Minimizing variants of visibly pushdown automata

Patrick Chervet, Igor Walukiewicz

Minimizing variants of visibly pushdown automata

Patrick Chervet and Igor Walukiewicz

LaBRI, Université de Bordeaux and CNRS
351, Cours de la Libération
F-33 405, Talence cedex, France

Abstract. The minimization problem for visibly pushdown automata (VPA) is studied. Two subclasses of VPA are introduced: call driven automata, and block automata. For the first class, minimization results are presented unifying and generalizing those present in the literature. A drawback of this class, and all the other classes known till now, is that it is exponentially less succinct than VPA. The second class, block automata, is introduced to address this problem. These automata are as succinct as VPA. A minimization procedure for them is presented that requires one additional parameter to be fixed. An example of an exponential gain in succinctness is presented.

Introduction

The class of visibly pushdown languages is the class of languages defined by pushdown automata where an input letter determines a stack action of the automaton. It seems that this class was first studied by Melhorn under the name of input driven automata. In [11] he shows that the parsing problem is in $\mathcal{O}(\log^2(n))$. Alur and Madhusudan [2] exhibit many good properties of this class. It is closed under boolean operations and it contains some interesting previously studied classes as: parenthesis languages [10] and balanced grammars [5]. Visibly pushdown languages have several different characterizations. One is via syntactic congruences in a style of Myhill-Nerode congruence for regular languages [3]. This characterization permits to obtain a canonical visibly pushdown automaton for a given language. Unfortunately, this canonical automaton is not always the minimal visibly pushdown automaton for the language.

In this paper we study the minimization problem for deterministic VPA. Our research is motivated by the presence of two different subclasses in the literature: SEVPA [3] and MEVPA [8]. These are two subclasses of VPA for which some minimization results are known. We introduce two new classes: call driven automata (CDA), and their expanded version (eCDA). The class CDA is a superset of both SEVPA and MEVPA; while eCDA is included in these two classes. We prove a minimization result for eCDA and show how it can be used to get known and new minimization results for other three classes. This gives a unified picture of the previous studies of the minimization problem.

The drawback of all of the above results is that translation from deterministic VPA to automata in one of these classes may incur exponential blowup. This is due to structural constraints on the form of automata. We propose a new subclass of VPA, called block VPA (BVPA) which is much better in this respect. The translation from VPA to a BVPA results in at most quadratic blowup. Thus a minimization result for BVPA would give an approximative minimization of VPA. Unfortunately, we are able to show minimization of BVPA only when some additional parameter is fixed. The advantage of this result is that minimizations of eCDA and SEVPA are its special cases. It makes also evident that minimizing VPA is related to optimizing the one parameter that we exhibit.

Since the paper of Alur and Madhusudan [2], VPA has appeared in several contexts: XML [14, 7], verification [9, 1], learning [8], semantics of programing languages [13]. It is in particular this last paper that motivated the current work. In that paper an algorithm is given for constructing a VPA describing the semantics of a given program expression. This way comparing program expressions is reduced to VPA equivalence. For the feasibility of the translation from programs to VPA it is essential to be able to reduce the size of intermediate automata during construction. This leads directly to the problem of minimization. It is worth noting that in other applications mentioned above minimization can also play an important role.

Let us comment on the importance and feasibility of minimization in general. Small canonical ways of representing objects are omnipresent. Consider two examples from verification: Binary Decision Diagrams [12] (BDD's) are nothing else but minimal automata for languages of bit strings; difference bounded matrices [6] (DBM's) are canonical representations of sets clock valuations. Good representations are rare. For example, nondeterministic automata are in principle more succinct than deterministic automata, still it seems very difficult to obtain, with a reasonable computational effort, a nondeterministic automaton of size close to a minimal one. Similar problems appear with two way deterministic automata, or even with two-pass deterministic automata that read the word first from left to right and then from right to left. In this context having minimal automata even for a restricted class of VPA is rather unusual. The general minimization of VPA seems at least as difficult as minimization of automata two-pass automata.

The plan of the paper is as follows. We start with basic definitions on VPA. In the following section we introduce the new classes CDA and eCDA. We also show the minimization result for eCDA and point out how it implies a minimization result for CDA. Section 3 discusses relations with MEVPA and SEVPA. Finally, we present BVPA, discuss their properties and present a minimization procedure for them.

1 Visibly Pushdown Automata

A *visibly pushdown alphabet* $\hat{\Sigma} = (\Sigma_{call}, \Sigma_{ret}, \Sigma_{int})$ consists of three disjoint finite sets where Σ_{call} is a set of *calls*, Σ_{ret} is a set of *returns* and Σ_{int} is a set of *internal actions*.

For any such $\hat{\Sigma}$, let Σ denote $\Sigma_{call} \cup \Sigma_{ret} \cup \Sigma_{int}$. In the following we will use: $c, c_1, \ldots$ for elements of Σ_{call}; $r, r_1, \ldots$ for elements of Σ_{ret}; $i, i_1, \ldots$ for elements of Σ_{int}.

Definition 1. *A* visibly pushdown automaton *(VPA) is a pushdown automaton* $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F \rangle$*, where* Q *is a finite set of states,* $\hat{\Sigma} = (\Sigma_{call}, \Sigma_{ret}, \Sigma_{int})$ *is a visibly pushdown alphabet,* $q_0 \in Q$ *is an initial state,* Γ *is a (finite) stack alphabet,* Q_F *is a set of final states and* $\delta = \delta_{call} \cup \delta_{ret} \cup \delta_{int}$ *is a transition function, such that:* $\delta_{call} : Q \times \Sigma_{call} \to Q \times \Gamma$*,* $\delta_{ret} : Q \times \Sigma_{ret} \times \Gamma \to Q$ *and* $\delta_{int} : Q \times \Sigma_{int} \to Q$*.*

A stack over Γ will be represented by a finite word over Γ with the top on the left of the word. We will write $\sigma\sigma'$ for the concatenation of stacks σ and σ'. In particular $\gamma\sigma$ denotes a stack with the top letter being γ. A *configuration* is a pair (q, σ) where q is a state and σ is a stack.

An *execution* of a VPA $\mathcal{A}$ as above on a word $w = a_1 \cdots a_k$ from Σ^* is a sequence of configurations $(q_0, \sigma_0), \ldots, (q_k, \sigma_k)$ where σ_0 is the empty stack ε, and for every $j \in [1, k]$:

- if a_j is a call then $\delta_{call}(q_j, a_j) = (q_{j+1}, \gamma)$ and $\sigma_{j+1} = \gamma\sigma_j$,
- if a_j is an internal action then $\delta_{int}(q_j, a_j) = q_{j+1}$ and $\sigma_j = \sigma_{j+1}$,
- if a_j is a return then $\delta_{ret}(q_j, a_j, \gamma) = q_{j+1}$ and $\sigma_j = \gamma\sigma_{j+1}$.

Intuitively, on reading a call the automaton is obliged to do a push operation and moreover it cannot look at the top of the stack. On internal actions the automaton cannot change the stack, neither it can look at the top of it. When reading a return, the automaton has to do a pop, but this time it can use the information on the top of the stack. We will write $q \xmapsto{c/\gamma} q'$, $q \xmapsto{i} q'$, and $q \xmapsto{r/\gamma} q'$ for push, internal, and pop transitions, respectively.

An execution $(q_0, \sigma_0), \ldots, (q_k, \sigma_k)$ is *accepting* if q_k is a final state ($q_k \in Q_F$). A word $w \in \Sigma^*$ is *recognized* by an automaton $\mathcal{A}$ if the unique execution of $\mathcal{A}$ on w is accepting. The *language recognized by* $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of all words of Σ^* recognized by $\mathcal{A}$. A language L over an alphabet $\hat{\Sigma}$ is a *VPL* if there is a visibly pushdown automaton over the alphabet $\hat{\Sigma}$ recognizing L.

If $\mathcal{A}$ is a VPA and δ its transition function, we will write $\delta(u)$ to denote the state reached by $\mathcal{A}$ after the reading of $u \in \Sigma^*$. We will sometimes also use $\to_{\mathcal{A}}$ to denote the transition function of $\mathcal{A}$.

Remark 1. A visibly pushdown automaton is a deterministic pushdown automaton with one important restriction that input letters determine stack actions. The restrictions disallowing to look at the top of the stack when doing push or internal actions are not essential if recognizability is concerned as one can always remember the top of the stack in a state. One can also consider nondeterministic visibly pushdown automata, but we will not do it here.

Remark 2. Without a loss of generality, one can assume that $\Gamma = Q \times \Sigma_{call}$ and that in a state q when reading a call c the automation pushes (q, c) on the stack. This works as (q, c) is the maximal information the automaton has when doing the push. In the following we will use this form of Γ when convenient.

