ is explicit. The termination of saturation procedure used is obvious. It gives EXPTIME bound on size of the automaton recognizing $pre^*_{\mathcal{H}}(\mathcal{C})$ if there is no alternation present in $\mathcal{H}$ and in the automaton recognizing $\mathcal{C}$.

Keywords: Reachability; higher-order pushdown automata; symbolic model-checking; verification.

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.*

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. Here we survey a few results which are related to our work.

In [1], 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. [1] gave an iterative procedure, called saturation procedure, to construct an automaton recognizing $pre^*(C)$, starting from the automaton recognizing C. The iteration step adds new edges to the automaton but no new states.

In [2], 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 *hpds* configurations is given by an automaton with finitely many states but whose edge labels are ordinary finite automata recognising order-1 stack configurations. Authors of [2] have generalized the saturation procedure to these automata to prove their results. Recently, Hague and Ong [3] have extended results of [2] to higher order pushdown systems. This is a technically non-trivial result. Let A be an order-2 finite automaton recognizing a set of order-2 *hpds* configurations C. The saturation procedure of [3] 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. In [3], details of all this are given.

In this paper, we give an alternative construction of an automaton recognizing $pre^*(C)$ for second order *hpds*. The state set of our automaton 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.

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

The authors in [3], in fact solve a more general problem of constructing an automaton to recognise winning regions in 2-player reachability game over an *hpds*. This informally can be stated as computing $pre^*(C)$ for an alternating *hpds*. Further, in [3], $\mathcal{A}$, recognising C is also assumed to be alternating. Size of the automaton constructed in [3] for this general case is double exponential in the size of *hpds* $\mathcal{H}$ (of order-2) and the size of automaton $\mathcal{A}$, recognising C. We consider the special case when $\mathcal{H}$ is nondeterministic and $\mathcal{A}$ is given as a nondeterministic automaton, in this case we show that an automaton recognising $pre^*(C)$ can be constructed in time single exponential in the size of *hpds* $\mathcal{H}$ and size of automaton $\mathcal{A}$. In fact, in this case we construct an alternating automaton with a polynomial number of states. In [3], complexity of their construction for this special case is not discussed but recently Hague [4], while comparing construction in [3] with our construction in [5], has pointed out that for this special case complexity of the two constructions match.

The present paper is based on the workshop paper [5] but due to lack of space does not cover all results of [5]. In [5], our construction is also extended to the two player case with the complexity bounds matching those in [3]. Further, [5] also contains generalizations of some results of [6] on uniform winning strategies to the case of two player reachability game on order-2 *hpds*. Full version of these results will be published elsewhere.

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.

- $push_1^w([_1\ a_1 \cdots a_l]_1) = [_1\ w\ a_2 \cdots a_l]_1,$
- $push_1^w([_2\ s_1\ s_2 \cdots s_k]_2) = [_2\ push_1^w(s_1)\ s_2 \cdots s_k]_2$
- $push_2([_2\ s_1\ s_2 \cdots s_k]_2) = [_2\ s_1\ s_1\ s_2 \cdots s_k]_2$
- $pop_1([_2\ s_1\ s_2 \cdots s_k]_2) = [_2\ pop_1(s_1)\ s_2 \cdots s_k]_2,$
- $pop_m([_m\ s_1\ s_2 \cdots s_k]_m) = \begin{cases} [_m\ s_2 \cdots s_k]_m & \text{if } k \geq 1 \\ \text{undefined} & \text{otherwise} \end{cases} \qquad m \in \{1,2\}$

We also define topmost element, $top(s)$, of a stack configuration s, as below.
$top([_1\ a_1 \cdots a_l]_1) = a_1,\quad top([_2\ s_1 \cdots s_k]_2) = top(s_1).$

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. Define $Q' = \{q' \mid q \in Q\}$. A finite (multi)automaton 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 [1] for order-1 pushdown systems.

For any automaton $\mathcal{B}$ 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-automaton $\mathcal{A}$ 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 automaton 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 $\delta : P \times \Sigma \rightarrow B^+(P)$, 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 automaton on an input is a tree. More efficiently we can represent it as a dag as in [6, 7]. 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 automaton $\mathcal{B}$ on input w is a rooted dag in which each node is labeled with a state of $\mathcal{B}$ or by a constant *true* or *false*. Nodes labeled with *true* or *false* do not have any outgoing edges. Each edge in a run-dag is labeled with an input symbol or an 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 $\{q_{i_1}, \dots, q_{i_k}\}$ then there must be a one step transition $(q, u, \{q_{i_1}, \dots, q_{i_k}\}) \in \delta$. 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 $\mathcal{B}$. Implicitly, we assume that any input is accepted from the node labeled *true*.

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 nondeterministic finite multi-automaton 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 multi-automaton $\mathcal{R}_{\mathcal{H},\mathcal{A}}$ to accept $pre^*_{\mathcal{H}}(\mathcal{C}_{\mathcal{A}})$.

- $\mathcal{R}_{\mathcal{H},\mathcal{A}} = (S_{\mathcal{R}}, \Gamma_2, Q'', \delta_{\mathcal{R}}, \mathcal{F} \times \{\downarrow\})$
- State set of $R_{\mathcal{H},\mathcal{A}}$, denoted $S_{\mathcal{R}}$, is a disjoint union of several sets as below.
- Let $Q'' = \{q'' \mid q \in Q\}$, $U_{Q,\mathcal{A},\downarrow} = Q \cup S_{\mathcal{A}} \cup \{\downarrow\}$, $U_{\mathcal{A},\downarrow} = S_{\mathcal{A}} \cup \{\downarrow\}$
  [To remember notation U can be thought of as union]
- $R_1 = (Q \times U_{Q,\mathcal{A},\downarrow}) \cup (S_{\mathcal{A}} \times U_{\mathcal{A},\downarrow})$

- $R_2 = \{s_w \mid s \in R_1, (w \in \Gamma) \text{ or } (Push_1^{q,w} \text{ occurs in } \Delta, \text{ for some } q \in Q)\}$
- $S_\mathcal{R} = Q'' \cup R_1 \cup R_2 \cup (Q \times \{2\})$

