# Antichains for Visibly Pushdown Automata

Mizuhito Ogawa, Nguyen Van Tang, and Nao Hirokawa

School of Information Science
Japan Advanced Institute of Science and Technology, Japan
{mizuhito,tang_nguyen,hirokawa}@jaist.ac.jp

**Abstract.** This paper presents an antichain algorithm for universality / inclusion checking of visibly pushdown automata (VPAs). Our idea is, combining determinization step of a VPA with its $\mathcal{P}$-automaton construction in an incremental way. A $\mathcal{P}$-automaton accepts the set of reachable configurations, and instead of antichains of subsets of configurations, we focus on antichains of transitions of a $\mathcal{P}$-automaton to minimize each step of the procedure. Preliminary experiments on randomly generated VPAs show significant improvement compared to existing techniques.

## 1 Introduction

When extending model checking with context-sensitive specification, a promising choice is a Visibly Pushdown Automaton (VPA) [2, 1]. A VPA is a proper subclass of deterministic pushdown automata, and a Dyck language (generalized parenthesis language) is an instance. A notable feature is that the class of VPAs is closed under the boolean operations, which reduces VPA model checking to the language inclusion problem $L(M) \subseteq L(S)$. Its typical solution is, (1) the complementation $L(S)^c$, (2) the intersection of $L(M) \cap L(S)^c$, and (3) the emptiness checking $L(M) \cap L(S)^c = \varnothing$. Among them, the hardest is the complementation, which is divided into the subset construction (determinization) and the alternation of final states (complementation). However, the determinization step really causes an exponential explosion [7], and worse for a VPA; it costs $O(2^{n^2})$. A direct implementation (e.g., VPAlib[1]) is stuck for few control states.

A recent solution for an automata-theoretic model checking is the *antichain* algorithm, e.g., for finite automata [8], finite tree automata [3], and Buchi automata [4]. Its idea is combining a minimized determinization, which projects frontiers of determinized transition sequences to minimal incomparable subsets (antichain), and the emptiness checking in an incremental way. Although it does not improve the complexity in theory, significant in practice. For instance, there had been virtually no implementations for complementing a Buchi automaton (which is $O(2^{n \log n})$), but its antichain algorithm [4] gives a working tool ALASKA [9]. Unfortunately, this technique cannot be directly applied to VPAs, since a VPA has infinitely many configurations.

[1] http://www.emn.fr/x-info/hnguyen/vpa/

This paper presents an antichain algorithm for universality / inclusion checking of VPAs. Our idea is, combining determinization step of a VPA with its $\mathcal{P}$-automaton construction in an incremental way. A $\mathcal{P}$-automaton accepts the set of reachable configurations [5, 6], and instead of antichains of subsets of configurations, we focus on antichains of transitions of a $\mathcal{P}$-automaton to minimize each step of the procedure. Determinization is regarded as a procedure to remove nested quantifiers of a target property (e.g., universality is "*for each word, there exists an accepting run*"), and universality / inclusion checking is reduced to judgment of $AG\psi$ (in CTL) for a simple valuation $\psi$ of a pushdown transition system (PDS). Preliminary experiments on randomly generated VPAs show significant improvement compared to existing techniques.

The remaining part of the paper is organized as follows: Section 2 shows an antichain algorithm (in terms of $\mathcal{P}$-automata) to detect $AG\psi$ for a PDS. Section 3 presents antichain algorithms for universality / inclusion checking of VPAs. Section 4 reports preliminary experiments and Section 5 concludes the paper.

## 2 Antichain algorithm for pushdown transition systems

### 2.1 Pushdown transition systems

**Definition 1.** *A* pushdown transition system (PDS) $\mathcal{P}$ *is a tuple* $(Q, \varDelta, \varGamma, s_0)$, *where*

- $Q$ *is a finite set of (control) states,*
- $\varGamma$ *is a finite set of stack alphabet with the special symbol* $\perp$,
- $s_0 \in Q$ *is the initial state (and* $(s_0, \perp)$ *is the initial configuration), and*
- $\varDelta$ *is the set of transitions rules either of*

$$\begin{cases} (q,\gamma) \to (q', \gamma_1\gamma_2) & \textit{(push)} \\ (q,\gamma) \to (q', \epsilon) & \textit{(pop)} \quad \textit{if } \gamma \neq \perp \\ (q,\gamma) \to (q', \gamma') & \textit{(internal)} \end{cases}$$

*We write* $St_\varGamma$ *for the set of* stacks $\{w\perp \mid w \in (\varGamma \setminus \{\perp\})^*\}$. *Elements in* $Q \times \varGamma$ *are called* mode, *and ones in* $Q \times St_\varGamma$ *are* configuration. *The transition* $\to_\mathcal{P}$ *on configurations is defined as:* $(q, \gamma w) \to_\mathcal{P} (q', ww')$ *if* $(q,\gamma) \to (q', w) \in \varDelta$. *Finally, the set of all reachable configurations* $\{(q,w) \mid \exists u \in \varSigma^*.\ (s_0, \perp) \xrightarrow{u}_\mathcal{P} (q,w)\}$ *is denoted by* $LC(\mathcal{P})$.