Definition 2 (Matched calls and returns). *Let $\hat{\Sigma}$ be a pushdown alphabet, and u be a word in Σ^*.*

The word u is matched calls *if every call has a matching return, i.e. if for every suffix u' of u the number of call symbols in u' is at most the number of return symbols of u'.*

Similarly, the word u is matched returns *if every return has a matching call, i.e. if for every prefix u' of u the number of return symbols in u' is at most the number of call symbols in u'.*

The word u is well-matched *if it is matched calls and matched returns. Observe that being well matched means being well bracketed when call symbols are considered as opening brackets and return symbols are considered as closing brackets.*

Let then $MC(\hat{\Sigma})$, $MR(\hat{\Sigma})$ and $WM(\hat{\Sigma})$ be respectively the set of matched calls, matched returns and well-matched words. A language L is well-matched *if $L \subseteq WM(\hat{\Sigma})$.*

Remark 3. As we forbid return transitions on empty stack, visibly pushdown automata can accept only matched returns words.

Given a VPL L over a visibly pushdown alphabet $\hat{\Sigma} = (\Sigma_{call}, \Sigma_{ret}, \Sigma_{int})$, let us define the equivalence relation $\approx_L$ on well-matched words:

$$w_1 \approx_L w_2 \quad \text{if for all } u, v \in \Sigma^* : uw_1v \in L \text{ iff } uw_2v \in L.$$

Observe that $\approx_L$ is a congruence with respect to the concatenation. We will sometimes omit the subscript L if it is clear from the context.

Theorem 1 ([3]). *A language $L \subseteq WM(\hat{\Sigma})$ is a VPL iff $\approx_L$ has finitely many equivalence classes.*

2 Call Driven Automata and their minimization

In this section we introduce the class of call driven automata. A call driven automaton is a special kind of visibly pushdown automaton where we require that a call letter determines uniquely the state to which the automaton goes. We will show later that this subclass of VPA is larger than previously introduced subclasses.

Definition 3. *A VPA $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F \rangle$ is a* Call Driven Automaton (CDA) *if there is a function Target : $\Sigma_{call} \mapsto Q$ and an equivalence relation $\overset{Q}{\leftrightarrow}$ on Q such that :*

- *for all $c \in \Sigma_{call}$, $q_0 \overset{Q}{\nrightarrow} Target(c)$*

– *if* $q \xrightarrow{i} q'$ *then* $q \overset{Q}{\leftrightarrow} q'$,
– *if* $q \xrightarrow{c/(q,c)} q'$ *then* $q' = Target(c)$,
– *if* $q \xrightarrow{r/(q',c)} q''$ *then* $q' \overset{Q}{\leftrightarrow} q''$.

This definition essentially says that the set of states is divided into equivalence classes of $\overset{Q}{\leftrightarrow}$. An internal transition stays in the same equivalence class. A call transition goes to the class determined by the call letter. A return transition goes back to the class from which the matching call was done. The first condition says that the equivalence class of q_0 is not the target of any call, so being in this class we know that the stack is empty.

An interesting subclass of CDA, which we call expanded CDA, is obtained by requiring that each call leads to a different equivalence class. We will show in the next subsection that eCDA are very easy to minimize. Moreover, minimization of CDA can be done via minimization of eCDA (cf. Theorem 2).

Definition 4. *A VPA $\mathcal{A}$ is an* expanded Call Driven Automaton *(eCDA) if it is a CDA for some function Target and equivalence relation $\overset{Q}{\leftrightarrow}$ such that if $Target(c) \overset{Q}{\leftrightarrow} Target(c')$ then $c = c'$.*

2.1 Minimization of eCDA

Due to the restriction on their structure, minimization of eCDA resembles very much minimization of finite automata. Later, we will show that minimizations of other subclasses of visibly pushdown automata can be obtained via reduction to minimization of eCDA. As usual, by the size of an automaton we mean the number of its states.

Theorem 2. *Let $\hat{\Sigma}$ be a visibly pushdown alphabet. For every VPL $L \subseteq WM(\hat{\Sigma})$ there is a unique (up to isomorphism) minimum-size eCDA recognizing L.*

Proof. The proof uses the same method as minimization of SEVPA [3] but it is notationaly simpler, due to the simpler structure of eCDA. Let $\hat{\Sigma}$ and L be as in the assumption of the theorem. The proof is done in two steps. First, we construct a syntactic eCDA $\mathcal{A}$ recognizing L. We then prove its minimality by showing for any eCDA recognizing L a surjective homomorphism from it to $\mathcal{A}$.

To construct a syntactic eCDA, we define the following equivalence relations on well-matched words:

$$w_1 \sim_0 w_2 \quad \text{iff} \quad \forall v \in \Sigma^*.\ w_1 v \in L \Leftrightarrow w_2 v \in L,$$

and, for every $c \in \Sigma_{call}$:

$$w_1 \sim_c w_2 \quad \text{iff} \quad \forall u, v \in \Sigma^*.\ ucw_1 v \in L \Leftrightarrow ucw_2 v \in L.$$

We denote their equivalence classes by $[w]_0$ and $[w]_c$, respectively. All these relations include $\approx_L$ so by Theorem 1 they have finite index.

States of the syntactic eCDA will be the equivalence classes of theses relations. The formal construction of $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F \rangle$ is as follows

– $Q = \{(a, [w]_a) : a \in \Sigma_{call} \cup \{0\},\ w \in WM(\hat{\Sigma})\}$,
– $\Gamma = Q, \quad q_0 = (0, [\epsilon]_0), \quad Q_F = \{(0, [w]_0) : w \in L\}$,
– the transitions δ are as below (here $a \in \Sigma_{call} \cup \{0\}$):
  - $(a, [w]_a) \xrightarrow{i} (a, [wi]_a)$, for $i \in \Sigma_{int}$;
  - $(a, [w]_a) \xrightarrow{c/(a,[w]_a)} (c, [\epsilon]_c)$, for $c \in \Sigma_{call}$;
  - $(a, [w]_a) \xrightarrow{r/(b,[w']_b)} (b, [w'awr]_b)$, for $r \in \Sigma_{ret}$ and $b \in \Sigma_{call} \cup \{0\}$.

We need to check that $\mathcal{A}$ is well defined which comes to checking that δ is well defined. This in turn follows from the fact that for every $a, b \in \Sigma_{call} \cup \{0\}$, $i \in \Sigma_{int}$, $r \in \Sigma_{ret}$, every $w_1 \sim_a w_2$, and every $w_1' \sim_b w_2'$ we have:

$$w_1 i \sim_a w_2 i \quad \text{and} \quad w_1' a w_1 r \sim_b w_2' a w_2 r.$$

It is also straightforward to check that $\mathcal{A}$ is an eCDA when we define $\mathit{Target}(c) = (c, [\epsilon]_c)$ and $(a, [w]_a) \overset{\mathcal{O}}{\leftrightarrow} (b, [w']_b)$ iff $a = b$.

To prove that $\mathcal{A}$ recognizes L, it is sufficient to observe that the following invariant is maintained during an execution: after reading a word $u \in MR(\hat{\Sigma})$ the automaton $\mathcal{A}$ reaches the configuration (q, σ) where:

– if $u \in WM(\hat{\Sigma})$ then $q = (0, [u]_0)$ and σ is empty stack.
– otherwise u has a unique decomposition $u = w_0 c_1 w_1 \cdots c_l w_l$, where each $w_j \in WM(\hat{\Sigma})$ and each $c_j \in \Sigma_{call}$; in this case $q = (c_l, [w_l]_{c_l})$ and the stack is $\sigma = (c_{l-1}, [w_{l-1}]_{c_{l-1}}) \cdots (c_1, [w_1]_{c_1})(0, [w_0]_0)$.

As L is a well-matched VPL, any $u \in L$ is well matched. The invariant implies that after the reading of $u \in L$, the state of $\mathcal{A}$ is $(0, [u]_0)$, which is a final state by definition. Conversely, if when reading a word u, $\mathcal{A}$ reaches a final state $(0, [w]_0)$, then u is matched returns (otherwise the execution stops before reading the whole word u). Then the invariant shows that u needs to be well matched and thus after reading u the state is $(0, [u]_0)$. As $(0, [u]_0)$ is a final state, we have $u \in L$. Hence, the above invariant guarantees that $u \in L$ iff $\mathcal{A}$ recognizes u.

The second step is to show that $\mathcal{A}$ is the unique minimal eCDA recognizing L. Let $\mathcal{A}' = \langle Q', \hat{\Sigma}, \Gamma', q_0, \delta', Q_F' \rangle$ be an eCDA recognizing L. We will construct a surjective function from Q' to the states of $\mathcal{A}$ preserving the structure of $\mathcal{A}'$. We define:

$$f(q') = \begin{cases} (0, [w]_0) & \text{if there is } w \in WM(\hat{\Sigma}) \text{ such that } \delta'(w) = q' \\ (c, [w]_c) & \text{if there are } u \in \Sigma^*, c \in \Sigma_{call}, w \in WM(\hat{\Sigma}) \text{ such that} \\ & \delta'(ucw) = q' \end{cases}$$

First, we have to verify that f is well defined.

