Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 915–922

DOI: 10.15439/2015F325
ACSIS, Vol. 5

A New Algorithm for the Determinisation of Visibly Pushdown Automata

Radomír Polách, Jan Trávníček, Jan Janoušek, Bořivoj Melichar
Department of Theoretical Computer Science
Faculty of Information Technology
Czech Technical University in Prague
ul. Thákurova 9, 160 00 Prague 6, Czech Republic
Email: Radomir.Polach@fit.cvut.cz, Jan.Travnicek@fit.cvut.cz, Jan.Janousek@fit.cvut.cz, Borivoj.Melichar@fit.cvut.cz

***Abstract*—Visibly pushdown automata are pushdown automata whose pushdown operations are determined by the input symbol, where the input alphabet is partitioned into three parts for push, pop and local pushdown operations. It is well known that nondeterministic visibly pushdown automata can be determinised. In this paper a new algorithm for the determinisation of nondeterministic visibly pushdown automata is presented. The algorithm improves the existing methods and can result in significantly smaller deterministic pushdown automata. This is achieved in a way that only necessary and accessible states and pushdown symbols are computed and constructed during the determinisation.**

***Index Terms*—Pushdown automata, visibly pushdown automata, deterministic pushdown automata, determinisation of visibly pushdown automata.**

I. Introduction

PUSHDOWN automata, which accept context–free formal languages, are one of the fundamental models of computation of the Theory of formal languages and automata [8]. Every nondeterministic finite automaton, which accepts a regular language, can be determinised and the theory of the determinisation of finite automata is simple and well–researched: states of an equivalent deterministic finite automaton represent so–called deterministic subsets of states of a given nondeterministic finite automaton [8]. The general determinisability does not hold for the case of all types of nondeterministic pushdown automata. The class of deterministic context–free languages is a proper subclass of context–free languages, ie for some nondeterministic pushdown automata their equivalent deterministic versions do not exist. Generally, it is not known how to decide for a given nondeterministic pushdown automaton whether there exists a deterministic equivalent or not. There is a lack of results in the theory of the determinisation of nondeterministic pushdown automata, although such results would be usable, eg when constructing practical deterministic algorithms from nondeterministic pushdown automata.

Visibly pushdown automata [3] are an important and well motivated subclass of pushdown automata, where pushdown operations are determined by the input symbol: the input alphabet is partitioned into three parts A_c, A_r and A_l for push, pop and local pushdown operations, respectively. This relates to function calls, for example. A function call is represented by push operation, local operations executed in the context of the called function are represented by local transitions, and, finally, the return from the function is represented by pop operation. The push, pop and local operations are sometimes referred as call, return and internal operations. Visibly pushdown automata are widely used, researched and known to be used in many practical applications, such as XML processing for example [1], [2], [5], [6], [7], [11].

It is well known that nondeterministic visibly pushdown automata can be determinised [3], [12]. These determinisations use principles that are also used in the well-known determinisation of finite automata: states of the equivalent deterministic automata are represented by so-called deterministic subsets [8]. Alur and Madhusudan [3] presented the proof of the determinisability of a given nondeterministic visibly pushdown automaton with n states by creating a cartesian product consisting of all possible states and then creating deterministic subsets, which resulted in 2^{n^2+n} states of the deterministic version of the pushdown automaton. This was improved in [12], where the upper bound for the number of states was lowered from 2^{n^2+n} to 2^{n^2} and the upper bound for the number of pushdown store symbols was lowered from $|A_c|2^{n^2+n}$ to $|A_c|2^{n^2}$.

In this paper a new algorithm of the determinisation of nondeterministic visibly pushdown automata is presented. The algorithm improves the existing methods and can result in significantly smaller deterministic pushdown automata in many practical examples. Only necessary and accessible states and pushdown symbols of the deterministic pushdown automaton are computed and constructed during the determinisation, which is done by analysing which states are used in transitions on the same level of the nesting of pushdown operations and which pushdown store symbols can appear at the top of the pushdown store for each state.

The paper is organized as follows. Section 2 defines basic notions. Section 3 contains information on related works. Section 4 presents the new incremental algorithm for visibly pushdown automata determinisation. An example of the use of the presented algorithm is presented in Section 5. Finally, the conclusion of the paper is in Section 6.

This research has been supported the Czech Science Foundation as project No. 13-03253S.

II. Basic notions

Basic notions are defined as in standard texts, such as [8].

A. Alphabet, string

An *alphabet* A is a finite nonempty set of *symbols*.

A *string* s is a sequence of n symbols $a_1a_2a_3\dots a_n$ from a given alphabet, where n is the length of the string. A sequence of zero symbols is called empty string. Empty string is denoted by symbol ε and its length is 0.