**Definition 2.** *Let* $\mathcal{P} = (Q, \varDelta, \varGamma, s_0)$ *be a PDS. A* valuation $\psi$ *is a predicate on* $Q \times St_\varGamma$. $\psi$ *is called* simple *if* $\psi$ *is determined by mode, namely, "*$\psi(q, \gamma w)$ *if and only if* $\psi(q, \gamma w')$*".*

### 2.2 $\mathcal{P}$-automata

**Definition 3.** *For a PDS* $\mathcal{P} = (Q, \varDelta, \varGamma, s_0)$, *a finite automaton* $\mathcal{C} = (P, \varGamma, \nabla, S, F)$ *with*

– *$P$ is a finite set of states,*
– *stack alphabet $\Gamma$ is the set of input alphabet,*
– *$\nabla \subseteq P \times (\Gamma \cup \{\epsilon\}) \times P$ is the set of transitions,*
– *$S \subseteq P$ is the set of (multiple) initial states, and*
– *$F \subseteq P$ is the set of final states,*

*is a $\mathcal{P}$-*automaton *if $S \subseteq P \cap Q$, $F \subseteq P \setminus Q$, and*

$$\nabla \subseteq ((P \setminus F) \times (\Gamma \cup \{\epsilon\} \setminus \{\bot\}) \times (P \setminus S)) \ \cup \ ((P \setminus F) \times \{\bot\} \times F)$$

*We write $q \overset{\gamma}{\rightsquigarrow}_{\nabla} p$ if $q \overset{\epsilon}{\rightarrow}{}^{*}_{\nabla} \cdot \overset{\gamma}{\rightarrow}_{\nabla} \cdot \overset{\epsilon}{\rightarrow}{}^{*}_{\nabla} p$ for $\gamma \in \Gamma$. The set of configurations accepted by $\mathcal{C}$ is denoted by $LC(\mathcal{C})$, i.e., $LC(\mathcal{C}) = \{(s, w) \in S \times St_{\Gamma} \mid \exists q \in F.\ s \rightarrow^{*}_{\nabla} q\}$. The set of modes $\{(q, \gamma) \in Q \times \Gamma \mid \exists p \in P.\ q \overset{\gamma}{\rightsquigarrow}_{\nabla} p\}$ is denoted by $mode(\mathcal{C})$.*

**Definition 4.** *Let $\mathcal{P}$ be a PDS $(Q, \Gamma, \Delta, s_0)$. Let $\mathcal{C}$ and $\mathcal{C}'$ be $\mathcal{P}$-automata. We write $\mathcal{C} \Longrightarrow_{\mathcal{P}} \mathcal{C}'$ if $\mathcal{C}'$ is obtained from $\mathcal{C}$ by one of the following rules:*

$$[\mathsf{r}]\ \frac{(P, \Gamma, \nabla, S, F)}{(P \cup \{q'\}, \Gamma, \nabla \cup \{(q', \epsilon, p)\}, S \cup \{q'\}, F)} \qquad \text{if } (q, \gamma) \rightarrow (q', \epsilon) \in \Delta \text{ and } q \overset{\gamma}{\rightsquigarrow}_{\nabla} p$$

$$[\mathsf{i}]\ \frac{(P, \Gamma, \nabla, S, F)}{(P \cup \{q'\}, \Gamma, \nabla \cup \{(q', \gamma', p)\}, S \cup \{q'\}, F)} \qquad \text{if } (q, \gamma) \rightarrow (q', \gamma') \in \Delta \text{ and } q \overset{\gamma}{\rightsquigarrow}_{\nabla} p$$

$$[\mathsf{c}]\ \frac{(P, \Gamma, \nabla, S, F)}{(P \cup \{q', (\mathsf{p}_{(q',\gamma_1)})\}, \Gamma, \nabla', S \cup \{q', \mathsf{p}_{(q',\gamma_1)}\}, F)} \qquad \text{if } (q, \gamma) \rightarrow (q', \gamma_1\gamma_2) \in \Delta \text{ and } q \overset{\gamma}{\rightsquigarrow}_{\nabla} p$$

*where, $\nabla' = \nabla \cup \{(q', \gamma_1, \mathsf{p}_{(q',\gamma_1)}), (\mathsf{p}_{(q',\gamma_1)}, \gamma_2, p)\}$ in* [c]. *The $\mathcal{P}$-automaton*

$$(\{s_0, \mathsf{f}\}, \Gamma, \{(s_0, \bot, \mathsf{f})\}, \{s_0\}, \{\mathsf{f}\})$$

*is called* initial *and denoted by $\mathcal{C}_0$. A $\mathcal{P}$-automaton $\mathcal{C}$ is* $\mathcal{P}$-saturated *if $\mathcal{C} = \mathcal{C}'$ whenever $\mathcal{C} \Longrightarrow_{\mathcal{P}} \mathcal{C}'$. A $\mathcal{P}$-saturated with $\mathcal{C}_0 \Longrightarrow^{*}_{\mathcal{P}} \mathcal{C}$ is uniquely determined, and we denote it by $post^{*}(\mathcal{C}_0)$.*

**Proposition 5 ([5, 6]).** *$LC(\mathcal{P}) = LC(post^{*}(\mathcal{C}_0))$.*

*Example 6.* Let a PDS $\mathcal{P} = (\{s_0, q, q'\}, \Delta, \{\gamma, \gamma'\}, \{s_0\})$ be with

$$\Delta = \left\{ \begin{array}{llll} \text{(i).} & (s_0, \bot) \rightarrow (q, \gamma\bot) & \text{(v).} & (q, \gamma') \rightarrow (q', \gamma') \\ \text{(ii).} & (q, \gamma) \rightarrow (q, \gamma'\gamma) & \text{(vi).} & (q', \gamma') \rightarrow (q', \epsilon) \\ \text{(iii).} & (q, \gamma') \rightarrow (q, \gamma'\gamma') & \text{(vii).} & (q, \gamma) \rightarrow (s_0, \epsilon) \\ \text{(iv).} & (q, \gamma') \rightarrow (q, \epsilon) & \text{(viii).} & (q', \gamma) \rightarrow (s_0, \epsilon) \end{array} \right\}$$

Fig. 1 shows its $\mathcal{P}$-automaton construction. It saturates before the transition rule (viii) is applied. Their modes are: $mode(\mathcal{C}) = \{(s_0, \bot), (q, \gamma), (q, \gamma'), (q', \gamma), (q', \gamma')\}$.
.