– If $\delta'(w) = \delta'(w')$, then the automaton is in the same configuration after reading w and w'. Hence, for every u, $\delta'(wu) = \delta'(w'u)$, and $w \sim_0 w'$.

– If $\delta'(ucw) = \delta'(u'cw')$ then for every u'', $\delta'(u''cw) = \delta'(u''cw')$, as when reading well matched words w and w' automaton cannot look at the parts of the stack created when reading uc, $u'c$ or $u''c$. Hence, for every $u'', v \in \Sigma^*$, $\delta'(u''cwv) = \delta'(u''cw'v)$, and $w \sim_c w'$.

Obviously, f is surjective. It is also straightforward to verify the following four conditions saying that f preserves the structure of $\mathcal{A}'$:

1. $f(q'_0) = q_0$ and for every $c \in \Sigma_{call}$, $f(\mathit{Target}'(c)) = \mathit{Target}(c)$.
2. For every $i \in \Sigma_{int}$, if $p' \xrightarrow{i}_{\mathcal{A}'} q'$ then $f(p') \xrightarrow{i}_{\mathcal{A}} f(q')$.
3. For every $c \in \Sigma_{call}$, if $p' \xrightarrow{c/(p',c)}_{\mathcal{A}'} q'$ then $f(p') \xrightarrow{c/(f(p'),c)}_{\mathcal{A}} f(q')$.
4. For every $r \in \Sigma_{ret}$, if $p' \xrightarrow{r/(s',c)}_{\mathcal{A}'} q'$ then $f(p') \xrightarrow{r/(f(s'),c)}_{\mathcal{A}} f(q')$.

The construction described in the proof above can be done in cubic time. We will not analyse it precisely but refer the reader to [4] where the cubic complexity of a more complicated construction is shown.

Corollary 1. *Given an eCDA, in cubic time it is possible to find a minimal equivalent eCDA.*

2.2 Minimization of CDA

An obvious question is whether one can minimize CDA in the same way as eCDA. The answer is negative as there is no unique minimal CDA for the language (see Example 2.2 below). Without much work though we can obtain an approximative minimization of CDA, i.e., minimization up to the factor $|\Sigma_{call}|$. The construction uses the minimization of eCDA.

Lemma 1. *Given a CDA of size n, an equivalent eCDA of size $\mathcal{O}(n \times |\Sigma_{call}|)$ can be constructed in linear time.*

Proof. Let $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F \rangle$ be a CDA for *Target* and $\overset{Q}{\leftrightarrow}$ recognizing $L \subseteq WM(\hat{\Sigma})$. We define an equivalent eCDA $\mathcal{B}$ recognizing L. We need to duplicate states so that $\overset{Q}{\leftrightarrow}$ relates no two states in the range of *Target*. Formally, $\mathcal{B} = \langle Q', \hat{\Sigma}, q'_0, \Gamma', \delta', Q'_F \rangle$ is such that:

– $Q' = \{(q, a) | a \in \Sigma_{call} \cup \{0\}, q \in Q\}$;
– $\Gamma' = \Gamma \times (\Sigma_{call} \cup \{0\})$, $q'_0 = (q_0, 0)$, $Q'_F = \{(q, 0) : q \in Q_F\}$;
– the transitions δ' are defined as follows (here $a, b \in \Sigma_{call} \cup \{0\}$):
  - $(p, a) \xrightarrow{i}_{\mathcal{B}} (q, a)$ if $p \xrightarrow{i}_{\mathcal{A}} q$,
  - $(p, a) \xrightarrow{c/(\gamma,a)}_{\mathcal{B}} (q, c)$ if $p \xrightarrow{c/\gamma}_{\mathcal{A}} q$,
  - $(p, a) \xrightarrow{r/(\gamma,b)}_{\mathcal{B}} (q, b)$ if $p \xrightarrow{r/\gamma}_{\mathcal{A}} q$.

$\mathit{Target}_{\mathcal{B}}$ is defined by $\mathit{Target}_{\mathcal{B}}(c) = (\mathit{Target}_{\mathcal{A}}(c), c)$ and $\overset{Q}{\leftrightarrow}_{\mathcal{B}}$ is defined by: $(p, a) \overset{Q}{\leftrightarrow}_{\mathcal{B}} (q, b)$ iff $a = b$. It is easy to check that $\mathcal{B}$ is an eCDA and recognizes the same language as $\mathcal{A}$. By definition its size is $|Q| \times |\Sigma_{call}|$.

Given a CDA, this lemma together with the Theorem 2 alow us to find the smallest equivalent eCDA. More, due to the Lemma 1, the smallest equivalent eCDA is in size $\mathcal{O}(n \times |\Sigma_{call}|)$, where n is the minimal size of an equivalent CDA. So we get the following corollary.

Corollary 2. *Given a CDA in a cubic time it is possible to find a CDA of size $|\Sigma_{call}| \times n$ where n is the size of a minimal equivalent CDA.*

The following example shows a language L such that there is no unique minimal CDA recognizing L. We take for L the VPL $c_1(1 \cdot 1)^* r + c_2 1^* r$ over the visibly pushdown alphabet $(\{c_1, c_2\}, \{r\}, \{1\})$.

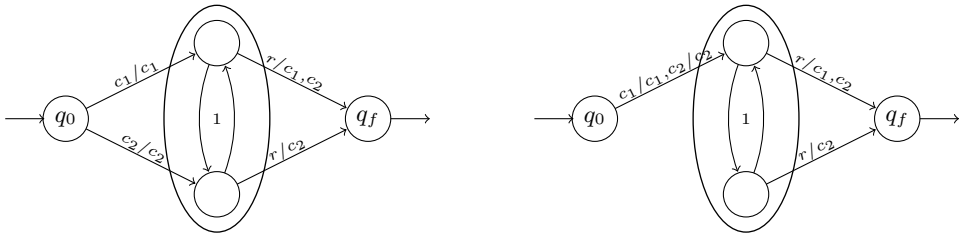


Fig. 1. Two non-isomorphic minimal CDA recognizing L (each automaton also has a sink that is not represented)

3 Comparison between different VPA subclasses

In this section we discuss the relations between CDA and two other classes of VPA introduced in the literature: single entry visibly pushdown automata (SEVPA) [3] and multi-entry visibly pushdown automata (MEVPA) [8]. Not only these two classes are included in CDA, but we can recover minimization results for them from our basic minimization result for eCDA.

A SEVPA is a CDA when every equivalence class of $\overset{Q}{\leftrightarrow}$ has a single entry, i.e. only one state in a class is a target of a push transition.

Definition 5. *A CDA with Target and $\overset{Q}{\leftrightarrow}$ is a* single entry visibly pushdown automaton (SEVPA) *if, for all call actions c and c':*

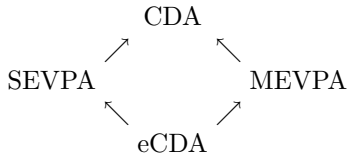
$$\textit{if } \mathit{Target}(c) \overset{Q}{\leftrightarrow} \mathit{Target}(c') \textit{ then } \mathit{Target}(c) = \mathit{Target}(c').$$

Multi-entry automata (MEVPA) represent another kind of restriction on CDA. In general we can assume that the stack alphabet of a visibly pushdown automaton is $Q \times \Sigma_{call}$ (cf. remark 2) as a push transition depends only on the current state and a letter being read. In MEVPA we assume that the symbol pushed on the stack depends only on the state and not on the input letter.

Definition 6. *A CDA $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F \rangle$ is a* Multi Entry Visibly Pushdown Automaton *(MEVPA) when for all transitions $q_1 \xrightarrow{c_1/\gamma_1} q_1'$ and $q_2 \xrightarrow{c_2/\gamma_2} q_2'$: if $q_1 = q_2$ then $\gamma_1 = \gamma_2$.*

Remark 4. Without a loss of generality one can assume that $\Gamma = Q$ and that in a state q when reading a call c the automaton pushes q on the stack. In the following we will use this form of Γ when convenient.

By definition, CDA includes all other classes: eCDA, SEVPA, MEVPA. Also by definition, eCDA is included in SEVPA. The class eCDA is also included in MEVPA. Indeed, if $(\mathcal{A}, \mathit{Target}, \overset{Q}{\leftrightarrow})$ is an eCDA: each state of $\mathcal{A}$ is related in $\overset{Q}{\leftrightarrow}$ to at most one state $\mathit{Target}(c)$, so the machine always knows what is the last call read, and does not need to put this information on the stack. This gives the following graph of inclusions between the subclasses.



For every VPA there is an equivalent CDA (cf. Theorem 2). The question is how small this CDA can be. The following example shows that the blow-up can be exponential; which is also an upper bound.