B. Language

A^* denotes the set of all strings over an alphabet A including the empty string. Set A^+ is defined as $A^+ = A^* \setminus \{\varepsilon\}$. A *language* L over an alphabet A is a set $L \in A^*$. Similarly, for string $x \in A^*$, symbol x^m, $m \geq 0$, denotes the m-fold concatenation of x with $x^0 = \varepsilon$. Set x^* is defined as $x^* = \{x^m : m \geq 0\}$ and $x^+ = x^* \setminus \{\varepsilon\} = \{x^m : m \geq 1\}$.

C. Pushdown automata

A *nondeterministic pushdown automaton* (nondeterministic PDA) is a seven-tuple $M = (Q, A, G, \delta, q_0, Z_0, F)$, where Q is a finite set of *states*, A is an *input alphabet*, G is a *pushdown store alphabet*, δ is a mapping from $Q \times (A \cup \{\varepsilon\}) \times G^*$ into a set of finite subsets of $Q \times G^*$, $q_0 \in Q$ is an initial state, $Z_0 \in G$ is the initial pushdown store symbol, and $F \subseteq Q$ is the set of final (accepting) states.

Triplet $(q, w, x) \in Q \times A^* \times G^*$ denotes the configuration of a pushdown automaton. Top of the pushdown store x is written on its left hand side. The initial configuration of a pushdown automaton is a triple (q_0, w, Z_0) for the input string $w \in A^*$.

The relation $\vdash_M \subset (Q \times A^* \times G^*) \times (Q \times A^* \times G^*)$ is a *transition* of a pushdown automaton M. It holds that $(q, aw, \alpha z) \vdash_M (p, w, \beta z)$ if $(p, \beta) \in \delta(q, a, \alpha)$, where $z, \alpha, \beta \in G^*$. The k-th power, transitive closure, and transitive and reflexive closure of the relation $\vdash_M$ is denoted $\vdash_M^k$, $\vdash_M^+$, $\vdash_M^*$, respectively.

A pushdown automaton M is a *deterministic* pushdown automaton (deterministic PDA), if it holds:

$$\forall q \in Q, \forall a \in A, \forall z \in G, \delta(q, \varepsilon, z) = \emptyset : |\delta(q, a, z)| \leq 1,$$
$$\forall a \in A, |\delta(q, \varepsilon, z)| \geq 1 : \delta(q, a, z) = \emptyset.$$

D. Visibly pushdown automata

Visibly pushdown automata are defined as in [3], [12].

Let $A = A_c \cup A_r \cup A_l$ be a partition of A. The intuition behind the partition is: A_c is the finite set of call (push) symbols, A_r is the finite set of return (pop) symbols, and A_l is the finite set of local symbols.

A *visibly pushdown automaton (VPA)* M over A is a seven-tuple $(Q, A, G, \delta, Q_0, \bot, F)$, where Q is a finite set of *states*, $A = A_c \cup A_r \cup A_l$, G is a finite *pushdown store alphabet*, a special symbol $\bot \in G$ represents the bottom-of-pushdown-store, which can be popped from the pushdown store unlimited number of times, $\delta = \delta_c \cup \delta_r \cup \delta_i$ is the transition mapping, where $(Q, G \setminus \{\bot\}) \in \delta_c(Q, A_c, \varepsilon)$, $(Q, \varepsilon) \in \delta_r(Q, A_r, G)$, and $(Q, \varepsilon) \in \delta_i(Q, A_l, \varepsilon)$, $Q_0 \subseteq Q$ is a set of initial states, $\bot \in G$ is initial pushdown store symbol, and $F \subseteq Q$ is a set of final (accepting) states.

III. Related works

Visibly pushdown automata were introduced in [3]. Moreover, it was shown that any nondeterministic visibly pushdown automaton can be transformed into an equivalent deterministic one. The determinisation principle is similar to the the determinisation principle of finite automata [8].

In [3], states of the resulting deterministic visibly pushdown automaton consist of two components (S, R). Component $R \in \mathcal{P}(Q)$ is an element of powerset of the states of the original automaton. Component $S = \mathcal{P}(Q \times Q)$ is a powerset of pairs of states of the original nondeterministic pushdown automaton that keeps tracking beginning states on path from push transitions to all states listed in R component. We note that, given the union of states in second parts of pairs in S component is equal to R component, the R component can be omitted but for keeping the automata hierarchy simple we maintain this R component in the following definition as a connection to finite automata [12], where the states of the determinized automata correspond to the R component.