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*

Initially, automaton $\mathcal{R}$ has x on its input tape and is in state $q'' \in Q''$. $\mathcal{R}$ has to accept x iff there is a sequence of zero or move hpds moves which when applied to configuration (q, x) results in an element of $\mathcal{C}_\mathcal{A}$. $\mathcal{R}$ non-deterministically guesses if $(q, x) \in \mathcal{C}_\mathcal{A}$ and verifies it by simulating $\mathcal{A}$ on (q, x). In other branches $\mathcal{R}$ guesses an hpds move m applicable from (q, x). Depending on m, $\mathcal{R}$ makes some transitions ensuring that after these transitions $\mathcal{R}$ accepts iff the hpds configuration C', resulting from (q, x) after move m, is in $pre^*_\mathcal{H}(\mathcal{C}_\mathcal{A})$. We now explain transitions of $\mathcal{R}$ for each move m of $\mathcal{H}$. The $push_1$ and pop_1 moves are handled as in order-1 case. Specifically, for pop_1, $\mathcal{R}$ just skips the top of the stack symbol (and changes its state to reflect the state of the new configuration). For $push_1^{p,w}$, $\mathcal{R}$ moves to a state $(p, \downarrow)_w$ which accepts u iff $\mathcal{R}$ in state $(p, \downarrow)$ accepts wu. This amounts to $\mathcal{R}$ accepting (q, x) if $\mathcal{R}$ accepts the configuration reached after $push_1^{p,w}$ move. ($\downarrow$ is present here because of the representation of states chosen by us, as explained later, it may be ignored for the present). Transitions from states of the form $(p, \downarrow)_w$ are added in saturation step (section 4.2).

Pop_2 move is simply handled by ignoring the input till the end of current order-1 stack. The $push_2$ move 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. Let the resulting configuration after $push_2$ move be C'. There are two possibilities. First that a configuration in $C_\mathcal{A}$ is reached after popping the topmost stack of C' and second that a configuration in $C_\mathcal{A}$ is reached before popping the topmost stack of C'. For the first case, we break the transition sequence of $\mathcal{H}$ leading to a configuration in $C_\mathcal{A}$ in two parts, (i) upto popping the topmost stack in C' (ii) after popping the topmost stack of C'. As we have only one copy in the input for both the topmost stack and the stack just below it in C', $\mathcal{R}$ needs to simulate both (i), (ii) simultaneously. For this $\mathcal{R}$ needs to know in which state is the top stack popped? This is done by guessing this state. Also note that the sequence in (i) depends only on the topmost stack of C'. So at a $push_2$ move the computation of $\mathcal{R}$ may split into two (conjunctive) 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 in state q. The verifying thread will die on meeting the end of current order-1 stack either successfully or unsuccessfully.

For the second case, let C'' be a configuration in $C_\mathcal{A}$ reached before popping the topmost stack of C'. Stack of C'' in this case extends x, it can be roughly written as $s_1''.x$, where s_1'' is the sequence of order-1 stacks in C'' above x. Note that s_1'' depends on the topmost stack of C' only. We break a run of $\mathcal{A}$ on the stack of C'' in two parts, (i) upto the end of s_1'' (ii) on x. Once again $\mathcal{R}$ needs to simulate both (i), (ii) simultaneously. For this $\mathcal{R}$ needs to know in which state a run of $\mathcal{A}$ is after reading s_1''. This is done by guessing a state in $S_{\mathcal{A}}$. So at a $push_2$ move computation of $\mathcal{R}$ may split into two threads, one thread verifies that there is a configuration of $\mathcal{H}$ (consisting of s_1'') reached from the topmost stack of C' and there is a path of $\mathcal{A}$ on it ending in state t and the other thread starts running $\mathcal{A}$ on the current input (x) in state t. The verifying thread will die on meeting the end of current order-1 stack either successfully or unsuccessfully. The other thread accepts if $\mathcal{A}$ accepts the input.

Overall, at a $push_2$ move $\mathcal{R}$ guesses a state $\in Q \cup S_{\mathcal{A}}$ and splits its computation in two threads. The first thread verifies a constraint and the second thread continues computaion in a normal way. 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 $\mathcal{R}$, using its runs, is able to keep information about *hpds* configurations that can be reached. At each step, the automaton 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.

In this paragraph, we briefly describe the meaning of each state in $\mathcal{R}$. For the purpose of this paragraph we let p, q range over states in Q and s, s_1, s_2 range over states in $S_{\mathcal{A}}$. $\mathcal{R}$ in state $(q, -)$ on input w corresponds to $\mathcal{R}$ looking at configuration $(q, [_2[_1w)$ of $\mathcal{H}$. There are three types of constraint checking states (q, p), (q, s), (s_1, s_2). A configuration C is accepted from (q, p) if $\mathcal{H}$ can pop the topmost stack of C in state p (formalized Theorem 1, part 2). A configuration C is accepted from (q, t) if without popping topmost stack of C a configuration C' of $\mathcal{H}$ is reached s.t. $\mathcal{A}$ when started on C' has a run which reaches state s at the end of reading topmost stack of C (formalized in Theorem 1, part 3). If $\mathcal{R}$ is in state $(s, -)$ then it is simulating $\mathcal{A}$. States of the form $(q, 2)$ are just to take care of the pop_2 case. States of the form $(-, -)_a$ or $(-, -)_w$ are convenient abbreviations for actions required in push operations. The saturation step adds transitions from these states only. Roughly, $\mathcal{R}$ accepts u from state $(p, -)_w$ iff it accepts wu from state $(p, -)$. These intuitions are formalized fully in Lemma 1 and Theorem 1.

4. Transition Relation of $\mathcal{R}$

In this section, we present the transition relation δ of $\mathcal{R}$.

4.1. *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. We also include a brief explanation after each transition, embedded between *'s.

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

(i). $(\mathbf{q}, \mathbf{a}, \mathbf{pop_1^p}) \in \boldsymbol{\Delta}$ then $((q,t), a, (\mathbf{p}, \mathbf{t})) \in \delta_{\mathbf{0}}$, for $t \in U_{Q,\mathcal{A},\downarrow}$.

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

(a)

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

(b)

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

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

(iii). $(\mathbf{q}, \mathbf{a}, \mathbf{pop_2^p}) \in \boldsymbol{\Delta}$ 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) $((r, 2),]_1]_2, (\mathbf{z}, \downarrow))$, if $[_2]_2 \in \mathcal{L}(\mathcal{A}, r')$ and $z \in \mathcal{F}$.
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). $(\mathbf{q}, \mathbf{a}, \mathbf{push_1^{p,u}}) \in \boldsymbol{\Delta}$ then $((q,t), a, (\mathbf{p}, \mathbf{t})_{\mathbf{u}}) \in \delta_{\mathbf{0}}$, for $t \in U_{Q,\mathcal{A},\downarrow}$.