Example Consider an alphabet $\hat{\Sigma} = (\{c\}, \{r\}, \{a_1, \ldots, a_k\})$ with one call symbol c and one return symbol r. Let $L_k = a_1 c L_{a_1} r + \cdots + a_k c L_{a_k} r$, where $L_{a_i} \subseteq \{a_1, \ldots, a_n\}^*$ is the set of words where the number of a_i is even. By counting equivalence classes one can show that the minimal eCDA recognizing L_k is of size bigger than 2^k. Lemma 1 gives the same bound also for CDA . On the other hand, L_k can be recognized by a VPA of size in $\mathcal{O}(k^2)$ (see figure 3 below).

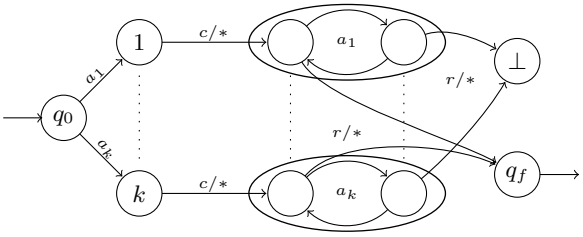


Fig. 2. VPA of size in $\mathcal{O}(k^2)$ recognizing L_k

4 Other results on MEVPA and SEVPA

The object of this section is to show how to reprove known minimization results on MEVPA and SEVPA using the result about eCDA. The main idea is to use an appropriately chosen injective morphism $\Phi : \Sigma^* \to \Lambda^*$ where Λ is some alphabet. The idea of the construction is presented in the schema in Figure 3. Given an

automaton $\mathcal{A}$ from a class $\mathcal{C}$, one constructs an eCDA $\overline{\mathcal{A}}$ recognizing $\Phi(L)$. Then one finds the minimal eCDA $\overline{\mathcal{B}}$ equivalent to $\overline{\mathcal{A}}$, and finally translates $\overline{\mathcal{B}}$ to an automaton $\mathcal{B}$ recognizing L. If Φ is suitably chosen then the size of $\overline{\mathcal{A}}$ can be bounded by a small polynomial in the size of $\mathcal{A}$ and usually the translation from $\overline{\mathcal{B}}$ to $\mathcal{B}$ does not introduce extra states.

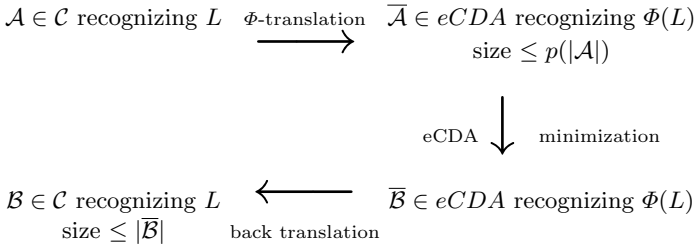


Fig. 3. Translation method: $p(n)$ is a fixed polynomial.

4.1 MEVPA

We explain how to obtain a minimization of MEVPA using eCDA.

Theorem 3 (Kumar, Madhusudan & Viswanathan). *[8] Let $\hat{\Sigma}$ be a visibly pushdown alphabet. For every VPL $L \subseteq WM(\hat{\Sigma})$, there is a unique (up to isomorphism) minimum-size equivalent MEVPA. Moreover, it can be constructed in a cubic time from a given MEVPA recognizing L.*

Proof. We apply the translation method for some homomorphism Φ.

Let $L \subseteq WM(\hat{\Sigma})$ be a VPL. Every VPA is equivalent to some eCDA (cf Theorem 2). Now an eCDA is in particular a MEVPA (see section 3 for a discussion of relations between different classes of VPA). Hence L is recognizable by some MEVPA. So let us fix a MEVPA $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma = Q, \delta, Q_F \rangle$ together with *Target* and $\overset{Q}{\leftrightarrow}$, such that $L(\mathcal{A}) = L$. $\hat{\Lambda}$ is the visibly pushdown language defined by $\Lambda_{int} = \Sigma_{int} \cup \Sigma_{call}$, $\Lambda_{call} = \{1\}$, $\Lambda_{ret} = \Sigma_{ret}$, and Φ is a morphism that is an identity on all letters but on the call letters where we put $\Phi(c) = 1c$. $\Phi(L)$ is a $WM(\hat{\Lambda})$ language.

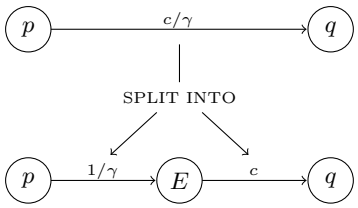


Fig. 4. $c \to 1c$ translation.

We now construct from $\mathcal{A}$ an eCDA $\overline{\mathcal{A}}$ recognizing $\Phi(L)$. The idea of the translation is that each call transition on a letter c is split into a 1 transition that does the push and goes to state E and then a c transition (see figure 4.1), that is now an internal transition. We do not need to make copies of $\mathcal{A}$, because when a push is done the automaton does not need to remember which call symbol is read. We will only add three distinguished states: an entry state E and two sinks $\bot_0$ (empty stack sink) and $\bot_1$ (non-empty stack sink). The construction of $\overline{\mathcal{A}} = \langle Q', \hat{\Lambda}, q_0, \Gamma', \delta', Q_F \rangle$ is as follows:

- $Q' = Q \cup \{E, \bot_0, \bot_1\}$;
- $\Gamma' = Q'$;
- the transitions δ' are defined as follows:
  - $p \xrightarrow{i}_{\overline{\mathcal{A}}} q$ if $p \xrightarrow{i}_{\mathcal{A}} q$,
  - $E \xrightarrow{c}_{\overline{\mathcal{A}}} Target(c)$ if $c \in \Sigma_{call}$,
  - $s \xrightarrow{1/s}_{\overline{\mathcal{A}}} E$ if $s \in Q'$,
  - $p \xrightarrow{r/q}_{\overline{\mathcal{A}}} s$ if $p \xrightarrow{r/q}_{\mathcal{A}} s$,
  - If not otherwise specified a transition goes to $\bot_0$ if the stack gets empty and to $\bot_1$ otherwise.

It is straightforward to check that $\overline{\mathcal{A}}$ is an eCDA when $Target_{\overline{\mathcal{A}}}(1) = E$ and $\overset{Q}{\leftrightarrow}_{\overline{\mathcal{A}}}$ has two equivalence relations: one is $\{q | \exists c \in \Sigma_{call}, q \overset{Q}{\leftrightarrow} Target(c)\} \cup \{E, \bot_1\}$ and the other one contains the rest of the states of Q'. The fact that the language recognized by the eCDA $\overline{\mathcal{A}}$ is L comes from the following property, which is easy to prove by induction. For every $v \in WM(\hat{\Lambda})$:

- if there is $u \in MR(\hat{\Sigma})$ such that $v = \Phi(u)$ then $\delta'(v) = \delta(u)$,
- if there is $u \in MR(\hat{\Sigma})$ such that $v = \Phi(u)1$ then $\delta'(v) = E$,
- otherwise $\delta'(v)$ is a sink.

Let $\overline{\mathcal{B}}$ be the minimal eCDA recognizing $\Phi(L)$ (here we use Theorem 2). An eCDA is in particular a MEVPA (again, see section 3), so without a loss of generality the stack alphabet of $\overline{\mathcal{B}}$ can be supposed to be its set of states. We now convert $\overline{\mathcal{B}}$ to a MEVPA $\mathcal{B}$ recognizing L. The construction is very similar to that for minimization of CDA. To fix the notation suppose that $\overline{\mathcal{B}} = \langle Q^b, \hat{\Lambda}, q_0^b, \Gamma^b = Q^b, \rightarrow_{\overline{\mathcal{B}}}, Q_f^b \rangle$ and that it comes with $\overline{Target}$ and $\overset{Q}{\leftrightarrow}$ defined. Automaton $\mathcal{B}$ will be obtained by changing just $\rightarrow_{\overline{\mathcal{B}}}$ into $\rightarrow_{\mathcal{B}}$ as follows:

- $q \xrightarrow{c/q}_{\mathcal{B}} Target(c)$, where $Target(c) = q$ if $\overline{Target}(1) \xrightarrow{c}_{\overline{\mathcal{B}}} q$;
- $\rightarrow_{\mathcal{B}}$ is the same as $\rightarrow_{\overline{\mathcal{B}}}$ on Σ_{int} and Σ_{ret}.

By definition $\mathcal{B}$ is a CDA with $Target$ defined as above and $\overset{Q}{\leftrightarrow}$. Here, transitions that had been split during translation have been collapsed back. Then the language recognized by $\mathcal{B}$ is obviously L. Note that both $\overline{\mathcal{B}}$ and $\mathcal{B}$ are the same no matter what $\mathcal{A}$ we start with. So the size of $\mathcal{B}$ is bounded from the above by $n + 3$, where n is the minimal size of a MEVPA equivalent to $\mathcal{A}$. Note that

during back translation some states of $\overline{\mathcal{B}}$ could become inaccessible. So we suppose every inaccessible states of $\mathcal{B}$ have been removed in the end of the back translation.