Let $M = (Q, A, G, \delta, Q_0, \bot, F)$ be a nondeterministic VPA. For $A = A_c \cup A_r \cup A_l$, an equivalent deterministic VPA $M' = (Q', A, G', \delta', q_0', \bot, F')$ can be constructed as follows: $Q' = 2^{Q \times Q} \times 2^Q$, $q_0' = (Id_{Q_0}, Q_0)$ where $Id_X = \{(x, x) | x \in Q\}$, $F' = \{(S, R) | R \cap F \neq \emptyset\}$, $G' = 2^{Q \times Q} \times 2^Q \times A_c \cup \{\bot\}$, the transition relation $\delta' = \delta_l' \cup \delta_c' \cup \delta_r'$ is given by:

Local: For every
$l \in A_l, ((S', R'), \varepsilon) \in \delta_l'((S, R), l, \varepsilon)$ where

$$S' = \{(q, q') | \exists q'' \in Q : (q, q'') \in S, (q', \varepsilon) \in \delta_l(q'', l, \varepsilon)\},$$
$$R' = \{q' | \exists q \in R : (q', \varepsilon) \in \delta_l(q, l, \varepsilon)\}.$$

Push: For every
$c \in A_c, ((Id_{R'}, R'), (S, R, c)) \in \delta_c'((S, R), c, \varepsilon)$ where

$$R' = \{q' | \exists q \in R : (q', \gamma) \in \delta_c(q, c, \varepsilon)\}.$$

Pop: For every $r \in A_r$,

- if the pushdown store is empty : $((S', R'), \varepsilon) \in \delta_r'((S, R), r, \bot)$ where $S' = \{(q, q') | \exists q'' \in Q : (q, q'') \in S, (q', \varepsilon) \in \delta_r(q'', r, \bot)\}$ and $R' = \{q' | \exists q \in R : (q', \varepsilon) \in \delta_r(q, r, \bot)\}$.
- otherwise: $((S'', R''), \varepsilon) \in \delta_r((S, R), r, (S', R', c))$, where

$$\begin{cases} R'' = \{q' | \exists q \in R' : (q, q') \in U\}, \\ S'' = \{(q, q') | \exists q_3 \in Q : (q, q_3) \in S', (q_3, q') \in U\}, \\ U = \{(q, q') | \exists q_1 \in Q, q_2 \in R : (q_1, q_2) \in S, \\ \quad (q_1, \gamma) \in \delta_c(q, c, \varepsilon), \\ \quad (q', \varepsilon) \in \delta_r(q_2, r, \gamma)\}. \end{cases}$$

The equivalent deterministic automaton has at most 2^{n^2+n} states and at most $|A_c|2^{n^2+n}$ pushdown store symbols. The size of the transition relation can be at most $|A_l|(2^{n^2+n})^2 + |A_c|(2^{n^2+n})^3 + |A_r||A_c|(2^{n^2+n})^3$.

In 2009 an improved upper bound of the number of states has been found by Nguyen Van Tang [12]. In that paper

two optimizations for Alur-Madhusudan's determinisation of visibly pushdown automata were introduced. First, the set of summaries S component of a state pair for some special cases concerning initial states was minimized. Second, R component of the state pair was removed. By removing R component of determinised visibly pushdown automaton the upper bound for the number of states was lowered from 2^{n^2+n} to 2^{n^2} and there were at most $|A_c|2^{n^2}$ pushdown store symbols. The optimization is based on the observation that information stored in R component of a state pair is already contained in S component of the state pair [12]. However, that determinisation algorithm is still not practical. As pointed by Nguyen Van Tang in its implementation of visibly pushdown automata determinisation library, named VPAlib, the determinisation was performed in an exhaustive way. Therefore, their determinisation easily gets stuck with visibly pushdown automata of small size [12].

IV. Determinisation algorithm

This section presents our new algorithm for the determinisation of nondeterministic visibly pushdown automata. The algorithm improves the original determinisation algorithm [3], [12]. As stated in the Introduction our algorithm computes and constructs only necessary and accessible states and pushdown symbols of the deterministic pushdown automaton. The basic idea of this improvement is analysing and tracking pushdown symbols that can appear on the top of the pushdown store for a particular state. With this information the explored pop transitions for the state can be reduced to only those that correspond to the possible pushdown store top symbols. Also, it is shown below that this information on the possible pushdown store top symbols for a state has certain interesting properties, which can be exploited for an effective way of calculation and storing this information for all states in the automaton.

We will use $\mathcal{T}_q$ to denote pushdown store top symbols. The *pushdown store top symbols* of state q, $\mathcal{T}_q \subseteq G'$ are the set of all pushdown store symbols that could appear at the top of the pushdown store for a state $q \in Q'$.