Toward an efficient algorithm, we investigate properties of intermediate $\mathcal{P}$-automata.

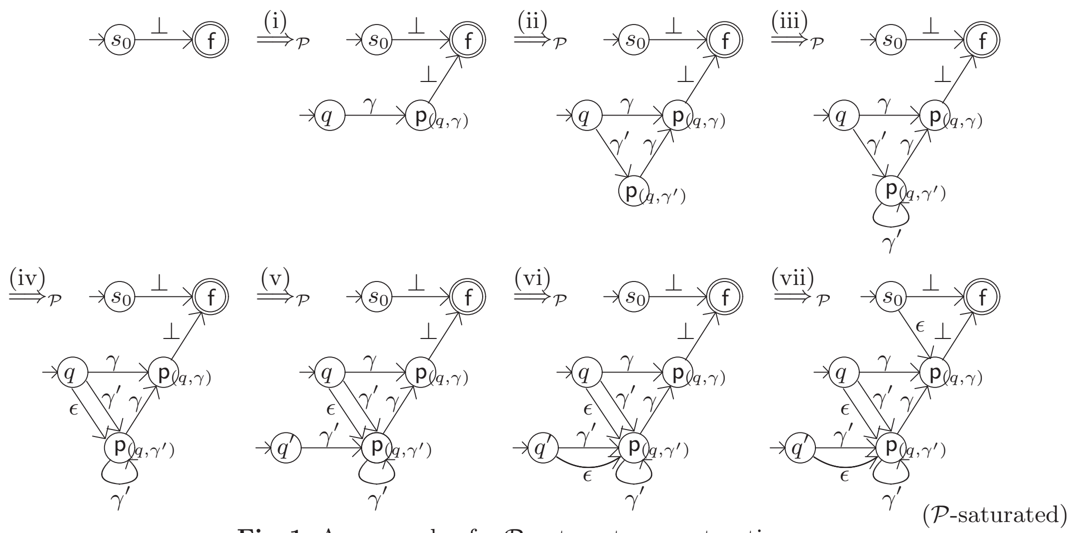


**Fig. 1.** An example of a $\mathcal{P}$-automaton construction

**Lemma 7.** *If* $\mathcal{C} \Longrightarrow_{\mathcal{P}} \mathcal{C}'$ *and* $LC(\mathcal{C}) \subseteq \psi$ *then* $LC(\mathcal{C}') \subseteq \psi$.

**Lemma 8.** $\Longrightarrow_{\mathcal{P}}$ *satisfy the diamond property: If* $\mathcal{C} \Longrightarrow_{\mathcal{P}} \mathcal{C}_1$ *and* $\mathcal{C} \Longrightarrow_{\mathcal{P}} \mathcal{C}_2$ *then* $\mathcal{C}_1 \Longrightarrow_{\mathcal{P}} \mathcal{C}'$ *and* $\mathcal{C}_2 \Longrightarrow_{\mathcal{P}} \mathcal{C}'$. *for some* $\mathcal{C}'$ *such that*

For a simple valuation $\psi$, the *on-the-fly* algorithm to decide $\mathsf{AG}\,\psi$ (i.e., $\psi$ holds for all reachable configurations) is given by Lemma 7, 8, and Proposition 5.

**On-the-fly algorithm to decide $\mathsf{AG}\,\psi$.** Let $\psi$ be a simple valuation of a PDS $\mathcal{P}$ and let $\mathcal{C}$ be a $\mathcal{P}$-automaton. Then, the *on-the-fly* algorithm for $\mathsf{AG}\,\psi$ is to repeat the following steps.

1. When $\mathcal{C}$ is initial: check whether $\psi((s_0, \bot))$ holds.
2. When $\mathcal{C}' \Longrightarrow_{\mathcal{P}} \mathcal{C}$: check whether $\psi((q, \gamma))$ holds for each $(q, \gamma) \in \mathit{mode}(\mathcal{C}) \setminus \mathit{mode}(\mathcal{C}')$.

If violation is found, halt with contradiction. If $\mathcal{C}$ becomes $\mathcal{P}$-saturated, halt in the affirmative.

*Example 9.* We revisit Example 6 with a simple valuation $\psi$ such that $\psi$ holds except for $(q', \gamma)$. The *on-the-fly* algorithm detects that $AG\psi$ fails at $\overset{\text{(vi)}}{\Longrightarrow}_{\mathcal{P}}$.

## 2.3 Antichain algorithm to decide $\mathsf{AG}\,\psi$ for an ordered PDS

**Definition 10.** *An* ordered PDS $\mathcal{P}_{\leq} = (\mathcal{P}, \leq_1, \leq_2)$ *is a PDS* $\mathcal{P} = (Q, \Delta, \Gamma, s_0)$ *associated with orderings* $\leq_1$ *on* $Q$ *and* $\leq_2$ *on* $\Gamma$. *We define the ordering* $\preceq$ *on configurations by* $(q, \gamma_1 \cdots \gamma_n \bot) \preceq (q', \gamma_1' \cdots \gamma_n' \bot)$ *such that* $q \leq_1 q'$ *and* $\gamma_i \leq_2 \gamma_i'$ *for* $1 \leq i \leq n$. *For sets* $C, C'$ *of configurations, we write* $C \preceq_0 C'$ *if* $\forall c' \in C' \exists c \in C. c \preceq c'$. *We denote* $C \sim_0 C'$ *if* $C \preceq_0 C'$ *and* $C \succeq_0 C'$.