To prove that $\mathcal{B}$ is the unique (up to isomorphism) minimal MEVPA recognizing L, we show for every such $\mathcal{A}$ a surjective function from states of $\mathcal{A}$ to states of $\mathcal{B}$ that preserves the structure of $\mathcal{A}$. Without a loss of generality one can assume that all the states of $\mathcal{A}$ are accessible. Let in be the injection from states of $\mathcal{A}$ to states of $\overline{\mathcal{A}}$ that maps q to $(q, 0)$. We denote by f the function defined in the proof of Theorem 2 mapping states of $\overline{\mathcal{A}}$ to states of $\overline{\mathcal{B}}$. Then we define $\overline{f} = f \circ in$. To conclude, it is enough to prove the following facts (where $Q_{\mathcal{A}}$ and $Q_{\mathcal{B}}$ are respectively the set of states of $\mathcal{A}$ and $\mathcal{B}$):

1. $\overline{f}(Q_{\mathcal{A}}) \subseteq Q_{\mathcal{B}}$.
2. $\overline{f} : Q_{\mathcal{A}} \mapsto Q_{\mathcal{B}}$ is surjective.
3. $\overline{f}$ preserves the structure of $\mathcal{A}$.

Let u be a word in $MR(\hat{\Sigma})$. When proving that $\mathcal{A}$ recognizes $\Phi(L)$ we got $\delta_{\overline{\mathcal{A}}}(\Phi(u)) = in(\delta_{\mathcal{A}}(u))$. Then, together with the definition of f (cf proof of Theorem 2), this implies: $\delta_{\overline{\mathcal{B}}}(\Phi(u)) = f(in(\delta_{\mathcal{A}}(u))) = \overline{f}(\delta_{\mathcal{A}}(u))$. Now it is straightforward by induction to prove that: $\delta_{\mathcal{B}}(u) = \delta_{\overline{\mathcal{B}}}(\Phi(u))$. Hence, for every $u \in MR(\hat{\Sigma})$, $\delta_{\mathcal{B}}(u) = \overline{f}(\delta_{\mathcal{A}}(u))$. Then: as every state of $\mathcal{A}$ is accessible, 1 is verified; 2 is obviously verified; if we normalize stack alphabets of $\mathcal{A}$ and $\mathcal{B}$ (cf remark 2), the four conditions saying that a function preserves the structure of an automaton (cf proof of Theorem 2) are obviously verified, and 3 is verified too.

Note that the minimization of $\overline{\mathcal{A}}$ into $\overline{\mathcal{B}}$ and the accessibility test can be done in $\mathcal{O}(|\mathcal{A}|^3)$, and other steps of this algorithm in linear time. So it is possible to construct $\overline{\mathcal{B}}$ in $\mathcal{O}(|\mathcal{A}|^3)$ time, where $\mathcal{A}$ is any CDA recognizing L.

4.2 SEVPA

In [3] Alur, Kumar, Madhusudan and Viswanathan give a mimization of SEVPA provided some additional structure is preserved. We show how to obtain this result using the method of translations. Before doing this we present two remarks. The first shows that preserving the structure can result in exponential blowup. The second points out that our result on CDA gives approximative minimization that avoids the blow-up.

For a SEVPA $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F \rangle$ with *Target* and $\overset{Q}{\leftrightarrow}$ we define an equivalence relation $\overset{\Sigma}{\leftrightarrow}$ on call actions Σ_{call} by:

$$c \overset{\Sigma}{\leftrightarrow} c' \text{ iff } \mathit{Target}(c) = \mathit{Target}(c').$$

We then call $\mathcal{A}$ a $\overset{\Sigma}{\leftrightarrow}$-*SEVPA*. Equivalence relation $\overset{\Sigma}{\leftrightarrow}$ fixes bit more of the structure of automaton. One can consider eCDA as SEVPA with $\overset{\Sigma}{\leftrightarrow}$ that is the identity relation.

Remark 5. It may happen that minimal SEVPA for one $\overset{\Sigma}{\leftrightarrow}$ relation can be much bigger than an equivalent SEVPA for some other relation. Consider visibly pushdown alphabet $\hat{\Sigma} = (\{c_1, \ldots, c_k\}, \{r\}, \{a_1, \ldots, a_k\})$, and a language

$L'_k = a_1 c_1 L_{a_1} r + \cdots + a_k c_k L_{a_k} r$, where $L_{a_i} \subseteq \{a_1, \ldots, a_k\}^*$ is the set of words with even number of occurences of a_i. When $\overset{\Sigma}{\leftrightarrow}$ is the identity relation, the minimal size of a $\overset{\Sigma}{\leftrightarrow}$-SEVPA recognizing L is in $\mathcal{O}(k^2)$. If on the other hand $\overset{\Sigma}{\leftrightarrow}$ is a complete relation where every pair of call letters is related then the size of the minimal $\overset{\Sigma}{\leftrightarrow}$-SEVPA is bigger than 2^k .

Remark 6. Lemma 2 implies that taking $\overset{\Sigma}{\leftrightarrow}$ to be the identity relation is not far from optimal. Indeed, every SEVPA is a CDA, so the minimal eCDA for a given language is only $|\Sigma_{call}|$ bigger than the minimal equivalent SEVPA. As noted above an eCDA is a $\overset{\Sigma}{\leftrightarrow}$-SEVPA when $\overset{\Sigma}{\leftrightarrow}$ is the identity relation. So modifying $\overset{\Sigma}{\leftrightarrow}$ relation we can gain at most a $|\Sigma_{call}|$ factor.

The following result is the main minimization result of [3].

Theorem 4 (Alur & Madhusudan). *Let $\hat{\Sigma}$ be a visibly pushdown alphabet and let $\overset{\Sigma}{\leftrightarrow}$ be an equivalence relation over Σ_{call}. For every VPL $L \subseteq WM(\hat{\Sigma})$ there is a unique (up to isomorphism) minimum-size $\overset{\Sigma}{\leftrightarrow}$-SEVPA recognizing L. Moreover it can be constructed in a cubic time given a $\overset{\Sigma}{\leftrightarrow}$-SEVPA for the language.*

Proof. The proof is very similar to that for minimization of MEVPA, but uses a different homomorphism Φ for the translation.

Let $\hat{\Sigma}$ be a visibly pushdown alphabet, let $L \subseteq WM(\hat{\Sigma})$ be a VPL, and let $\overset{\Sigma}{\leftrightarrow}$ be an equivalence relation over Σ_{call}. Due to Theorem 2, every VPA is equivalent to some eCDA. Now, an eCDA is in particular a SEVPA (see Section 3) with respect to the identity relation on Σ_{call}. With a simple transformation, which we will not detail here, an eCDA can be transformed to $\overset{\Sigma}{\leftrightarrow}$-SEVPA for arbitrary equivalence relation $\overset{\Sigma}{\leftrightarrow}$. Hence, L is recognizable by some $\overset{\Sigma}{\leftrightarrow}$-SEVPA. We fix a SEVPA $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F \rangle$ together with *Target* and relations $\overset{Q}{\leftrightarrow}$, $\overset{\Sigma}{\leftrightarrow}$, such that $L(\mathcal{A}) = L$.

We will use the method of translations (see Figure **??**) but to SEVPA in place of CDA. Let $T \subseteq Q$ be the range of *Target*, and let $\hat{\Lambda}$ be the visibly pushdown alphabet defined by:

$$\Lambda_{int} = \Sigma_{int} \cup \Sigma_{call}, \quad \Lambda_{call} = T, \quad \Lambda_{ret} = \Sigma_{ret}$$

(where T is supposed to be disjoint from Σ). We take a homomorphism Φ that is an identity on all the letters but Σ_{call} for which we let $\Phi(c) = ct$ where $t = \mathit{Target}(c)$. It is easy to verify that $\Phi(L)$ is a $WM(\hat{\Lambda})$ language.

We now construct an eCDA $\overline{\mathcal{A}}$ over $\hat{\Lambda}$ recognizing $\Phi(L)$. When reading c we need to remember this letter in the state, so for every $q \in Q$ we will encode each call $c \in \Sigma_{call}$ in a new state (q, c) and also add an associated sink $(q, \perp)$. For notational reasons, a state $q \in Q$ will be renamed into $(q, 0)$. The formal construction of $\overline{\mathcal{A}} = \langle Q', \hat{\Lambda}, q'_0, \Gamma', \delta', Q'_F \rangle$ is as follows:

- $Q' = Q \times (\Sigma_{call} \cup \{0, \perp\})$;

– $\Gamma' = Q'$, $q'_0 = (q_0, 0)$ and $Q'_F = \{(q,0) | q \in Q_F\}$;
– where the transitions δ' are:
  - $(p,0) \xrightarrow{i}_{\overline{\mathcal{A}}} (q,0)$ if $p \xrightarrow{i}_{\mathcal{A}} q$,
  - $(q,0) \xrightarrow{c}_{\overline{\mathcal{A}}} (q,c)$, if $q \in Q$ and $c \in \Sigma_{call}$,
  - $(p,c) \xrightarrow{t/(p,c)}_{\overline{\mathcal{A}}} (t,0)$ if $(p,c) \in Q'$ and $t \in T$,
  - $(q,0) \xrightarrow{r/(p,c)}_{\overline{\mathcal{A}}} (q',0)$ if $q \xrightarrow{r/(p,c)}_{\mathcal{A}} q'$ and $q \overset{Q}{\leftrightarrow} \mathit{Target}(c)$,
  - $(q,c) \xrightarrow{a}_{\overline{\mathcal{A}}} (q,\bot)$ if $q \in Q$, $c \in \Sigma_{call} \cup \{\bot\}$ and $a \in \Lambda_{int}$,
    $(q,c) \xrightarrow{r/(p,c')}_{\overline{\mathcal{A}}} (p,\bot)$ if $(q,c),(p,c') \in Q'$ with $c \neq 0$ or $q \overset{Q}{\nleftrightarrow} \mathit{Target}(c')$.