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

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

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

(b) $((q,t), \epsilon, (\mathbf{z}, \mathbf{t}))$, for all $q \in Q$, $t \in U_{\mathcal{A},\downarrow}$ and $z \in \delta_{\mathcal{A}}(q', [_2[_1)$.
The automaton $\mathcal{R}$ guesses a configuration

(c) For $t_1 \in S_{\mathcal{A}}$,

$$((t_1, t_2), a, (\mathbf{z}, \mathbf{t_2})) \text{ if } \begin{cases} (t_2 = \downarrow, a \in \Gamma \cup \{[_1,]_1\} \text{ and } z \in \delta_{\mathcal{A}}(t_1, a)) \\ \textbf{or} \\ (t_2 \in S_{\mathcal{A}}, a \in \Gamma \text{ and } z \in \delta_{\mathcal{A}}(t_1, a)) \end{cases}$$

Simulates $\mathcal{A}$ in the first component

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

$$\begin{array}{ll} ((t_1,t_2),]_1,\mathbf{true}) \in \delta_0 & \text{if } t_2 \in \delta_{\mathcal{A}}(t_1,]_1) \\ ((t_1,t_2),]_1,\mathbf{false}) \in \delta_0 & \text{otherwise} \end{array}$$

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

(vi). Transitions from states in R_2

$((t_1,t_2)_a,\epsilon,(\mathbf{z},\mathbf{t_2})) \in \delta_0$,

for $t_1 \in S_{\mathcal{A}}$, $t_2 \in U_{\mathcal{A},\downarrow}$, $a \in \Gamma$ and $z \in \delta_{\mathcal{A}}(t_1,[_1a)$.

4.2. *Saturation step*

Let $s \in Q \times U_{Q,\mathcal{A},\downarrow}$. During the construction we created states of the form s_w, with the intuitive property that $\mathcal{R}$ accepts u from s_w iff R accepts wu from s. We have not yet added any transitions from states s_w. We ensure above property by adding transitions of the form (s_w,ϵ,S) to δ_0 to have $(s_w,\epsilon,S) \in \delta_{\mathcal{R}}$ iff $(s,w,S) \in \delta_{\mathcal{R}}$. Also, we need to add only a minimum number of such transition to δ_0 so that each transition in $\delta_{\mathcal{R}}$ is needed and $\mathcal{R}$ does not accept anything more than $pre^*(C_{\mathcal{A}})$. Therefore, we begin by adding transitions (s_w,ϵ,S) to $\delta_{\mathcal{R}}$ if $(s,w,S) \in \delta_0$. This however may create more triples of the form $(s,w,S) \in \delta_{\mathcal{R}}$ than in δ_0. This addition process therefore has to be iterated till we can't add any more transitions. This is called saturation process [1], the resulting transition function δ_k after k^{th} iteration is defined as follows. The new transitions are defined from states in $Q \times U_{Q,\mathcal{A},\downarrow}$ only.

$\delta_{k+1} := \delta_k \cup \{((p,t)_x,\epsilon,\mathbf{S}) \mid p \in Q, (p,t)_x \in R_2, ((p,t),x,\mathbf{S}) \in \delta_k\}$

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.

As mentioned in section 3, $\delta_{\mathcal{R}} = \bigcup_{i\geq 0} \delta_i$.

5. Easy Observations about Construction

The Lemma below follows easily from the construction.

Lemma 1 *For all $w \in \Gamma_2^*$, $t, t_1, t_2 \in S_{\mathcal{A}}$ and $a \in \Gamma$ the following hold.*

1. *$w \in \mathcal{L}(\mathcal{R},(t,\downarrow))$ iff $w \in \mathcal{L}(\mathcal{A},t)$.*
2. *$w \in \mathcal{L}(\mathcal{R},(t_1,t_2))$ iff $w = x]_1v$, $x \in \Gamma^*$ and $t_2 \in \delta_{\mathcal{A}}(t_1,x]_1)$.*
3. *$w \in \mathcal{L}(\mathcal{R},(q,2))$ iff $w = x]_1[_1u$ 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 $w = [_2[_1v$ 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. $w \in \mathcal{L}(\mathcal{R}, (t, \downarrow)_a)$ *iff* $[_1 aw \in \mathcal{L}(\mathcal{A}, t)$.

7. $w \in \mathcal{L}(\mathcal{R}, (t_1, t_2)_a)$ *iff* $w = x]_1 v$, *for some* $x \in \Gamma^*$ *and* $t_2 \in \delta_{\mathcal{A}}(t_1, [_1 ax]_1)$.

Proof.

1. This follows from transitions in (v.c) and noting that only these transitions are applicable in states $(t, \downarrow)$.

2. ($\Leftarrow$) This direction follows easily by transitions in (v.c) and (v.d). v is arbitrary as every input is accepted from '*true*' state.

   ($\Rightarrow$) Note that only transitions in $\mathcal{R}$ from (t_1, t_2) are (v.c) and (v.d). Transitions of (v.c) can't lead to an accepting state (because an accepting transition in $\mathcal{A}$ can occur only on reading $]_2$). So the accepting dag must have (v.c) transitions zero or more times followed by transition (iv.d) first case. It follows that $w = x]_1 v$ for some $x \in \Gamma^*$ and $t_2 \in \delta_{\mathcal{A}}(t_1, x]_1)$.

3. ($\Leftarrow$) Apply transition (iii.b) till a '$]_1$' is seen, at this point a transition (iii.c) or (iii.d) is applied.

   ($\Rightarrow$) From state $(q, 2)$ only transitions (iii.b-d) are possible. Therefore after some applications of (iii.b) either (iii.c) or (iii.d) must occur. In the former case, we have $w = ax]_1[_1 u$ for some $x \in \Gamma^*$ and $u \in \mathcal{L}(\mathcal{R}, (q, \downarrow))$. In the latter case, $w = ax]_1]_2$, for some $x \in \Gamma^*$ and $[_2]_2 \in \mathcal{L}(\mathcal{A}, q')$.

4. Follows from transitions in (0) and (v.a) and noting that these are the only transitions from state q''.

5. ($\Rightarrow$) The only transitions from state r_u are the ones added in saturation step. So if $w \in \mathcal{L}(\mathcal{R}, r_u)$ then there is some S, such that $(r, u, S) \in \delta$ and for each $q \in S$, $w \in \mathcal{L}(\mathcal{R}, q)$. This shows that $uw \in \mathcal{L}(\mathcal{R}, r)$.