We will also use symbol λ for a local connection: State q' is locally connected to q'' if there is a sequence of transitions from state q'' to state q', the pushdown store depth in both states is the same and the pushdown store depth for all other states along the sequence of transitions is greater than the pushdown store depth in q''. This relation between two states is not symmetric but is transitive. See Figure 1 shows an example of various local connections. Note that for example the path $1 \rightarrow 7 \rightarrow 8$ is not a local connection. The notation $a|\alpha \mapsto \beta$ denotes a transition that reads symbol $a \in A$ and replaces $\alpha \in G'^*$ with $\beta \in G'^*$ on the top of the pushdown store. This notation will be used in figures throughout the paper.

It can be easily seen that the pushdown store top symbols are shared between locally connected states.

With local connection from q' to q and local connection from q to q'', the q' is locally connected to q'' by transitive closure. If $\mathcal{T}_q$ is the set of pushdown store top symbols of state q, then $\mathcal{T}_{q'} \subseteq T_q$ and $\mathcal{T}_q \subseteq \mathcal{T}_{q''}$ and also $\mathcal{T}_{q'} \subseteq \mathcal{T}_{q''}$.

The local closure of state q is $\lambda^*(q)$. See Figure 2 as an example of a local closure. See Figure 1 and note that the path $1 \rightarrow 7 \rightarrow 8$ mentioned before is in fact a local closure.

Closing all states under the $\lambda^*(q)$ connects all $\mathcal{T}_q$. See Figure 3.

We define these notions formally:

Definition 1: A local connection $\lambda(q)$ of state q. Given a deterministic visibly pushdown automaton $M' = (Q', A, G', \delta', q'_0, \perp, F')$, where $A = A_c \cup A_r \cup A_l$, states $q, q' \in Q'$, then $\lambda(q) = \{q' : (q, uw, \gamma_0) \vdash^k (q', w, \gamma_k), u \in A^*, w \in A^*, \gamma_0, \gamma_k \in G'^*, 1 \leq k, |\gamma_0| = |\gamma_k|, |\gamma_0| < |\gamma_i|, 1 \leq i < k\}$.

Definition 2: A local connection closure $\lambda^*(q)$ of state q. The local connection closure λ^* is defined by these equalities:

$$\lambda^0(q) = q, \tag{1}$$

$$\lambda^{i+1}(q) = \bigcup_{\forall q' \in \lambda^i(q)} \lambda(q'), \tag{2}$$

$$\lambda^*(q) = \bigcup_{i \geq 0} \lambda^i(q). \tag{3}$$

Definition 3: A set of pushdown store top symbols $\mathcal{T}_q$ of state q. Given a deterministic visibly pushdown automaton $M' = (Q', A, G', \delta', q'_0, \perp, F')$, where $A = A_c \cup A_r \cup A_l$, states $q, q', q'' \in Q'$, then $T_q = \{g : (q', g) \in \delta(q'', c, \varepsilon), c \in A_c, g \in G', q' \in \lambda^*(q)\}$.

Due to convenient properties of $\lambda^*(q)$, $\mathcal{T}_q$ can be stored in a space optimal way. Given $q, q' \in Q'$, then $\forall q' \in \lambda^*(q)$ holds that $\mathcal{T}_{q'} \subseteq \mathcal{T}_q$, ie parts of T_q can be shared between locally connected and locally closed states. See Figure 4 as an example of this process.

Further, we show by induction on the length of an input sentence that pushdown store top symbols are given by the $\lambda^*(q)$ and states that are source and target of the appropriate push and pop transition.

Step one:

$$\varepsilon : \perp \in \mathcal{T}_{q'_0}. \tag{4}$$

Then, assume that pushdown store top symbols are given by the local closure for first i symbols of input word. Symbol a is a $(i+1)$-th symbol of input word. Given $q_1, q_2, q_3, s_1, s_2, s_3 \in Q', a \in A_c$, then we have three distinct cases:

$$\lambda(r_1) = \{q_1 : (q_1, aw, \gamma) \vdash (r_1, w, \gamma)\}, \tag{5}$$

$$\mathcal{T}_{r_2} = \{(q_2, a) : (q_2, aw, \gamma) \vdash (r_2, w, (q_2, a)\gamma)\}, \tag{6}$$

$$\lambda(r_3) = \{q_2 : (q_3, aw, (q_2, a)\gamma) \vdash (r_3, w, \gamma)\}. \tag{7}$$

Notice that γ does not change between states q_1 and r_1. Pushdown store top symbol (q_2, s) was pushed in state q_2 and popped in state q_3 so γ does not change between states q_2 and r_3 either. Pushdown store top symbols are given by the $\lambda^*(q)$ and states that are source and target of appropriate push and pop transition for first $i+1$ symbols of input word. The induction holds for $i+1$.