*Let $\mathcal{C} = (P, \mathit{\Gamma}, \nabla, S, F)$ be a $\mathcal{P}$-automaton. For $(p_1, \gamma, p_2), (p_1', \gamma', p_2') \in \nabla$, we define $(p_1, \gamma, p_2) \preceq (p_1', \gamma', p_2')$ if*

- *$p_1, p_1' \in Q$ and $p_1 \leq_1 p_1'$, or $p_1 = p_1' \in P \setminus Q$,*
- *$p_2 = p_2'$, and*
- *$\gamma \leq_2 \gamma'$ or $\gamma = \gamma' = \epsilon$.*

**Definition 11.** *A valuation $\psi$ is an* ideal *if $\psi((q, w))$ and $(q, w) \preceq (q', w')$ always imply $\psi((q', w'))$.*

**Lemma 12.** *Assume that $\psi$ is an ideal. If $C \preceq_0 C'$ and $C \subseteq \psi$ then $C' \subseteq \psi$.*

*Proof.* Assume that $\psi(c')$ does not hold for $c' \in C'$. Since $C \preceq_0 C'$, there exists $c \in C$ with $c \preceq c'$. $\psi(c)$ does not hold; otherwise, contradicts that $\psi$ is an ideal.

**Definition 13.** *Let $\mathcal{P}_{\leq}$ be an ordered PDS and let $\mathcal{C}$ and $\mathcal{C}'$ be $\mathcal{P}$-automata. We write $\mathcal{C} \Longrightarrow_{\mathcal{P}_{\leq}} \mathcal{C}'$ if $\mathcal{C} \Longrightarrow_{\mathcal{P}} \mathcal{C}'$ or $\mathcal{C} \Longrightarrow_{[\mathsf{m}]} \mathcal{C}'$, where*

$$[\mathsf{m}]\ \frac{(P, \mathit{\Gamma}, \nabla \uplus \{(p_1, \gamma, p_2)\}, S, F)}{(P, \mathit{\Gamma}, \nabla, S, F)} \quad \text{if } \exists (p_1', \gamma', p_2').\ p_1' \overset{\gamma'}{\rightsquigarrow}_{\nabla} p_2' \wedge (p_1', \gamma', p_2') \preceq (p_1, \gamma, p_2)$$

Since $C \subseteq C'$ implies $C \succeq_0 C'$, the next lemma holds.

**Lemma 14.** *Let $\mathcal{C}, \mathcal{C}'$ be $\mathcal{P}$-automata.*

1. *$LC(\mathcal{C}) \succeq_0 LC(\mathcal{C}')$ whenever $\mathcal{C} \Longrightarrow_{\mathcal{P}_{\leq}} \mathcal{C}'$.*
2. *$LC(\mathcal{C}) \succeq_0 LC(post^*(\mathcal{C}_0))$ whenever $\mathcal{C}_0 \Longrightarrow^*_{\mathcal{P}_{\leq}} \mathcal{C}$.*

Although $\Longrightarrow_{\mathcal{P}}$ satisfies the diamond property, $\Longrightarrow_{\mathcal{P}_{\leq}}$ does not. Instead, we have a simulation relation if $\mathcal{P}_{\leq}$ is downward-complete.

**Definition 15.** *An ordered PDS $\mathcal{P}_{\leq}$ is* downward complete *if the inclusion $\preceq \cdot \rightarrow_{\mathcal{P}} \subseteq \rightarrow_{\mathcal{P}} \cdot \preceq$ holds on configurations.*

**Lemma 16.** *An ordered PDS $\mathcal{P}_{\leq}$ is downward-complete if and only if for every transition rule $(q_1, \gamma) \rightarrow (q_2, w)$ and $(q_1', \gamma')$ with $(q_1', \gamma') \preceq (q_1, \gamma)$, there exists a transition rule $(q_1', \gamma') \rightarrow (q_2', w')$ with $(q_2', w') \preceq (q_2, w)$.*

**Lemma 17.** *Assume that an ordered PDS $\mathcal{P}_{\leq}$ is downward-complete. Let $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_1'$ be $\mathcal{P}$-automata with $\mathcal{C}_1 \Longrightarrow_{\mathcal{P}} \mathcal{C}_2$ and $\mathcal{C}_1 \Longrightarrow^{[\mathsf{m}]} \mathcal{C}_1'$. Then, there exists a $\mathcal{P}$-automaton $\mathcal{C}_2'$ such that $LC(\mathcal{C}_2') \preceq_0 LC(\mathcal{C}_2)$ and $\mathcal{C}_1' \Longrightarrow_{\mathcal{P}} \mathcal{C}_2'$.*

*Proof.* Assume that $\mathcal{C}_1 \Longrightarrow_{\mathcal{P}} \mathcal{C}_2$ is induced by $(q, \gamma) \rightarrow (q', w') \in \delta$ and $\mathcal{C}_1 \Longrightarrow^{[\mathsf{m}]} \mathcal{C}_1'$ removes $(p_1, \gamma_1, p_1')$. If $p_1 \neq q$ or $\gamma_1 \neq \gamma$, apply the same rule to $\mathcal{C}_1'$ such that $\mathcal{C}_1' \Longrightarrow_{\mathcal{P}} \mathcal{C}_2'$. If $p_1 = q$ and $\gamma_1 = \gamma$, let $(p_2, \gamma_2, p_2')$ be a transition in both $\mathcal{C}_1$ and $\mathcal{C}_1'$ such that $(p_2, \gamma_2, p_2') \preceq (p_1, \gamma_1, p_1')$. Then, there exists $(p_2, \gamma_2) \rightarrow (q'', w'') \in \mathit{\Delta}$ by Lemma 16, and it induces $\mathcal{C}_1' \Longrightarrow_{\mathcal{P}} \mathcal{C}_2'$. In either case, $LC(\mathcal{C}_2') \preceq_0 LC(\mathcal{C}_2)$.