   ($\Leftarrow$) Let D be an accepting dag witnessing $uw \in \mathcal{L}(\mathcal{R}, r)$. We let $D \uparrow u$ be the dag obtained by truncating each path in D to have label u only and let S be the set of states labeling leaves of $D \uparrow u$. It follows that $(r, u, S) \in \delta$, hence $(r_u, \epsilon, S) \in \delta$. Also as for each $q \in S$, there is an accepting dag on input w, we have that $w \in \mathcal{L}(\mathcal{R}, r_u)$.

6. ($\Rightarrow$) Only transition from $(t, \downarrow)_a$ is (vi). So there is a $z \in \delta_{\mathcal{A}}(t, [_1 a)$ s.t. $w \in \mathcal{L}(\mathcal{R}, (z \downarrow))$. By part 1., $w \in \mathcal{L}(\mathcal{A}, z)$. It follows that, $[_1 aw \in \mathcal{L}(\mathcal{A}, t)$.

   ($\Leftarrow$) As $[_1 aw \in \mathcal{L}(\mathcal{A}, t)$, there is a $z \in \delta_{\mathcal{A}}(t, [_1 a)$ and $w \in \mathcal{L}(\mathcal{A}, z)$. By part 1., $w \in \mathcal{L}(\mathcal{R}, (z \downarrow))$. By (vi), there is also a transition $((t, \downarrow)_a, \epsilon, (z, \downarrow)) \in \delta$. It follows that $w \in \mathcal{L}(\mathcal{R}, (t, \downarrow)_a)$.

7. Only transition from state $(t_1, t_2)_a$ is in (vi), from this it is clear that $w \in \mathcal{L}(\mathcal{R}, (t_1, t_2)_a)$ iff there is a $t_3 \in \delta_{\mathcal{A}}(t_1, [_1 a)$ and $w \in \mathcal{L}(\mathcal{R}, (t_3, t_2))$. Using (2.), this is equivalent to: there is a $t_3 \in \delta_{\mathcal{A}}(t_1, [_1 a)$, $w = x]_1 v$, for some $x \in \Gamma^*$ and $t_2 \in \delta_{\mathcal{A}}(t_3, x]_1)$. This is equivalent to: $w = x]_1 v$, for some $x \in \Gamma^*$ and $t_2 \in \delta_{\mathcal{A}}(t_1, [_1 ax]_1)$. The claim follows.

□

6. Completeness of $\mathcal{R}$

In this section we see that $\mathcal{R}$ accepts all desired inputs.

Lemma 2 *Let $\hookrightarrow_n$ be the (at most) n-step reachability relation on configurations of hpda $\mathcal{H}$. The following hold for all $n \geq 0$, $p, q \in Q$, $t \in S_{\mathcal{A}}$, $x, y \in \Gamma^*$ and $w, u, v \in \Gamma_2^*$.*

(1) If $(q, [_2[_1 w) \hookrightarrow_n \tau$ and $\tau \in \mathcal{L}(\mathcal{A})$ then $w \in \mathcal{L}(\mathcal{R}, (q, \downarrow))$.

(2) If $(p, [_2[_1 x]_1]_2) \hookrightarrow_n (q, [_2]_2)$ then $x]_1 u \in \mathcal{L}(\mathcal{R}, (p, q))$.

(3) If $(p, [_2[_1 y]_1]_2) \hookrightarrow_n (r, [_2[_1 v]_1]_2)$ and $t \in \delta_{\mathcal{A}}(r', [_2[_1 v]_1)$ then $y]_1 u \in \mathcal{L}(\mathcal{R}, (p, t))$.

Proof. We prove assertions (1),(2) and (3) simultaneously by induction on n.

Base case: $n = 0$

For Assertion (1), note that $(q, [_2[_1 w) = \tau$. Let z be s.t. $z \in \delta_{\mathcal{A}}(q', [_2[_1)$ and $w \in \mathcal{L}(\mathcal{A}, z)$. From (v.b), $\mathcal{R}$ has a transition $((q, \downarrow), \epsilon, (z, \downarrow))$ and by Lemma 1(1), $w \in \mathcal{L}(\mathcal{R}, (z, \downarrow))$. It follows that $w \in \mathcal{L}(\mathcal{R}, (q, \downarrow))$.

Assertion (2) follows because $(p, [_2[_1 x]_1]_2) \hookrightarrow (q, [_2]_2)$ is not possible in zero steps.

For (3) note that in this case $p = r$ and $y = v$. Let z be s.t. $z \in \delta_{\mathcal{A}}(p', [_2[_1)$ and $t \in \delta_{\mathcal{A}}(z, y]_1)$. From (v.b), $\mathcal{R}$ has a transition $((p, t), \epsilon, (z, t))$ and by Lemma 1(2), $y]_1 u \in \mathcal{L}(\mathcal{R}, (z, t))$. It follows that $y]_1 u \in \mathcal{L}(\mathcal{R}, (p, t))$.

Induction step: $n = m + 1$.

Let $C_0 \hookrightarrow C_1 \hookrightarrow_m C$ be the form of reduction in the 'if' part of assertions (1),(2) and (3). We consider cases depending on the transition $C_0 \hookrightarrow C_1$. In each case, we show (1),(2) and (3) to hold.

- **push$_1$**.
  (1) Let $C_0 = (q, [_2[_1 a w_1)$ and $C_1 = (p, [_2[_1 y_1 w_1)$ where $(q, a, push_1^{p,y_1}) \in \Delta$. By induction hypothesis (1), $y_1 w_1 \in \mathcal{L}(\mathcal{R}, (p, \downarrow))$. By Lemma 1(5), $w_1 \in \mathcal{L}(\mathcal{R}, (p, \downarrow)_{y_1})$. By the transition in the group (iv), it follows that $a w_1 \in \mathcal{L}(\mathcal{R}, (q, \downarrow))$.
  (2) Let $C_0 = (p, [_2[_1 a x_1]_1]_2)$ and $C_1 = (p_1, [_2[_1 y_1 x_1]_1]_2)$, where $(p, a, push_1^{p_1,y_1}) \in \Delta$. By induction hypothesis (2), $y_1 x_1]_1 u \in \mathcal{L}(\mathcal{R}, (p_1, q))$. By Lemma 1(5), $x_1]_1 u \in \mathcal{L}(\mathcal{R}, (p_1, q)_{y_1})$. By the transition in the group (iv), it follows that $a x_1]_1 u \in \mathcal{L}(\mathcal{R}, (p, q))$.
  (3) Let $C_0 = (p, [_2[_1 a z_1]_1]_2)$ and $C_1 = (p_1, [_2[_1 y_1 z_1]_1]_2)$, where $(p, a, push_1^{p_1,y_1}) \in \Delta$. As $C_1 \hookrightarrow_m (r, [_2[_1 v]_1]_2)$ and $t \in \delta_{\mathcal{A}}(r', [_2[_1 v]_1)$, by induction hypothesis (3), $y_1 z_1]_1 u \in \mathcal{L}(\mathcal{R}, (p_1, t))$. By Lemma 1(5), $z_1]_1 u \in \mathcal{L}(\mathcal{R}, (p_1, t)_{y_1})$. By the transition in group (iv), it follows that $a z_1]_1 u \in \mathcal{L}(\mathcal{R}, (p, t))$.
- **pop$_1$**.
  (1) Let $C_0 = (q, [_2[_1 a w_1)$ and $C_1 = (p, [_2[_1 w_1)$, where $(q, a, pop_1^p) \in \Delta$. By transition in group (i), we have $((q, \downarrow), a, (p, \downarrow)) \in \delta_0$. As $C_1 \hookrightarrow_m C$, by induction hypothesis $w_1 \in \mathcal{L}(\mathcal{R}, (p, \downarrow))$.
  Therefore $w = a w_1 \in \mathcal{L}(\mathcal{R}, (q, \downarrow))$, proving (1).