More informally: The deterministic automaton is constructed from the initial state. The deterministic subset of the

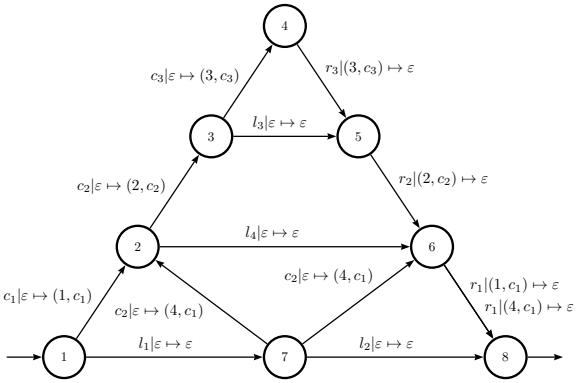

Fig. 1. Various λ relations of state 8 to state 1.

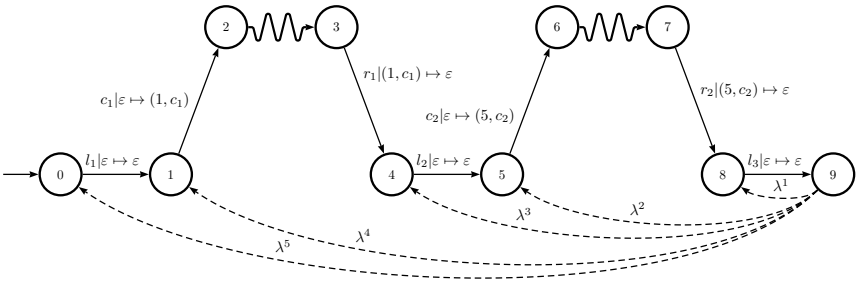

Fig. 2. The λ^* relation computed for state 9.

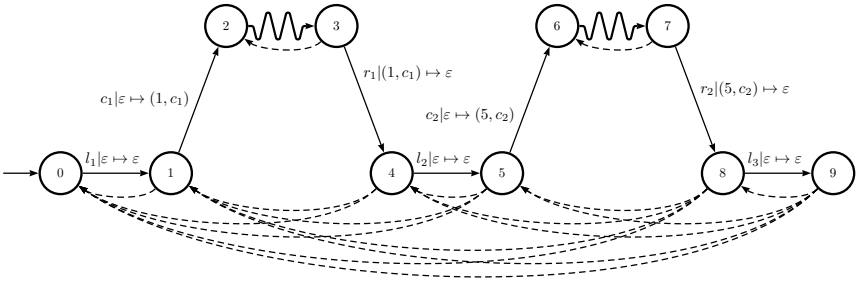

Fig. 3. The λ^* relation computed for all states.

initial state is created from all initial states of the original automaton. Initial pushdown store symbol $\perp \in \mathcal{T}_{q_0'}$ forms the pushdown store top symbols set.

In every iteration, all local and push transitions are explored for the known states. Base set of possible pushdown store top symbols $\mathcal{T}_q$ of the pushdown store for given state $q \in Q'$ is given by push transitions. Then, we track pushdown store top symbols for each state.

Any two states q, q', where a local transition exists from state q' to state q, share part of $\mathcal{T}_{q'}$ in form of that everything from $\mathcal{T}_{q'}$ is in $\mathcal{T}_q$.

Any two states q, q', where a pop transition popping symbol (q', r) exists from state x to state q, share part of $\mathcal{T}_{q'}$ in form of that everything from $\mathcal{T}_{q'}$ is in $\mathcal{T}_q$. Given $q, q', q'' \in Q', l \in A_l, c \in A_c, r \in A_r$, then for $\mathcal{T}$ the following properties hold:

$$\perp \in \mathcal{T}_{q_0'}, \tag{8}$$

$$\forall (q, \varepsilon) \in \delta(q', l, \varepsilon) \Rightarrow \mathcal{T}_{q'} \subseteq \mathcal{T}_q, \tag{9}$$

$$\forall (q, \varepsilon) \in \delta(q'', r, (q', c)) \Rightarrow \mathcal{T}_{q'} \subseteq \mathcal{T}_q, \tag{10}$$

$$\forall (q, (q'', c)) \in \delta(q'', c, \varepsilon) \Rightarrow (q'', c) \subseteq \mathcal{T}_q. \tag{11}$$