The idea of the translation is that each call transition on letter c is split into a c transition that is now an internal transition, and a t transition that does the push. It is straightforward to check that $\overline{\mathcal{A}}$ is an eCDA for $\mathit{Target}_{\overline{\mathcal{A}}}$ and $\overset{Q}{\leftrightarrow}_{\overline{\mathcal{A}}}$ defined by:

– for every $t \in T$, $\mathit{Target}_{\overline{\mathcal{A}}}(t) = (t,0)$;
– for every $(p,c),(q,c') \in Q'$, $(p,c) \overset{Q}{\leftrightarrow}_{\overline{\mathcal{A}}} (q,c')$ iff $p \overset{Q}{\leftrightarrow}_{\mathcal{A}} q$.

To see that $\overline{\mathcal{A}}$ recognizes $\Phi(L)$ we show that for every $v \in WM(\hat{\Lambda})$:

– if $v = \Phi(u)$ for some $u \in MR(\hat{\Sigma})$ then $\delta'(v) = (\delta(u), 0)$,
– if $v = \Phi(u)c$ for some $u \in MR(\hat{\Sigma})$ and $c \in \Sigma_{call}$ then $\delta'(v) = (\delta(u), c)$,
– otherwise $\delta'(v)$ is a sink.

Let $\overline{\mathcal{B}}$ be the minimal eCDA recognizing $\Phi(L)$ (here we use Theorem 2). We convert $\overline{\mathcal{B}}$ to a $\overset{\Sigma}{\leftrightarrow}$-SEVPA $\mathcal{B}$ recognizing L. To fix the notation suppose that $\overline{\mathcal{B}} = \langle Q^b, \hat{\Lambda}, q_0^b, \Gamma^b, \delta^b, Q_F^b \rangle$ and that it comes with $\mathit{Target}_{\overline{\mathcal{B}}}$ and $\overset{Q}{\leftrightarrow}_{\overline{\mathcal{B}}}$. Automaton $\mathcal{B}$ will be obtained by changing just $\rightarrow_{\overline{\mathcal{B}}}$ to $\rightarrow_{\mathcal{B}}$ as follows:

– $q \xrightarrow{c/\gamma}_{\mathcal{B}} q'$ if $q \xrightarrow{c}_{\overline{\mathcal{B}}} q' \xrightarrow{t/\gamma}_{\overline{\mathcal{B}}} q''$, where $t \in T$ is defined by $t = \mathit{Target}(c)$;
– $\rightarrow_{\mathcal{B}}$ is the same as $\rightarrow_{\overline{\mathcal{B}}}$ on Σ_{int} and Σ_{ret}.

By definition $\mathcal{B}$ is a $\overset{\Sigma}{\leftrightarrow}$-SEVPA when it comes with $\mathit{Target}_{\mathcal{B}}(c) = \mathit{Target}_{\overline{\mathcal{B}}}(\mathit{Target}(c))$ and $\overset{Q}{\leftrightarrow}_{\mathcal{B}}$. Here, transitions that had been split during the translation have been collapsed back. Then the language recognized by $\mathcal{B}$ is obviously L. As in the proof of Theorem 2, both $\overline{\mathcal{B}}$ and $\mathcal{B}$ are the same no matter what $\mathcal{A}$ we start with, so the size of $\mathcal{B}$ is bounded from above by $|\Sigma_{call}| \times n$, where n is the minimal size of a $\overset{\Sigma}{\leftrightarrow}$-SEVPA recognizing L. We are looking for an exact minimization, so we suppose every inaccessible states of $\mathcal{B}$ have been removed in the end of the back translation. Now, same considerations as in the proof Theorem 3 apply, and $\mathcal{B}$ is the unique (up to isomorphism) minimal MEVPA recognizing L.

Again, the minimization of $\overline{\mathcal{A}}$ into $\overline{\mathcal{B}}$ and the accessibility test can be done respectively in $\mathcal{O}(|\overline{\mathcal{A}}|^3)$ time and $\mathcal{O}(|\overline{\mathcal{B}}|^3)$. Thus both in $\mathcal{O}(|\mathcal{A}|^3)$. Other steps of this algorithm can be done in linear time. So it is possible to construct $\overline{\mathcal{B}}$ in $\mathcal{O}(|\mathcal{A}|^3)$ time, where $\mathcal{A}$ is any CDA recognizing L.

5 Block Visibly Pushdown Automata

In the previous section we have observed that there is an exponential lower bound for the translation from VPA to equivalent CDA. This motivates the quest for a new subclass of VPA which admits good properties with respect to minimization. Here we introduce the class of block visibly pushdown automata (BVPA). One important property is that for every VPA there is an equivalent BVPA of quadratic size. The idea of BVPA is that its states are divided in equivalence classes of some $\overset{Q}{\leftrightarrow}$ and only call transitions can change the classes. But this time a call does not determine a class to which the automaton has to go.

Definition 7 (Block Visibly Pushdown Automaton).
A VPA $\mathcal{A} = (Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F)$ is a Block Visibly Pushdown Automaton (BVPA) if there is a function Target : $\Sigma_{call} \times Q \mapsto Q$ and an equivalence relation $\overset{Q}{\leftrightarrow}$ on Q such that:

- *for all $(q, c) \in Q \times \Sigma_{call}$, $q_0 \overset{Q}{\nleftrightarrow} Target(q, c)$*
- *if $q \xrightarrow{i} q'$ then $q \overset{Q}{\leftrightarrow} q'$,*
- *if $q \xrightarrow{c/(q,c)} q'$ then $q' = Target(q, c)$,*
- *if $q \xrightarrow{r/(q',c)} q''$ then $q' \overset{Q}{\leftrightarrow} q''$.*
- *$Target(q, c) \overset{Q}{\leftrightarrow} Target(q', c')$ then $Target(q, c) = Target(q', c')$*

Theorem 5. *Given a VPA of size n, there is an equivalent BVPA of size $\mathcal{O}(n^2)$, that can be computed in quadratic time.*

Proof. Let $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F \rangle$ be a VPA. The idea of the construction is to make, for each $q \in Q$, a copy of $\mathcal{A}$ simulating the action of $\mathcal{A}$ after a call transition pointing at q. In fact we only need to make a copy of $\mathcal{A}$ for each $q \in Q$ pointed at by a call transition. Let T denote the set of states of $\mathcal{A}$ pointed at by a call transition, and $T_0 = T \cup \{0\}$. We define the BVPA $\mathcal{B} = \langle Q', \hat{\Sigma}, q'_0, \Gamma', \delta', Q'_F \rangle$ by:

- $Q' = \{(q, t) | q \in Q, t \in T_0\}$,
- $\Gamma' = \Gamma \times T_0, \quad q'_0 = (q_0, 0), \quad Q_F = \{(q^0, 0) : q \in Q_F\}$,
- the transitions δ' are defined as follows (here $j \in T_0$)
  - $(p, t) \xrightarrow{i}_{\mathcal{B}} (q, t)$ (where $p \xrightarrow{i}_{\mathcal{A}} q$);
  - $(p, t) \xrightarrow{c/(\gamma,t)}_{\mathcal{B}} (q, q)$ (where $p \xrightarrow{c/\gamma}_{\mathcal{A}} q$);
  - $(p, t) \xrightarrow{r/(\gamma,t')}_{\mathcal{B}} (q, t')$ (where $t' \in T_0$ and $p \xrightarrow{r/\gamma}_{\mathcal{A}} q$).

We define $Target_{\mathcal{B}}((q, t), c) = (t', t')$ where $\delta(q, c) = t'$ and $\overset{Q}{\leftrightarrow}_{\mathcal{B}}$ by: $(p, t) \overset{Q}{\leftrightarrow}_{\mathcal{B}} (q, t')$ iff $t = t'$. Obviously, $\mathcal{B}$ is a BVPA. It is equivalent to $\mathcal{A}$ and is of size in $\mathcal{O}(|\mathcal{A}|^2)$.