(2) Let $C_0 = (p, [_2[_1ax_1]_1]_2)$ and $C_1 = (p', [_2[_1x_1]_1]_2)$, where $(p, a, pop_1^{p'}) \in \Delta$. By group (i) transition we have that $((p,q), a, (p',q)) \in \delta$. As $C_1 \hookrightarrow_m (q, [_2]_2)$, by induction hypothesis $x_1]_1u \in \mathcal{L}(\mathcal{R}, (p',q))$. Therefore $ax_1]_1u \in \mathcal{L}(\mathcal{R}, (p,q))$, proving (2).

(3) Let $C_0 = (p, [_2[_1ay_1]_1]_2)$ and $C_1 = (p', [_2[_1y_1]_1]_2)$, where $(p, a, pop_1^{p'}) \in \Delta$. As in (2), we have $((p,t), a, (p',t)) \in \delta$. As $C_1 \hookrightarrow_m (r, [_2[_1v]_1]_2)$ and $t \in \delta_{\mathcal{A}}(r', [_2[_1v]_1)$, by induction hypothesis $y_1]_1u \in \mathcal{L}(\mathcal{R}, (p',t))$. Therefore $ay_1]_1u \in \mathcal{L}(\mathcal{R}, (p,t))$, proving (3).

- **push$_2$**.

(1) Let $C_0 = (q, [_2[_1ax]_1v), x \in \Gamma^*, C_1 = (q_1, [_2[_1ax]_1[_1ax]_1v)$, where $(q, a, Push_2^{q_1}) \in \Delta$.

We consider two subcases. First, $C_1 \hookrightarrow_m C$ without popping topmost order-1 stack of C_1. In this case, $C = (p, [_2[_1u_1]_1[_1ax]_1v)$ and $(q_1, [_2[_1ax]_1]_2) \hookrightarrow_m (p, [_2[_1u_1]_1]_2)$, for some u_1. As $C \in \mathcal{L}(\mathcal{A})$, let r_1 be s.t. $r_1 \in \delta_{\mathcal{A}}(p', [_2[_1u_1]_1)$ and $[_1ax]_1v \in \mathcal{L}(\mathcal{A}, r_1)$. By induction hypothesis (3), $ax]_1v \in \mathcal{L}(\mathcal{R}, (q_1, r_1))$. Further, by Lemma 1(5.), $x]_1v \in \mathcal{L}(\mathcal{R}, (q_1, r_1)_a)$.

As $[_1ax]_1v \in \mathcal{L}(\mathcal{A}, r_1)$, by Lemma 1(6.) $x]_1v \in \mathcal{L}(\mathcal{R}, (r_1, \downarrow)_a)$.

By transition (ii.b), choosing $t_1 = r_1$ in the disjunction, we have that $ax]_1v \in \mathcal{L}(\mathcal{R}, (q, \downarrow))$.

For the other subcase, let the topmost order-1 stack of C_1 be popped in state q_2. In this case, $(q_1, [_2[_1ax]_1]_2) \hookrightarrow_m (q_2, [_2]_2)$ and $(q_2, [_2[_1ax]_1v) \hookrightarrow_m C$. By induction hypothesis (2) and (1), $ax]_1v \in \mathcal{L}(\mathcal{R}, (q_1, q_2))$ and $ax]_1v \in \mathcal{L}(\mathcal{R}, (q_2, \downarrow))$. By Lemma 1(5.), $x]_1v \in \mathcal{L}(\mathcal{R}, (q_1, q_2)_a)$ and $x]_1v \in \mathcal{L}(\mathcal{R}, (q_2, \downarrow)_a)$. By transition (ii.b), choosing $q_l = q_2$ in the disjunction, we have that $ax]_1v \in \mathcal{L}(\mathcal{R}, (q, \downarrow))$.

(2) Let $C_0 = (p, [_2[_1ax_1]_1]_2)$ and $C_1 = (p_1, [_2[_1ax]_1[_1ax_1]_1]_2)$, where $(p, a, Push_2^{p_1}) \in \Delta$. As $C = (q, [_2]_2)$, topmost order-1 stack of C_1 must be popped to reach C. Let it be popped in state p_2. In this case, $(p_1, [_2[_1ax]_1]_2) \hookrightarrow_m (p_2, [_2]_2)$ and $(p_2, [_2[_1ax]_1]_2) \hookrightarrow_m C$. By induction hypothesis (2), $ax]_1u \in \mathcal{L}(\mathcal{R}, (p_1, p_2))$ and $ax]_1u \in \mathcal{L}(\mathcal{R}, (p_2, q))$. By Lemma 1(5.), $x]_1u \in \mathcal{L}(\mathcal{R}, (p_1, p_2)_a)$ and $x]_1u \in \mathcal{L}(\mathcal{R}, (p_2, q)_a)$. By transition (ii.a), choosing $q_l = p_2$ in the disjunction, we have that $ax]_1u \in \mathcal{L}(\mathcal{R}, (p, q))$.

(3) Let $C_0 = (p, [_2[_1ay_1]_1]_2)$ and $C_1 = (p_1, [_2[_1ay_1]_1[_1ay_1]_1]_2)$, where $(p, a, Push_2^{p_1}) \in \Delta$.