The algorithm of the determinisation is formally described by Algorithm 1. Given $q, q'' \in Q'$, its main part can be written in an abstract way as follows:

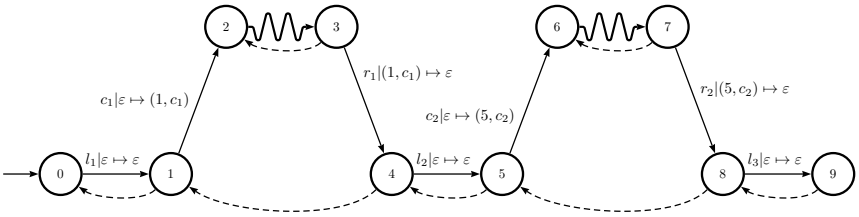

Fig. 4. Connecting T_q between locally connected states.

1) create the initial state,
2) while $\nexists(q',\varepsilon)\in\delta(q,r,\gamma)$, where $r\in A_r$ and $\gamma\in\mathcal{T}_q$, do create pop transition (q,r,γ),
   while $\nexists(q',\varepsilon)\in\delta(q,l,\varepsilon)$, where $l\in A_l$, do create local transition (q,l,ε),
   while $\nexists(q',\gamma)\in\delta(q,c,\varepsilon)$, where $c\in A_c$, do create push transition (q,c,ε),
3) set final states.

Only a pair of states (q',q), where $q'\in\lambda^*(q)$, can appear as an element in S component. As it is described in II, the new pairs of S component are only created from push and local transitions. The push transitions only yield elements of S component based on identity. On the other hand, the local transitions yield exactly the pairs that conform to local closure, because they connect appropriate targets of push and sources of pop operations.

Let L be the set of all distinct pairs of locally closed states of the nondeterministic automaton. Then, $2^{|L|}$ is the maximum number of states of the deterministic automaton. The $|L|$ is at most n^2 when all states in the nondeterministic automaton are locally closed.

V. Example

In this section the determinisation of a simple nondeterministic visibly pushdown automaton is demonstrated. For the sake of clarity, the component R of states and pushdown store symbols is omitted, because the information it holds is already contained in the component S (it is the second value of each pair), as it was described in [12].

Example 5.1: The nonderterministic visibly pushdown automaton a_1 is shown in Figure 6.

Let $a_1=(Q,A_c\cup A_r\cup A_l,G,\delta,Q_0,\bot,F)$ be a nondeterministic visibly pushdown automaton. An equivalent deterministic visibly pushdown automaton $d_1=(Q',A_c\cup A_r\cup A_l,G',\delta',q'_0,\bot,F')$ can be constructed as follows.

The initial state is constructed as powerset of all identity pairs of initial states of automaton a_1, so $q'_0=\{(0,0)\}$.

The push and the local transitions can be easily deduced from determinisation rules from Section IV. All transitions are shown in Figure 5.

In each push transition, the pushdown store top symbol is tracked for target state of the push transition. When a local transition occurs, all pushdown store top symbols are shared from a source state of the local transition to a target state of the local transition and all other locally connected states. The tracking of all locally connected states could be achieved by creating virtual transitions serving as a transitive closure.

Then, the pop transitions are created based on known input symbols (from the nondeterministic automaton) and tracked pushdown store top symbols. The transitions are created according to the determinisation rules from Section IV.

For an illustration of the determinisation see Figure 5. The pushdown store top symbols are tracked as follows. Push, local and pop transitions are marked green, gray and red, respectively. Arrows describe movements of tracked tops of the pushdown store. Dashed arrows represents source of top pushdown store symbols that are shared with target state when local transition occur.

The resultant deterministic PDA d_1 is shown in Figure 7.

The resultant deterministic PDA can be also reproduced by running Algorithm 1.

Given the nondeterministic pushdown automaton from the example above, the VPAlib algorithm [9] constructs a deterministic pushdown automaton with 45 states and 1206 transitions. We note that 45 (states) is not a power of 2, which is caused by the fact that the implementation of VPAlib library does not consider states in which components R or S are empty sets (in this way, it performs another optimization of the determinisation algorithm [12]). For comparison, our algorithm constructs an equivalent deterministic pushdown automaton with only 3 states and 8 transitions. This is a significant improvement over the previously existing determinisation algorithms.