Due to Theorem 5, every VPA can be approached by a BVPA of quadratic size. So approximative minimization of BVPA is as difficult as approximative minimization of VPA. We propose here a weaker minimization in the same sense as SEVPA minimization preserving $\overset{\Sigma}{\leftrightarrow}$ relation (cf. Section 4.2). Here again, our minimization will keep the structure of call transitions fixed. The automata studied in this section are not call driven any more. So we need a new way to characterize the structure of call transitions. Let $\mathcal{A}$ and $\mathcal{B}$ be two BVPA recognizing the same language L. $\mathcal{A}$ and $\mathcal{B}$ will have the same structure if when reading $u \in L$, the two automata pass through the "same" blocks simultaneously.

We use $Pref(L)$ for the set of prefixes of a language L. Take a BVPA recognizing L: $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F \rangle$ together with $Target$ and $\overset{Q}{\leftrightarrow}$. Let T be the range of $Target$. The *associated partition* is the partition of $K_L = \{uc | u \in MR(\hat{\Sigma}), c \in \Sigma_{call}, uc \in Pref(L)\}$ defined by:

$$\text{for every } t \in T,\ K_t = \{uc | u \in MR(\hat{\Sigma}), c \in \Sigma_{call}, \delta_0(uc) = t, uc \in Pref(L)\}$$

Theorem 6. *Given a consistent BVPA, in a cubic time it is possible to find the unique (up to isomorphism) minimal equivalent BVPA of same associated partition.*

Remark 7. A $\overset{\Sigma}{\leftrightarrow}$-SEVPA is a BVPA whose associated partition $(K_t)_{t \in T}$ verifies: uc and $u'c'$ are in the same K_t iff $c \overset{\Sigma}{\leftrightarrow} c'$. So the previous theorem gives a minimization of $\overset{\Sigma}{\leftrightarrow}$-SEVPA as a special case.

Proof. The proof generalizes that for SEVPA (cf. **??**), but this time the function Φ will not be a homomorphism. We fix a BVPA $\mathcal{A} = \langle Q, \hat{\Sigma}, q_0, \Gamma, \delta, Q_F \rangle$ together with $Target$ and $\overset{Q}{\leftrightarrow}$ recognizing $L \subseteq WM(\hat{\Sigma})$. Let $T \subseteq Q$ be the range of $Target$. We define the visibly pushdown alphabet $\hat{\Lambda}$ by:

$$\Lambda_{int} = \Sigma_{int} \cup \Sigma_{call}, \quad \Lambda_{call} = T, \quad \Lambda_{ret} = \Sigma_{ret}$$

(where T is supposed to be disjoint from Σ). We take an application Φ that adds to a word m, after each occurence of a call symbol $c \in \Sigma_{call}$, the state reached by $\mathcal{A}$ during the execution of m when reading this c. Formally:

$$\Phi : \begin{cases} \varepsilon \mapsto \varepsilon \\ ua \mapsto \Phi(u)a \text{ if } a \in \Sigma_{ret} \cup \Sigma_{int} \\ ua \mapsto \Phi(u)a\delta(ua) \text{ if } a \in \Sigma_{call} \end{cases}$$

It is easy to verify that $\Phi(L)$ is a $WM(\hat{\Lambda})$ language.

We now construct an eCDA $\overline{\mathcal{A}}$ over $\hat{\Lambda}$ recognizing $\Phi(L)$. When reading c we need to remember this letter in the state, so for every $q \in Q$ we will encode each call $c \in \Sigma_{call}$ in a new state (q, c) and also add an associated sink $(q, \bot)$. For notational reasons, a state $q \in Q$ will be renamed into $(q, 0)$. The formal construction of $\overline{\mathcal{A}} = \langle Q', \hat{\Lambda}, q'_0, \Gamma', \delta', Q'_F \rangle$ is as follows:

- $Q' = Q \times (\Sigma_{call} \cup \{0, \bot\})$;

– $\Gamma' = Q'$, $q'_0 = (q_0, 0)$ and $Q'_F = \{(q,0) | q \in Q_F\}$;
– the transitions δ' are defined as follows:
  • $(p,0) \xrightarrow{i}_{\overline{\mathcal{A}}} (q,0)$ if $p \xrightarrow{i}_{\mathcal{A}} q$,
  • $(q,0) \xrightarrow{c}_{\overline{\mathcal{A}}} (q,c)$, if $q \in Q$ and $c \in \Sigma_{call}$,
  • $s \xrightarrow{t/s}_{\overline{\mathcal{A}}} (t,0)$ if $s \in Q'$ and $t \in T$,
  • $(q,0) \xrightarrow{r/(p,c)}_{\overline{\mathcal{A}}} (q',0)$ if $q \xrightarrow{r/(p,c)}_{\mathcal{A}} q'$ and $q \overset{Q}{\leftrightarrow} \mathit{Target}(p,c)$,
  • $(q,c) \xrightarrow{a}_{\overline{\mathcal{A}}} (q,\bot)$ if $q \in Q$, $c \in \Sigma_{call} \cup \{\bot\}$ and $a \in \Lambda_{int}$,
  $(q,c) \xrightarrow{r/(q',c')}_{\overline{\mathcal{A}}} (q',\bot)$ if $(q,c),(q',c') \in Q'$ with $c \neq 0$ or $q \overset{Q}{\nleftrightarrow} \mathit{Target}(q',c')$.

It is straightforward to check that $\overline{\mathcal{A}}$ is an eCDA when it comes with $\mathit{Target}_{\overline{\mathcal{A}}}$ and $\overset{Q}{\leftrightarrow}_{\overline{\mathcal{A}}}$ defined by:

– for every $t \in T$, $\mathit{Target}_{\overline{\mathcal{A}}}(t) = (t,0)$;
– for every $(p,c),(q,c') \in Q'$, $(p,c) \overset{Q}{\leftrightarrow}_{\overline{\mathcal{A}}} (q,c')$ iff $p \overset{Q}{\leftrightarrow}_{\mathcal{A}} q$.

It is also easy to prove by induction that for every $v \in WM(\hat{\Lambda})$:

– if there is $u \in MR(\hat{\Sigma})$ such that $v = \Phi(u)$ then $\delta'(v) = (\delta(u), 0)$,
– if there is $u \in MR(\hat{\Sigma})$ and $c \in \Sigma_{call}$ such that $v = \Phi(u)c$ then $\delta'(v) = (\delta(u), c)$,
– otherwise $\delta'(v)$ is a sink.

Hence, due to the definition of Q'_F the language recognized by the eCDA $\overline{\mathcal{A}}$ is $\Phi(L)$. Let $\overline{\mathcal{B}}$ be the minimal eCDA recognizing $\Phi(L)$ (here we use Theorem 2). During this minimization, states of $\overline{\mathcal{A}}$ are merged into states of $\overline{\mathcal{B}}$. We remember the function f that maps a state of $\overline{\mathcal{A}}$ to the state of $\overline{\mathcal{B}}$ into which it is merged. Note that this is exactly the function f defined in the proof of Theorem 2 when discussing unicity, and that f can effectively be computed (cf minimization algorithm in [4]). We convert $\overline{\mathcal{B}}$ to a BVPA $\mathcal{B}$ recognizing L and of same associated partition as $\mathcal{A}$. To fix the notation suppose that $\overline{\mathcal{B}} = \langle Q^b, \hat{\Lambda}, q_0^b, \Gamma^b, \delta^b, Q_F^b \rangle$ and that it comes with $\mathit{Target}_{\overline{\mathcal{B}}}$ and $\overset{Q}{\leftrightarrow}_{\overline{\mathcal{B}}}$. Automaton $\mathcal{B}$ will be obtained by changing just $\rightarrow_{\overline{\mathcal{B}}}$ to $\rightarrow_{\mathcal{B}}$ as follows:

– $\rightarrow_{\mathcal{B}}$ is the same as $\rightarrow_{\overline{\mathcal{B}}}$ on Σ_{int} and Σ_{ret};
– Let s be in Q^b, c be in Σ_{call}. If there is some state q in $\mathcal{A}$ such that $f((q,0)) = s$, then choose such a q. Otherwise, as f is surjective there is some state (q,c') ($c' \in \Sigma_{call} \cup \{0,\bot\}$) of $\overline{\mathcal{A}}$ such that $f((q,c')) = s$. Then choose such a q. Define $t = \mathit{Target}(q,c)$. Let then s' and s'' be the states of $Q_{\overline{\mathcal{B}}}$ such that $s \xrightarrow{c}_{\overline{\mathcal{B}}} s' \xrightarrow{t/\gamma}_{\overline{\mathcal{B}}} s''$. Then:
  $s \xrightarrow{c/\gamma}_{\mathcal{B}} s''$.

Let $\mathit{Target}_{\mathcal{B}}(s,c)$ be the sate s'' defined just above and $\overset{Q}{\leftrightarrow}_{\mathcal{B}}$ be identical to $\overset{Q}{\leftrightarrow}_{\mathcal{B}}$. Then $\mathcal{B}$ is obviously a BVPA. This time, the back translation is more complicated. In order to collapse transitions that had been split during translation, we have to choose a correct call transition in $\overline{\mathcal{B}}$. As $\mathcal{A}$ is not call driven, the call

read does not characterize the target state of a call transition. f allow us to find a state q of $\mathcal{A}$ from which some state s of $\overline{\mathcal{B}}$ comes from, and then deduce a target for s such that $\mathcal{B}$ will have the same associated partition as $\mathcal{A}$. Note that the states q in the definition of $\rightarrow_{\mathcal{B}}$ are not uniquely determined, but this will not matter anyway.