We consider two subcases. First, $C_1 \hookrightarrow_m C$ without popping topmost order 1 stack of C_1. In this case $C = (r, [_2[_1u_1]_1[_1ay_1]_1]_2)$ and $(p_1, [_2[_1ay_1]_1]_2) \hookrightarrow_m (r, [_2[_1u_1]_1]_2)$, for some u_1. As $t \in \delta_{\mathcal{A}}(r', [_2[_1u_1]_1[_1ay_1]_1)$, let r_1 be s.t. $r_1 \in \delta_{\mathcal{A}}(r', [_2[_1u_1]_1)$ and $t \in \delta_{\mathcal{A}}(r_1, [_1ay_1]_1)$. By induction hypothesis (3.), $ay_1]_1u \in \mathcal{L}(\mathcal{R}, (p_1, r_1))$. By Lemma 1(5.), $y_1]_1u \in \mathcal{L}(\mathcal{R}, (p_1, r_1)_a)$.

As $t \in \delta_{\mathcal{A}}(r_1, [_1ay_1]_1)$, by Lemma 1(7.), we have $y_1]_1u \in \mathcal{L}(\mathcal{R}, (r_1, t)_a)$. By transition (ii.b), choosing $t = r_1$ in the disjunction, we have that $ay_1]_1u \in \mathcal{L}(\mathcal{R}, (p, t))$.

For the other subcase, let p_2 be the state in which topmost stack of C_1 is popped. In this case,
$(p_1, [_2[_1ay_1]_1]_2) \hookrightarrow_m (p_2, [_2]_2)$ and $(p_2, [_2[_1ay_1]_1]_2) \hookrightarrow_m C = (r, [_2[_1v]_1]_2)$. Further, we are given that $t \in \delta_{\mathcal{A}}(r', [_2[_1v]_1)$. By induction hypothesis (2) and (3) respectively, $ay_1]_1u \in \mathcal{L}(\mathcal{R}, (p_1, p_2))$ and $ay_1]_1u \in \mathcal{L}(\mathcal{R}, (p_2, t))$. By Lemma 1(5.), $y_1]_1u \in \mathcal{L}(\mathcal{R}, (p_1, p_2)_a)$ and $y_1]_1u \in \mathcal{L}(\mathcal{R}, (p_2, t)_a)$. By transition (ii.b), choosing $q_l = p_2$ in the disjunction, we have that $ay_1]_1u \in \mathcal{L}(\mathcal{R}, (p, t))$.

- **pop$_2$**.

(1) Let $C_0 = (q, [_2[_1aw_1]_1[_1w_2]_1v)$ and $C_1 = (p, [_2[_1w_2]_1v)$ or $C_0 = (q, [_2[_1aw_1]_1]_2)$ and $C_1 = (p, [_2]_2)$ where $(q, a, pop_2^p)) \in \Delta$.

By transition (iii.a), $((q, \downarrow), a, (p, 2)) \in \delta$.

By induction hypothesis, $w_2]_1v \in \mathcal{L}(\mathcal{R}, (p, \downarrow))$ or $C_1 = (p, [_2]_2) = C$ (because no other configuration of $\mathcal{H}$ is reachable from C_1). In the latter case $[_2]_2 \in \mathcal{L}(\mathcal{A}, p')$. By Lemma 1(3), $w_1]_1[_1w_2]_1v$ or $w_1]_1]_2 \in \mathcal{L}(\mathcal{R}, (p, 2))$ respectively. It follows that $aw_1]_1[_1w_2]_1v$ or $aw_1]_1]_2 \in \mathcal{L}(\mathcal{R}, (q, \downarrow))$ respectively. This is what was to be shown.

(2) Let $C_0 = (p, [_2[_1ax]_1]_2)$. As a pop_2 transition is applied to C_0, we have $C_1 = (p_1, [_2]_2)$, where $(p, a, pop_2^{p_1}) \in \Delta$.

In this case $C = C_1$ and $q = p_1$. So $(p, a, pop_2^q) \in \Delta$ and $((p, q), a, \textbf{true}) \in \delta$ by transition (iii.e). It follows that $ax]_1u \in \mathcal{L}(\mathcal{R}, (p, q))$, as any string is accepted from 'state' *true*.

(3) Let $C_0 = (p, [_2[_1ay_1]_1]_2)$. Now the result of a pop operation will be $C_1 = (p_1, [_2]_2)$, from which no configuration of the form $(r, [_2[_1v]_1]_2)$ is reachable. Therefore this case does not arise.

□

7. Soundness of $\mathcal{R}$

In this section we see that $\mathcal{R}$ accepts only desired inputs.

Lemma 3 *The following hold for all $w \in \Gamma_2^*$, $x \in \Gamma^*$, $p, q \in Q$ and $t \in S_{\mathcal{A}}$.*

(1) If $w \in \mathcal{L}(\mathcal{R}, (q, \downarrow))$ then $[_2[_1w \in S_2$ and for some τ, $(q, [_2[_1w) \hookrightarrow^* \tau$ and $\tau \in \mathcal{L}(\mathcal{A})$

(2) If $x]_1u \in \mathcal{L}(\mathcal{R}, (p, q))$ then $(p, [_2[_1x]_1]_2) \hookrightarrow^* (q, [_2]_2)$.

(3) If $w \in \mathcal{L}(\mathcal{R}, (p, t))$ then $w = y]_1u$ for some $y \in \Gamma^*$ and $(p, [_2[_1y]_1]_2) \hookrightarrow^* (r, [_2[_1v]_1]_2)$ for some v, r and $t \in \delta_{\mathcal{A}}(r', [_2[_1v]_1)$.

Proof. We prove (1), (2) and (3) simultaneously by induction on the size of an accepting dag of $\mathcal{R}$ on the given input. We show that (1), (2) and (3) hold for a dag D if these hold for all dags of strictly lesser size. Size of a dag of $\mathcal{R}$ is defined below. Size induces a well founded measure on all dags.

Assume that the saturation procedure for defining the transition function of $\mathcal{R}$ terminates in k_0 iterations. We say that a transition of $\mathcal{R}$ is of level k if it is in $\delta_k - \delta_{k-1}$, for $k > 0$. If a transition is in δ_0 then we say that it is of of level 0. Given a dag D, its size is defined to be a tuple $(n_{k_0}, \cdots, n_0)$, where n_i is the number of transitions of level i in D. The ordering on size is lexicographic with highest weight assigned to coordinate for n_{k_0}.