**Corollary 18.** *If an ordered PDS $\mathcal{P}_{\leq}$ is downward-complete and $\mathcal{C}_0 \Longrightarrow^*_{\mathcal{P}_{\leq}} \mathcal{C}$ then $LC(\mathcal{C}) \sim_0 LC(post^*(\mathcal{C}_0))$.*

**Definition 19.** *Let $\mathcal{P}_{\leq}$ be an ordered PDS. A $\mathcal{P}$-automaton $\mathcal{C}$ is* minimally $\mathcal{P}$-saturated *if $LC(\mathcal{C}) \sim_0 LC(\mathcal{C}')$ whenever $\mathcal{C} \Longrightarrow_{\mathcal{P},[\mathsf{m}]} \mathcal{C}'$.*

**Theorem 20.** *Let $\mathcal{P}_{\leq}$ be a downward-complete ordered PDS, and suppose $\mathcal{C}_0 \Longrightarrow^*_{\mathcal{P}_{\leq}} \mathcal{C}$. Then, $mode(\mathcal{C}) \subseteq \psi$ if $LC(\mathcal{P}) \subseteq \psi$. Furthermore, the converse holds if $\mathcal{C}$ is minimally $\mathcal{P}$-saturated.*

Corollary 18 shows that, regardless of a strategy, we can reach to $\mathcal{C}$ from the initial $\mathcal{P}$-automaton $\mathcal{C}_0$ such that $LC(\mathcal{C}) \sim_0 LC(post^*(\mathcal{C}_0))$. Since repeated applications may swing between $\Longrightarrow_{\mathcal{P}}$ and $\Longrightarrow_{[\mathsf{m}]}$, we give a specific strategy of $\Longrightarrow_{\mathcal{P},[\mathsf{m}]}$, which applies $\Longrightarrow_{[\mathsf{m}]}$ as early as possible.

**Definition 21.** *Let $\mathcal{P}_{\leq}$ be an ordered PDS and let $\mathcal{C}$ and $\mathcal{C}'$ be $\mathcal{P}$-automata. We define $\Longrightarrow_{\mathcal{P},[\mathsf{m}]}$ by $\Longrightarrow_{\mathcal{P}} \cdot \Longrightarrow^!_{[\mathsf{m}]}$. Here $\Longrightarrow^!_{[\mathsf{m}]}$ indicates repeated applications of $\Longrightarrow_{[\mathsf{m}]}$ until no more $\Longrightarrow_{[\mathsf{m}]}$ can be applied.*

Since $\nabla$ monotonically increases, $\Longrightarrow_{\mathcal{P},[\mathsf{m}]}$ will saturate in finitely many steps.

*Example 22.* We revisit Example 6 with an ordering $q <_1 q'$ (adding to the identity). It is downward-complete and a construction of a $\mathcal{P}$-automaton by $\Longrightarrow_{\mathcal{P},[\mathsf{m}]}$ is the same as in Figure 1, except that transitions from $q'$ are deleted.

The next theorem (immediate from Lemma 12, 14, and Corollary 18) states $\mathsf{AG}\,\psi$ can be incrementally verified.

**Antichain algorithm to decide $\mathsf{AG}\,\psi$.** Let $\psi$ be a simple valuation of an ordered PDS $\mathcal{P}_{\leq}$ and let $\mathcal{C}$ be a $\mathcal{P}$-automaton. Then, the *antichain* algorithm for $\mathsf{AG}\,\psi$ is to repeat the following steps.

1. When $\mathcal{C}$ is initial: check whether $\psi((s_0, \bot))$ holds.
2. When $\mathcal{C}' \Longrightarrow^*_{\mathcal{P},[\mathsf{m}]} \mathcal{C}$: check whether $\psi((q,\gamma))$ holds for each $(q,\gamma) \in mode(\mathcal{C}) \setminus mode(\mathcal{C}')$.

If violation is found, halt with contradiction. If $\mathcal{C}$ becomes minimally $\mathcal{P}$-saturated, halt in the affirmative.

## 3 Visibly Pushdown Automata

### 3.1 Visibly Pushdown Automata and determinization

Before tackling the universality problem of VPAs, we briefly recall definitions for VPAs and determinization from [2, 1].

Let $\Sigma$ be the finite input alphabet, and let $\Sigma = \Sigma_c \uplus \Sigma_r \uplus \Sigma_i$ be a partition of $\Sigma$. Elements in $\Sigma_c$, $\Sigma_r$, or $\Sigma_i$ are called *call* (push) symbols, *return* (pop) symbols, or *internal* symbols, respectively.

**Definition 23.** *A* visibly pushdown automaton *(VPA) $M$ over $\Sigma$ is a tuple $(Q, S, \Gamma, \Delta, F)$ where $Q$ is a finite set of states, $S \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states, $\Gamma$ is a finite stack alphabet with a special symbol $\bot$ (so-called bottom of stack), and*

$$\Delta \subseteq (Q \times \Sigma_c \times Q \times (\Gamma \setminus \{\bot\})) \cup (Q \times \Sigma_i \times Q) \cup (Q \times \Sigma_r \times \Gamma \times Q)$$

*is a set of transitions. We denote these transitions as:*

- $q \xrightarrow{a/+\gamma} q'$ *if* $a \in \Sigma_c$ *and* $(q, a, q', \gamma) \in \Delta$,
- $q \xrightarrow{a/-\gamma} q'$ *if* $a \in \Sigma_r$ *and* $(q, \gamma, a, q') \in \Delta$, *and*
- $q \xrightarrow{a} q'$ *if* $a \in \Sigma_i$ *and* $(q, a, q') \in \Delta$.