VI. Conclusion

A new incremental algorithm of the determinisation of nondeterministic visibly pushdown automata has been described. The algorithm creates only necessary states and pushdown symbols by analysing and tracking which states are achievable by computing transitions on the same levels of pushdown operations nesting. Possible tops of the pushdown store are stored for each state when a pop transition is in progress and then they are shared through local transitions with states on the same levels of the nesting. The behavior of the algorithm

Determinisation algorithm.
Input : A nondeterministic visibly pushdown automaton $M_n(Q, A_c \cup A_l \cup A_r, G, \delta, Q_0, \bot, F)$.
Output: An equivalent deterministic visibly pushdown automaton $M_d(Q', A_c \cup A_l \cup A_r, G', \delta', q'_0, \bot, F')$.
Method:
set $q'_0 = \{(x, x) : x \in Q_0\}, Q' = \{q'_0\}, G' = \emptyset, \delta' = \emptyset, F' = \emptyset$;
create queue $Dirty = ((q'_0, \bot))$, // the set of pairs (state, pushdown store symbol);
create set $Clean = \emptyset$, // the set of states;
while $Dirty$ *is not empty* **do**
  $(state, symbol)$ = dequeue $Dirty$;
  create set $States = \emptyset$;
  if $state \notin Clean$ **then**
    forall the $l \in A_l$ **do**
      forall the $(q, q_2) \in state$ **do**
        $state2 = \{(q, q_1) : (q_1, \varepsilon) \in \delta(q_2, l, \varepsilon)\}$; **if** $state2 = \emptyset$ **then** continue;
        add $state2$ to Q'; add $(state2, \varepsilon)$ to $\delta'(state, l, \varepsilon)$; add $state2$ to $States$;
        update $T_x, x \in \lambda^*(state2)$;
      end
    end
    forall the $c \in A_c$ **do**
      forall the $(q, q_2) \in state$ **do**
        $state2 = \{(q_1, q_1) : (q_1, g) \in \delta(q_2, c, \varepsilon)\}$; **if** $state2 = \emptyset$ **then** continue;
        add $state2$ to Q'; add $(state, c)$ to G';
        add $(state2, (state, c))$ to $\delta'(state, c, \varepsilon)$; add $state2$ to $States$;
        update $T_x, x \in \lambda^*(state2)$;
      end
    end
    add $state$ to $Clean$;
  end
  forall the $r \in A_r$ **do**
    if *symbol is* $\bot$ **then**
      forall the $(q, q_2) \in state$ **do**
        $state2 = \{(q, q_1) : (q_1, \varepsilon) \in \delta(q_2, r, \bot)\}$; **if** $state2 = \emptyset$ **then** continue;
        add $state2$ to Q'; add $(state2, (state, \varepsilon))$ to $\delta'(state, r, symbol)$; add $state2$ to $States$;
        update $T_x, x \in \lambda^*(state2)$;
      end
    else
      create set $Update = \emptyset$;
      forall the $(q_1, q_2) \in state$ **do**
        $pairs = \{(q, q_I) : (q_I, g) \in \delta(q_2, r, g),\ (q_1, \varepsilon) \in \delta(q, c, g),\ c = second(symbol)\}$;
        add $pairs$ to $Update$;
      end
      forall the $(q, q_3) \in first(source)$ **do**
        $state2 = \{(q, q_2) : (q_3, q_2) \in Update\}$; **if** $state2 = \emptyset$ **then** continue;
        add $state2$ to Q'; add $(state2, (state, \varepsilon))$ to $\delta'(state, r, symbol)$; add $state2$ to $States$;
        update $T_x, x \in \lambda^*(state2)$;
      end
    end
  end
  forall the $s \in States$ **do**
    forall the $g \in T_{state} \setminus T_s$ **do** enqueue (s, g) to $Dirty$;
  end
end
forall the $q \in Q'$ **do**
  if *exists* $(q_1, q_2) \in q$ *such that* $q_2 \in F$ **then** add q into F' ;
end

Algorithm 1: Determinisation

Fig. 5. Construction of deterministic automaton d_1.

	State q	$\{(0,0)\}$	$\{(0,0),(1,1)\}$	$\{(0,1)\}$
	Pushdown store top symbols T_q	$\perp$	$\perp$, $(a,\{(0,0)\})$, $(a,\{(0,0)\})$, $(a,\{(0,0),(1,1)\})$	$(a,\{(0,0)\})$, $(a,\{(0,0),(1,1)\})$
1.	$a\|\varepsilon \mapsto (a,\{(0,0)\})$	$\{(0,0),(1,1)\}$		
2.	$c\|\varepsilon \mapsto \varepsilon$	$\{(0,1)\}$	$\{(0,1)\}$	
3.	$a\|\varepsilon \mapsto (a,\{(0,0),(1,1)\})$		$\{(0,0),(1,1)\}$	
4.	$d\|(a,\{(0,0),(1,1)\}) \mapsto \varepsilon$		$\{(0,1)\}$	
5.	$d\|(a,\{(0,0)\}) \mapsto \varepsilon$		$\{(0,1)\}$	
4.	$b\|(a,\{(0,0),(1,1)\}) \mapsto \varepsilon$			$\{(0,1)\}$
6.	$b\|(a,\{(0,0)\}) \mapsto \varepsilon$			$\{(0,1)\}$