- **(1).** We consider various cases for the first transition of $\mathcal{R}$ from state $(q, \downarrow)$.
  - **(i)** It is the transition of group (i). Let this transition be $(q, a, pop_1^{q_1})$. In this case $w = aw_1$. The child of this node in dag D is $(q_1, \downarrow)$ and by induction hypothesis we get $(q_1, [_2[_1 w_1) \hookrightarrow^* \tau$ and $\tau \in \mathcal{L}(\mathcal{A})$. $(q, [_2[_1 w) \hookrightarrow (q_1, [_2[_1 w_1)$ as $(q, a, pop_1^{q_1}) \in \Delta$ in this case. This proves (1).
  - **(ii)** It is the second transition of group (ii). Let this transition correspond to $(q, a, push_2^{q_j}) \in \Delta$. In this case $w = aw_1$. Further, children of this node in dag D are $(q_j, t_1)_a$ and $(t_1, \downarrow)_a$, for some $t_1 \in Q \cup S_{\mathcal{A}}$.

    For the first subcase, assume $t_1 \in Q$. As only ϵ transitions are possible from states $(q_j, t_1)_a$ and $(t_1, \downarrow)_a$, the children of these nodes have states corresponding to some satisfying assignment of boolean expression $\delta_k((q_j, t_1), a)$ and to some satisfying assignment of expression $\delta_{k'}((t_1, \downarrow), a)$ respectively for some k, k'. Expanding the ϵ transitions by these dags, we obtain dags of lesser size (because we are replacing one transition of higher level by several transitions of lower levels in lexicographic ordering) rooted at (q_j, t_1) and $(t_1, \downarrow)$ for input w.

    By induction hypothesis (part 1), $[_2[_1 w \in S_2$ and $(t_1, [_2[_1 w) \hookrightarrow^* \tau$ for some $\tau \in \mathcal{L}(\mathcal{A})$.

    Therefore, $w = ax]_1 u$ for some $x \in \Gamma^*$. By induction hypothesis (part 2), $(q_j, [_2[_1 ax]_1]_2) \hookrightarrow^* (t_1, [_2]_2)$.

    So we have,

    $(q, [_2[_1 w) = (q, [_2[_1 ax]_1 u) \hookrightarrow (q_j, [_2[_1 ax]_1[_1 ax]_1 u) \hookrightarrow^* (t_1, [_2[_1 ax]_1 u) \hookrightarrow^* \tau$.

    In the second subcase, $t_1 \in S_{\mathcal{A}}$. Only ϵ transition is possible from states $(q_j, t_1)_a$ and $(t_1, \downarrow)_a$. As before from dag rooted at $(q_j, t_1)_a$ on input w_1, we get a smaller size dag rooted at (q_j, t_1) on input w. By induction hypothesis (part 3), $w = ay]_1 u$ for some $y \in \Gamma^*$,

    $(q_j, [_2[_1 ay]_1]_2) \hookrightarrow^* (r, [_2[_1 v]_1]_2)$ for some r, v and $t_1 \in \delta_{\mathcal{A}}(r', [_2[_1 v]_1)$. We also have an accepting dag rooted at $(t_1, \downarrow)_a$ for input $y]_1 u$. By Lemma 1(6.), $[_1 ay]_1 u \in \mathcal{L}(\mathcal{A}, t_1)$. It follows that $[_2[_1 v]_1[_1 ay]_1 u \in \mathcal{L}(\mathcal{A}, r')$.

    So we have,

    $(q, [_2[_1 w) = (q, [_2[_1 ay]_1 u) \hookrightarrow (q_j, [_2[_1 ay]_1[_1 ay]_1 u) \hookrightarrow^* (r, [_2[_1 v]_1[_1 ay]_1 u)$ and $(r, [_2[_1 v]_1[_1 ay]_1 u) \in \mathcal{C}_{\mathcal{A}}$.

(iii) It is the transition of group (iii.a). Let this transition be $(q, a, (q_j, 2))$. It implies that $(q, a, pop_2^{q_j})\Delta$.In this case $w = aw_1$.

From state $(q_j, 2)$ only transitions (iii.b-d) are possible. Therefore after some applications of (iii.b) either (iii.c) or (iii.d) must occur. In the former case we have $w = ax]_1[_1u$ for some $x \in \Gamma^*$ and there is an accepting dag rooted at $(q_j, \downarrow)$ for input u. By induction hypothesis, we have that $(q_j, [_2[_1u) \hookrightarrow \tau$, for some $\tau \in \mathcal{C}_\mathcal{A}$.

Putting together we have, $(q, [_2[_1w) = (q, [_2[_1ax]_1[_1u) \hookrightarrow (q_j, [_2[_1u) \hookrightarrow \tau$.

In the latter case (where (iii.d) occurs), (iii.d) is of the form $((q_j, 2),]_1]_2, (z, \downarrow))$, where $z \in \mathcal{F}$. There is no accepting path of $\mathcal{R}$ from state $(z, \downarrow)$, because no proper prefix of S_2 is in S_2. It follows that there is no further input after $]_1]_2$ in the dag. Therefore $w = ax]_1]_2$, by side condition of transition (iii.d), we also have that $(q_j, [_2]_2) \in \mathcal{L}(\mathcal{A})$. Summing up we have, $(q, [_2[_1w) = (q, [_2[_1ax]_1]_2) \hookrightarrow (q_j, [_2]_2) \in \mathcal{L}(\mathcal{A})$.

(iv) It is the transition of group (iv). Let this transition be $(q, a, (q_j)_x)$. So $w = aw_1$. In this case, root of this dag has a child dag labeled with state $(q_j, \downarrow)_x$, from which only ϵ transition is possible. So the children of state $(q_j, \downarrow)_x$, consist of state set S, such that $((q_j, \downarrow), x, S) \in \delta_k$, for some k. By expanding this transition as a run-dag of δ_k, we obtain a dag of lesser size rooted at $(q_j, \downarrow)$ accepting xw_1. By induction hypothesis on this dag and using the fact that, $(q, [_2[_1w) \hookrightarrow (q_j, [_2[_1xw_1)$, the claim follows.

(v) It is the transition of (v.b). In this case we have a $z \in \delta_\mathcal{A}(q', [_2[_1)$, and an accepting dag rooted at $(z, \downarrow)$ on input w. By Lemma 1(1), $w \in \mathcal{L}(\mathcal{A}, z)$, it follows that $[_2[_1w \in \mathcal{L}(\mathcal{A}, q')$. The claim follows as $(q, [_2[_1w) \in \mathcal{C}_\mathcal{A}$.

- **(2).** We consider cases depending on the transition of $\mathcal{R}$ from state (p, q). The cases of pop_1 and $push_1$ are the same as in (1) except that we use induction hypothesis for (2) instead of (1). $Push_2$ is also similar to (1) but the second subcase does not arise at all.