*We write $(q, w) \xrightarrow{a}_M (q', w')$ if (a) $w' = w$ and $q \xrightarrow{a} q'$, (b) $w' = \gamma w$, $q \xrightarrow{a/+\gamma} q'$, or (c) $q \xrightarrow{a/-\gamma} q'$ and either $w = \gamma w'$ or $w = w' = \bot$. Let $u = a_1 \cdots a_n$. Whenever $(q_0, w_0) \xrightarrow{a_1}_M \cdots \xrightarrow{a_n}_M (q_n, w_n)$ holds, we write $(q_0, w_0) \xrightarrow{u}_M (q_n, w_n)$. The acceptance language $L(M)$ of $M$ is given by the next set:*

$$\{u \in \Sigma^* \mid \exists s \in S, q \in F, w \in St_\Gamma. \quad (s, \bot) \xrightarrow{u}_M (q, w)\}$$

*A VPA $M$ is said to be* deterministic *if $|S| \leq 1$ and for every $u \in \Sigma^*$ there is at most one $(q, w)$ with $(s, \bot) \xrightarrow{u}_M (q, w)$ and $s \in S$. By forgetting input symbols in a deterministic VPA $M$, the corresponding PDS is induced. We denote it by $\mathcal{P}^M$.*

**Lemma 24 ([2, 1]).** *For each $u \in \Sigma^*$, there exists the unique decomposition (called* well-matched decomposition*) $u = v_0 c_1 v_1 \cdots c_n v_n$ ($n \geq 0$) such that*

- $c_1, \ldots, c_n \in \Sigma_c$,
- $|x|_{\Sigma_r} \leq |x|_{\Sigma_c}$ *for all prefixes $x$ of each $v_i$ ($1 \leq i \leq n$), and*
- $|y|_{\Sigma_r} \geq |y|_{\Sigma_c}$ *for all suffixes $y$ of each $v_i$ ($0 \leq i \leq n$).*

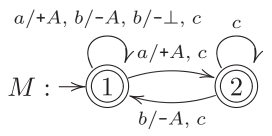
*Here $|w|_X$ denotes the number of $X$-symbol occurrences in $w$.*

**Definition 25.** *Let $M = (Q, S, \Gamma, \Delta, F)$ be a VPA, $u \in \Sigma^*$ and $v_0 c_1 v_1 \cdots c_n v_n$ its well-matched decomposition. We define $[u]_M$ as $(R, \mathtt{void})$ if $n = 0$, $(R, S)$ otherwise. Here* `void` *is a fresh symbol, and $R$ and $S$ are given by the sets:*

$$R = \{q \in Q \mid \exists s \in S, w \in St_\Gamma.\ (s, \bot) \xrightarrow{u}_M (q, w)\}$$
$$S = \{(q, q') \in Q \times Q \mid \exists w, w' \in St_\Gamma.\ (q, w) \xrightarrow{v_n}_M (q', w')\}$$

*The set $\{(R, S) \in X \mid R \cap F \neq \varnothing\}$ is denoted by $\mathsf{F}_M(X)$.*

*Example 26.* Consider the following VPA $M$. We list several values of $[\cdot]_M$.



$$[\epsilon] = [b] = [ab] = (\{1\}, \mathtt{void})$$
$$[a] = [aa] = (\{1, 2\}, \{(1, 1), (2, 2)\})$$
$$[c] = (\{1, 2\}, \mathtt{void})$$
$$[ac] = (\{1, 2\}, \{(1, 1), (1, 2), (2, 1), (2, 2)\})$$

In the following definition we formalize their determinization by inference rules.

**Definition 27.** *Let $M$ be a VPA. The* determinization step $\Longrightarrow_M$ *on VPAs is defined as follows:*

$$(Q, S, \Gamma, \Delta, F) \Longrightarrow_M (Q \cup \{rhs(\delta)\}, S, \Gamma', \Delta \cup \{\delta\}, F \cup \{\mathsf{F}_M(\{rhs(\delta)\})\})$$

*holds if $[u]_M \in Q$, $\delta$ and $\Gamma'$ are defined as follows:*

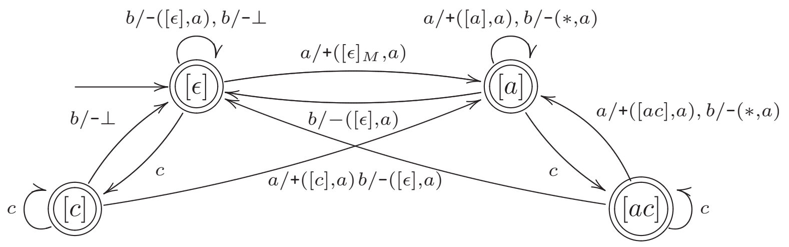
$$\delta = \begin{cases} [u]_M \xrightarrow{a} [ua]_M & \text{if } a \in \Sigma_i \\ [u]_M \xrightarrow{a/+([u]_M,a)} [ua]_M & \text{if } a \in \Sigma_c \\ [u]_M \xrightarrow{a/-\bot} [ua]_M & \text{if } n = 0 \text{ and } a \in \Sigma_r \\ [u]_M \xrightarrow{a/-([u']_M,a')} [u'a'u_n a]_M & \text{if } n > 0,\ a \in \Sigma_r, \text{ and } ([u'], a') \in \Gamma \end{cases} \qquad \Gamma' = \begin{cases} \Gamma \cup \{([u]_M, a)\} & \text{if } a \in \Sigma_c \\ \Gamma & \text{otherwise} \end{cases}$$

*where $u_0 c_1 u_1 \cdots c_n u_n$ is the well-matched decomposition of $u$. The initial VPA $(Q, Q, \{\bot\}, \varnothing, \mathsf{F}_M(Q))$ with $Q = \{[\epsilon]_M\}$ is denoted by $N_0$. A VPA $N$ is $M$-*saturated *if $N = N'$ whenever $N \Longrightarrow_M N'$. The VPA $M$-saturated from $N_0$ uniquely exists. We denote it by $M^d$.*

**Proposition 28** **([2]).** *Let $M$ be an $n$-state VPA and $N$ be the initial VPA for $M$. If $N \Longrightarrow_M^* N'$ and $N'$ is $M$-saturated then $N'$ is completely deterministic with $L(M) = L(N')$.*

Note that when $M$ has $n$ states, $N^d$ contains at most $O(2^{n^2})$ states and also $O(2^{n^2})$ stack symbols. It is known that these bounds cannot be improved (cf. [2]).