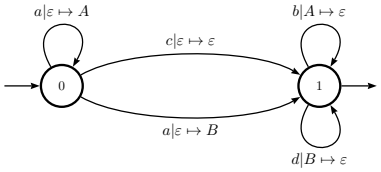

Fig. 6. The nondeterministic visibly pushdown automaton a_1 from Example 5.1.

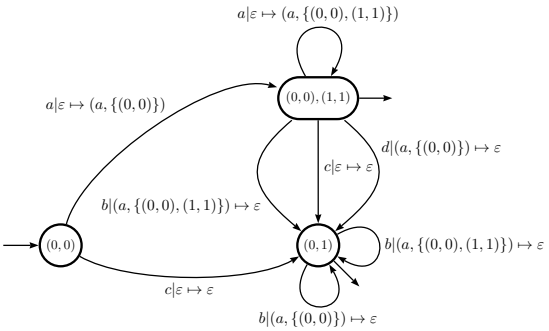

Fig. 7. The resultant deterministic PDA d_1 from Example 5.1 created by determinisation of automaton a_1.

is inspired by the behavior of the incremental construction of the deterministic finite automaton.

The algorithm has been implemented as a part of an experimental automata library [13].

Although the number of states of the deterministic automaton for the worst case is still 2^{n^2} for a given nondeterministic visibly pushdown automaton with n states, it has been shown that the upper bound of the number of states of the deterministic automaton is dependant on the number of distinct pairs of locally connected states. For those and other reasons in many practical cases our algorithm provides significantly smaller deterministic automata than the previously existing determinisation algorithms.

Furthermore, a similar approach can be adapted for the determinisation of Height-deterministic pushdown automata [10].

Further information can be found on [4].

REFERENCES

[1] R. Alur. Marrying words and trees. In J. Meseguer and G. Rosu, editors, *Algebraic Methodology and Software Technology, 12th International Conference, AMAST 2008, Urbana, IL, USA, July 28-31, 2008, Proceedings*, volume 5140 of *Lecture Notes in Computer Science*, page 1. Springer, 2008.

[2] R. Alur, V. Kumar, P. Madhusudan, and M. Viswanathan. Congruences for visibly pushdown languages. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 1102–1114. Springer, 2005.

[3] R. Alur and P. Madhusudan. Visibly pushdown languages. In L. Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 202–211. ACM, 2004.

[4] Arbology www pages. Available on: http://arbology.fit.cvut.cz/, 2015. May 2015.

[5] V. Bárány, C. Löding, and O. Serre. Regularity problems for visibly pushdown languages. In B. Durand and W. Thomas, editors, *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, volume 3884 of *Lecture Notes in Computer Science*, pages 420–431. Springer, 2006.

[6] D. Debarbieux, O. Gauwin, J. Niehren, T. Sebastian, and M. Zergaoui. Early nested word automata for xpath query answering on XML streams. *Theor. Comput. Sci.*, 578:100–125, 2015.

[7] E. Filiot, J. Raskin, P. Reynier, F. Servais, and J. Talbot. Properties of visibly pushdown transducers. In P. Hlinený and A. Kucera, editors, *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 355–367. Springer, 2010.

[8] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesley, second edition, 2001.

[9] H. Nguyen. Visibly pushdown automata library. Available on: http://www.emn.fr/z-info/hnguyen/vpa/, 2015. May 2015.

[10] D. Nowotka and J. Srba. Height-deterministic pushdown automata. In L. Kučera and A. Kučera, editors, *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Český Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, volume 4708 of *Lecture Notes in Computer Science*, pages 125–134. Springer, 2007.

[11] J. Srba. Visibly pushdown automata: From language equivalence to simulation and bisimulation. In Z. Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume

4207 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2006.

[12] N. V. Tang. A tighter bound for the determinization of visibly pushdown automata. In A. Legay, editor, *Proceedings International Workshop on Verification of Infinite-State Systems, INFINITY 2009, Bologna, Italy, 31th August 2009.*, volume 10 of *EPTCS*, pages 62–76, 2009.

[13] J. Trávníček. ALT: Automata (Algorithms) Library Toolkit, 2015.