In the lemma 2 we will prove by induction that:

$$\begin{cases} \forall u \in \mathit{Pref}(L),\ \delta_{\mathcal{B}}(u) = \delta_{\overline{\mathcal{B}}}(\Phi(u)) \\ \forall u \notin \mathit{Pref}(L),\ u \notin L(\mathcal{B}) \end{cases}$$

Now it becomes obvious that the language recognized by $\mathcal{B}$ is L and that $\mathcal{A}$ and $\mathcal{B}$ have the same associated partition. Eventually, the same considerations about f as in the proof of Theorem 2 allow us to prove that $\mathcal{B}$ is the unique minimal consistent BVPA of same partition as $\mathcal{A}$ and recognizing L. As usual, it is possible to construct it in a cubic time.

Lemma 2.

$$\begin{cases} \forall u \in \mathit{Pref}(L),\ \delta_{\mathcal{B}}(u) = \delta_{\overline{\mathcal{B}}}(\Phi(u)) \\ \forall u \notin \mathit{Pref}(L),\ u \notin L(\mathcal{B}) \end{cases}$$

Proof. We proceed by induction on u. Let u be a word in $\mathit{Pref}(L)$ such that $\delta_{\mathcal{B}}(u) = \delta_{\overline{\mathcal{B}}}(\Phi(u))$, and c be in Σ_{call}.

First, suppose uc is in $\mathit{Pref}(L)$. There is a unique $t \in T$ such that $\Phi(uc) = \Phi(u)ct \in \mathit{Pref}(L)$. Let q be the state of $\mathcal{A}$ such that $\delta_{\mathcal{A}}(u) = q$. Then $\delta_{\overline{\mathcal{A}}}(\Phi(u)) = (q, 0)$ and $\delta_{\overline{\mathcal{B}}}(\Phi(u)) = f((q, 0))$. Let $s = f((q, 0))$. If this q has been choosen to define the c transition from s in $\mathcal{B}$, then $\delta_{\mathcal{B}}(uc) = \delta_{\overline{\mathcal{B}}}(\Phi(u)ct) = \delta_{\overline{\mathcal{B}}}(\Phi(uc))$. But if the state choosen was q' such that, $f((q', 0)) = s = f((q, 0))$. Then $(q', 0)$ and $(q, 0)$ have been collapsed during minimization, so for $m \in \Lambda^*$, $w, w' \in WM(\hat{\Lambda})$ such that $\delta_{\overline{\mathcal{A}}}(mw) = (q, 0)$ and $\delta_{\overline{\mathcal{A}}} = (q', 0)$: for every $v \in \Lambda^*$, $mwv \in \Phi(L)$ iff $mw'v \in \Phi(L)$. Hence $\mathit{Target}(q', c)$ has to be the sames as $\mathit{Target}(q, c)$ and we still have: $\delta_{\mathcal{B}}(uc) = \delta_{\overline{\mathcal{B}}}(\Phi(u)ct) = \delta_{\overline{\mathcal{B}}}(\Phi(uc))$.

Now, suppose uc is not in $\mathit{Pref}(L)$. For every $t \in T$, $\Phi(u)ct \notin \mathit{Pref}(\Phi(L))$. So there is no $v \in \Lambda^*$ such that $\delta_{\overline{\mathcal{B}}}(\Phi(uc)v) \in Q_F^b$. Hence, whatever the t choosen in the definition of $\rightarrow_{\mathcal{B}}$ there is no $v \in \Sigma^*$ such that $\delta_{\mathcal{B}}(ucv) \in L$.

We are now able, given a BVPA, to find the minimal equivalent BVPA of same associated partition. The question is how small this minimal BVPA can be. This is a first step in minimization out of the class of CDA. But this is not sufficient, as the example 3 shows that the blow-up between the minimal equivalent BVPA and the minimal equivalent BVPA with respect to a given associated partition can be exponential; which, again, is also an upper bound. Indeed, recall L_k is the VPL $L_k = a_1 c L_{a_1} + \cdots + a_k c L_{a_k} r$ over $\hat{\Sigma} = (\{c\}, \{r\}, \{a_1, \ldots, a_k\})$, where $L_{a_i} \subseteq \{a_1, \ldots, a_n\}^*$ is the set of words where the number of a_i is even. Here K_{L_k} is $\{a_1 c, \ldots, a_k c\}$. The minimal BVPA of associated partition the trivial one (K_{L_k}) is again of size bigger than 2^k, and the example of figure 3 gives a BVPA of associated partition $(\{a_1 c\}, \ldots, \{a_k c\})$ recognizing L_k and of size in $\mathcal{O}(k^2)$.

So the choice of the associated partition can induce an exponential blow-up. Nevertheless, in the case of SEVPA, even with an optimal choice of the equivalence relation $\overset{\Sigma}{\leftrightarrow}$ over Σ_{call} the minimal automata $\overset{\Sigma}{\leftrightarrow}$-SEVPA constructed can be exponentially bigger than an equivalent VPA, although the choice of an optimal associated partition gives a BVPA in quadratic size of a minimal VPA.

6 Conclusions

Our results indicate that the class eCDA is the best if the call alphabet is small. The class CDA is more interesting than SEVPA because it includes SEVPA and moreover it permits a general minimization result without requiring to preserve additional structure. Class MEVPA is still interesting, but: for getting canonical machines eCDA is simpler, and for getting the smallest machines possible CDA is better because it is strictly larger. The problem with all these classes is that VPA are exponentially more succinct than CDA.

The above results can be compared with the situation for regular languages. Deterministic automata are not the most succinct way of representing regular languages. Nondeterministic automata or even two-pass deterministic automata (which reads the input first from left to right and then from right to left) are exponentially more succinct. Still, as no minimization, even approximative, is known for other classes, deterministic automata are widely used.

The class of BVPA seems quite promising. Our results show that minimization of VPA can be understood as fixing a partition of calls. At present we do not know how to calculate a good partition. Still in some contexts it may be easy to guess a partition that gives good results. Example 3 shows such a partition that is simple but takes us outside CDA.

References

[1] R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In "TACAS'04", volume 2988, pages 467–481 (2004).

[2] R. Alur and P. Madhusudan. Visibly pushdown languages (2004).

[3] Rajeev Alur, Viraj Kumar, P. Madhusudan, and Mahesh Viswanathan. Congruences for visibly pushdown languages. In "ICALP", pages 1102–1114 (2005).

[4] Rajeev Alur, Viraj Kumar, P. Madhusudan, and Mahesh Viswanathan. Congruences for visibly pushdown languages. (UIUCDCS-R-2005-2565) (2005).

[5] J. Berstel and L. Boasson. Balanced grammars and their languages. In "Formal and Natural Computing: Essays Dedicated to Grzegorz Rozenberg", volume 2300 of "LNCS", pages 3–25 (2004).

[6] David Dill. Timing assumptions and verification of finite-state concurrent systems. In "Wokshop on Automatic VerificationMethods for Finite State Systems", volume 407 of "LNCS", pages 197–212 (1989).

[7] Viraj Kumar, P. Madhusdan, and Mahesh Viswanathan. Visibly pushdown automata for streaming xml. In "Intl World Wide Web Conference (WWW)" (2007).

[8] Viraj Kumar, P. Madhusudan, and Mahesh Viswanathan. Minimization, learning, and conformance testing of boolean programs. In "CONCUR", pages 203–217 (2006).

[9] C. Loding, P. Madhusudan, and O. Serre. Visibly pushdown games. In "FSTTCS 04", LNCS (2004).

[10] R. McNaughton. Parenthesis grammars. *Journal of the ACM* **14**(3), 490–6500 (1967).

[11] Kurt Mehlhorn. Pebbling mountain ranges and its application of dcfl-recognition. In Jacobus W. Bakker de and Jan van Leeuwen, editors, "Automata, languages and programming (ICALP-80) : 7th annual international colloquium", volume 85 of "Lecture Notes in Computer Science", pages 422–435, Noordwijkerhout, The Netherlands (1980). Springer.

[12] Jean-Francis Michon and Jean-Marc Champarnaud. Automata and binary decision diagrams. In "WIA '98: Revised Papers from the Third International Workshop on Automata Implementation", pages 178–182, London, UK (1999). Springer-Verlag.

[13] Andrzej Murawski and Igor Walukiewicz. Third-order idealized algol with iteration is decidable. In "FOSSACS'05", volume 3441 of "LNCS", pages 202–218 (2005).

[14] C. Pitcher. Visibly pushdown expression effects for xml streem processing. In "Programming Language Techonologies for XML", pages 1–14 (2005).