*Example 29 (Continued from Example 26).* After 19 steps from $N_0$, we obtain the next saturated automaton $N$, which consists of 4 states and 19 transitions. (In the figure $*$ means $[a]$, $[ac]$, and $[ca]$).



$b/\text{-}([\epsilon],a),\ b/\text{-}\bot$
$a/\text{+}([a],a),\ b/\text{-}(*,a)$
$a/\text{+}([\epsilon]_M,a)$
$[\epsilon]$
$[a]$
$b/\text{-}\bot$
$b/-([\epsilon],a)$
$a/\text{+}([ac],a),\ b/\text{-}(*,a)$
$c$
$a/\text{+}([c],a)$
$b/\text{-}([\epsilon],a)$
$c$
$c$
$[c]$
$[ac]$
$c$

## 3.2 Deciding universality and inclusion

Determinized VPAs are often very large, and thus constructing such VPAs are infeasible (cf. Section 4).

We are ready to describe how to check universality with the antichain algorithm. How to describe universality in the form of $\mathsf{AG}\,\psi$? Obviously, the definition of $L(M) = \Sigma^*$

$$\forall u \in \Sigma^*. \quad \exists s \in S, q \in F, w \in St_\Gamma. \quad (s, \bot) \xrightarrow{u}_M (q, w)$$

is not form of $\mathsf{AG}\,\psi$. Instead of $M$, we consider the determinized VPA $M^d = (Q^d, S^d, \varGamma^d, \varDelta^d, F^d)$. For every input word $u$ the deterministic VPA $M^d$ yields a unique path $(s, \bot) \xrightarrow{u}_{M^d} (q, \sigma)$. Therefore, the input $u$ is no longer needed. Hence, universality is described as the property of $\mathcal{P}^{M^d}$:

$$\forall q \in Q^d, w \in St_{\varGamma^d}. \quad (s, \bot) \rightarrow^*_{\mathcal{P}^{M^d}} (q, w) \text{ implies } q \in F^d$$

which is equivalent to $\mathcal{P}^{M^d} \models \mathsf{AG}\,\psi^{\mathsf{u}}$. Here $\psi^{\mathsf{u}}$ is defined as $F^d \times St_{\varGamma^d}$.

**Definition 30.** *We define the orderings $\leq_1$ on $Q^d$ and $\leq_2$ on $\varGamma^d$ as follows:*

- *$(R, S) \leq_1 (R', S')$ if $R \subseteq R'$, and $S = S' =$ `void` or $S \subseteq S'$.*
- *$([u]_M, a) \leq_2 ([u']_M, a')$ if $[u]_M \leq_1 [u']_M$ and $a = a'$.*

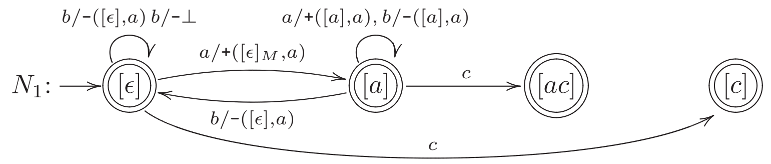
One can verify that $\mathcal{P}^{M^d}_{\leq} = (\mathcal{P}^{M^d}, \leq_1, \leq_2)$ is a downward-complete ordered PDS, and also $\psi^{\mathsf{u}}$ is ideal. Thus, one can apply the antichain algorithm. A crucial point of our formalization is that $M^d$ can be lazily constructed.

**Definition 31.** *Let $M$ be a VPA. We write $(N, \mathcal{C}) \Longrightarrow_M (N', \mathcal{C}')$ if* determinization step *$N \Longrightarrow_M N'$ and $\mathcal{C} = \mathcal{C}'$, or* $\mathcal{P}$-automaton step *$N = N'$ and $\mathcal{C} \Longrightarrow_{\mathcal{P}^N_{\leq}} \mathcal{C}'$ holds. A pair $(N, \mathcal{C})$ is* $M$-saturated *if $\mathcal{C}$ is minimally $\mathcal{P}^{N'}_{\leq}$-saturated whenever $N \Longrightarrow_M N'$.*

**Theorem 32.** *Let $M$ be a VPA and suppose $(N_0, \mathcal{C}_0) \Longrightarrow^*_M (N, \mathcal{C})$.*

- *$L(M) = \varSigma^*$ if $\mathsf{AG}\,\psi^{\mathsf{u}}$ holds for $\mathcal{C}$, and $(N, \mathcal{C})$ is minimally $\mathcal{P}^{N'}_{\leq}$-saturated.*
- *$L(M) \neq \varSigma^*$ if $\mathsf{AG}\,\psi^{\mathsf{u}}$ does not hold for $\mathcal{C}$.*

*Example 33 (Continued from Example 26).* By 8 determinization steps and saturation steps for $\mathcal{C}$, we obtain $(N_0, \mathcal{C}_0) \Longrightarrow^*_M (N, \mathcal{C})$, where $N$ is the following VPA and $\mathcal{C}$ is a minimally $\mathcal{P}^{N_{\leq}}$-saturated automaton.