  For pop_2, the first transition can only be of the form (iii.e), let it be $((p, q), a, \mathbf{true})$. In this case $x = ax_1$, and $(p, a, pop_2^q) \in \Delta$. It follows immediately that $(p, [_2[_1ax_1]_1]_2) \hookrightarrow (q, [_2]_2)$.

  No other cases are possible.
- **(3).** We consider cases depending on the transition of $\mathcal{R}$ from state (p, t). Once again the cases of pop_1 and $push_1$ as the first transitions in the accepting dag are the same as in (1) except that we use induction hypothesis (3) instead of (1). $Push_2$ is Similar to (1), both the subcases are treated in the same way as in (1). We discuss the second subcase here.

  Let the transition at the root of the dag be the second transition of group (ii). Let this transition correspond to $(p, a, push_2^{p_j}) \in \Delta$. In this case $w = aw_1$. Further, children of this node in dag D are $(p_j, t_1)_a$ and $(t_1, t)_a$, for some $t_1 \in S_\mathcal{A}$.

  As in (1), we obtain a dag rooted at (p_j, t_1) for input w. By induction hypothesis (part 3.), we have $w = ay]_1u$ for some $y \in \Gamma^*$,

$(p_j, [_2[_1 ay]_1]_2) \hookrightarrow^* (r, [_2[_1 v]_1]_2)$ and $t_1 \in \delta_{\mathcal{A}}(r', [_2[_1 v]_1)$.
As $y]_1 u \in \mathcal{L}(\mathcal{R}, (t_1, t)_a)$, by Lemma 1(7.), we get $t \in \delta_{\mathcal{A}}(t_1, [_1 ay]_1)$. Therefore, $t \in \delta_{\mathcal{A}}(r', [_2[_1 v]_1[_1 ay]_1)$.

Putting together we have, $w = ay]_1 u$,
$(p, [_2[_1 ay]_1]_2) \hookrightarrow (p_j, [_2[_1 ay]_1[_1 ay]_1]_2) \hookrightarrow^* (r, [_2[_1 v]_1[_1 ay]_1]_2)$
and $t \in \delta_{\mathcal{A}}(r', [_2[_1 v]_1[_1 ay]_1)$. This is what was to be shown.

The case pop_2 does not arise (there is no transition of group (*iii*) in state (p, t)).
The only case left is the transition (v.b). In this case we have a $z \in \delta_{\mathcal{A}}(r', [_2[_1)$, and an accepting dag rooted at (z, t) on input w. By Lemma 1(2.), $w = y]_1 u$ for some $y \in \Gamma^*$ and $t \in \delta_{\mathcal{A}}(z, y]_1)$. It follows that $t \in \delta_{\mathcal{A}}(r', [_2[_1 y]_1)$. So we have $(p, [_2[_1 y]_1]_2) \hookrightarrow^* (r, [_2[_1 v]_1]_2)$ (in zero steps with $p = r$ and $y = v$) and $t \in \delta_{\mathcal{A}}(r', [_2[_1 v]_1)$.

□

8. The Main Results

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

1. *$w \in \mathcal{L}(\mathcal{R}, (q, \downarrow))$ iff $[_2[_1 w \in S_2$ and there is a $\tau \in \mathcal{L}(\mathcal{A})$ such that $(q, [_2[_1 w) \hookrightarrow^* \tau$.*
2. *$x]_1 u \in \mathcal{L}(\mathcal{R}, (p, q))$ iff $(p, [_2[_1 x]_1]_2) \hookrightarrow^* (q, [_2]_2)$.*
3. *$w \in \mathcal{L}(\mathcal{R}, (p, t))$ iff $w = y]_1 u$, for some $y \in \Gamma^*$, $(p, [_2[_1 y]_1]_2) \hookrightarrow^* (r, [_2[_1 v]_1]_2)$ for some $v \in \Gamma_2^*$ and $t \in \delta_{\mathcal{A}}(r, [_2[_1 v]_1)$.*

Proof. Immediate from Lemma 2 and Lemma 3. □

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 $w \in pre^*(C_{\mathcal{A}})$.*

From the construction of $\mathcal{R}$ we also have the following.

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 automaton with $O((|Q| + |S_{\mathcal{A}}|)^2 \cdot |\Delta|)$ states to recognize $pre^*(C_{\mathcal{A}})$.*

From the above alternating automaton recognizing $pre^*(C_{\mathcal{A}})$, we can get by standard construction an equivalent nondeterministic automaton with 2^z states, where z is a polynomial in the size of $\mathcal{H}$ and $\mathcal{A}$.

9. Conclusion

In [3], $pre^*(C)$ for *hpds* of arbitrary order was shown to be regular, for any regular set C of configurations. In this paper, we have given another construction of an automaton recognising $pre^*(C)$ for (non-alternating) order-2 pushdown systems. Our construction is explicit and termination of the saturation procedure constructing an automaton is obvious. In [3] saturation procedure constructing the automaton may add new states during construction, in our construction we reason beforehand about the states needed during the construction. This results in a saturation procedure similar to order-1 case. Further, the number of these states is bounded by a polynomial (in fact $O(n^2)$) in the size of hpds and the non-deterministic automaton computing C. Our approach also generalizes to *hpds* above order-2, using certain finitary higher-order functions which are stored in states of finite automaton recognising $pre^*(C)$. Finally, we think that it is easy to generalize our technique to order-2 extensions of other order-1 pushdown systems.

References

1. A. Bouajjani, J. Esparza and O. Maler, "Reachability analysis of pushdown automata: Applications to model checking," *Proc. Concur*, 1997, pp. 135–150.
2. A. Bouajjani and A. Meyer, "Symbolic reachability analysis of higher-order context-free processes," *Proc. FSTTCS*, 2004, LNCS 3328, pp. 135–147.
3. M. Hague and L. Ong, "Symbolic backwards-reachability analysis for higher-order pushdown systems," *Proc. FoSSaCS*, 2007, pp. 213–227.
4. M. Hague, "Global Model Checking of Higher-Order Pushdown Systems," *PhD Thesis, Submitted, Oxford University*, May 2008.
5. A. Seth, "An alternative construction in symbolic reachability analysis of second order pushdown automata," *Proc. Satellite workshops of DLT 2007*, TUCS publication (45), 2007, pp. 80-95.
6. T. Cachat, "Symbolic strategy synthesis for games on pushdown graphs," *Proc. ICALP*, 2003, pp. 315–333.
7. C. Loding and W. Thomas, "Alternating automata and logics over infinite words," *Proc. IFIP TCS'00*, 2000, LNCS 1872, pp. 521–535.