Then, $(N', \mathcal{C}')$ is also saturated. Since $\mathsf{AG}\,\psi^u$ holds for $\mathcal{C}'$, we can conclude that $M$ is universal.

We can also handle inclusions by the same observation. In the below $M_1 \times M_2$ denotes a VPA constructed by the *product construction* of VPAs $M_1$ and $M_2$ (cf. [2]).

**Definition 34.** *Let $M_1, M_2$ be VPAs. We write $(N, \mathcal{C}) \Longrightarrow_{M_1, M_2} (N', \mathcal{C}')$ if $N \Longrightarrow_{M_2} N'$ and $\mathcal{C} = \mathcal{C}'$, or $N = N'$ and $\mathcal{C} \Longrightarrow_{\mathcal{P}^{M_1 \times N}_{\leq}} \mathcal{C}'$ holds. We define $\psi^{\mathsf{i}}((q_1, q_2), \cdot)$ as $(q_1, q_2) \in (F_1^c \times F_2^d)$. A pair $(N, \mathcal{C})$ is* $(M_1, M_2)$-saturated *if $\mathcal{C}$ is minimally $\mathcal{P}^{M_1 \times N}_{\leq}$-saturated.*

A statement similar to Theorem 32 holds for $\Longrightarrow_{M_1, M_2}$.

## 4 Experimental Results

### 4.1 Random generation of VPA

For inputs of experiments, we prepare sets of randomly generated VPAs: $r$-random VPAs and $r$-regular random VPAs. Throughout experiments, we assume an average nondeterminism $r$, i.e., there are $r$-transitions as an average for each input symbol, control state, and stack symbol (for $\Sigma_r$-symbols only). Thus, the number $N$ of transitions to be generated is $r \times |Q| \times (|\Sigma_c| + |\Sigma_i| + |\Sigma_r| \times (|\Gamma|))$ (following notations in Definition 23).

**Definition 35.** *A set of* $r$-random VPAs *consists of randomly chosen $N$ transitions from* $(Q \times \Sigma_c \times Q \times (\Gamma \setminus \{\bot\})) \cup (Q \times \Sigma_i \times Q) \cup (Q \times \Sigma_r \times \Gamma \times Q)$.

**Definition 36.** *A set of* $r$ regular random VPAs *consists of $N$ transitions given by random choice of $r$-destinations in $Q$ for each input symbol, control state, and stack symbol (for $\Sigma_r$-symbols only).*

A $r$-regular random VPA is complete (i.e., each input word has a run) by definition. Note that a $r$-regular random VPA is *almost always not complete*, similar to the connectivity in random graph theory. During experiments, we fix the sizes such that $|\Sigma_c| = |\Sigma_r| = |\Sigma_i| = 2$, $|\Gamma| = 3$, and $r = 2$. All control states are set to be final, which is the hardest to conquer.

### 4.2 Implementation and experimental results

Our experimental bed is implemented in Java 1.5.0 on Windows XP. The algorithms are classified into STANDARD (sequential execution of determinization and $\mathcal{P}$-automaton construction), ON-THE-FLY, and ANTICHAIN. All tests are performed on a laptop PC equipped with 1.50 GHz Intel® Core$^{\mathrm{TM}}$ Duo Processor L2300 and 1.5 GB of memory.

Although the choice of random generation deviates its characteristic ($r$-random VPA quickly fails the universality, $r$-regular random VPA requires heavy exploration), results generally show that:

- STANDARD is stuck for few control states.
- ON-THE-FLY improves a lot when universality / inclusion are violated.
- ANTICHAIN further improves when universality / inclusion are violated, and works even when universality / inclusion hold.

First, Table 4.2 shows universality checking for 2-random VPAs. Each combination is performed for 50 instances with timeout 60 seconds.

Next, Table 4.2 shows universality checking of 2-regular random VPAs. Although timeout is enlarged to 180 seconds, STANDARD can solve no instances.

Last, Figure 2 shows comparison of the numbers of timeouts for inclusion checking of pairs of 2-regular random VPAs. Note that, since we set all control states to be final, all cases must answer the affirmative. Each combination has 20 instances and timeout is 300 seconds. If not timeout, ANTICHAIN is several times faster than ON-THE-FLY. Although the specification with 5 control states is small, but expressive enough for parenthesis match checking.

| | *number of states* | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ANTICHAIN | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 150 |
| not universal | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 47 |
| total time | 19 | 35 | 43 | 57 | 87 | 144 | 147 | 206 | 222 | 299 | 496 | 336 |
| timeouts (*60* s) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| ON-THE-FLY | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 150 |
| not universal | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 46 | 47 |
| total time | 23 | 46 | 52 | 71 | 110 | 186 | 210 | 274 | 247 | 407 | 686 | 372 |
| timeouts (*60* s) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 3 |
| STANDARD | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 150 |
| not universal | 21 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| total time | 456 | 31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| timeouts (*60* s) | 29 | 49 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |

**Table 1.** Universality checking for 2-random VPAs

| | *number of states* | | | | | |
|---|---|---|---|---|---|---|
| ANTICHAIN | 5 | 10 | 15 | 20 | 30 | 40 |
| universal | 50 | 45 | 19 | 2 | 0 | 0 |
| total time | 52 | 833 | 892 | 86 | 0 | 0 |
| timeouts (*180* s) | 0 | 5 | 31 | 48 | 50 | 50 |
| ON-THE-FLY | 5 | 10 | 15 | 20 | 30 | 40 |
| universal | 50 | 43 | 13 | 0 | 0 | 0 |
| total time | 68 | 1425 | 754 | 0 | 0 | 0 |
| timeouts (*180* s) | 0 | 7 | 37 | 50 | 50 | 50 |

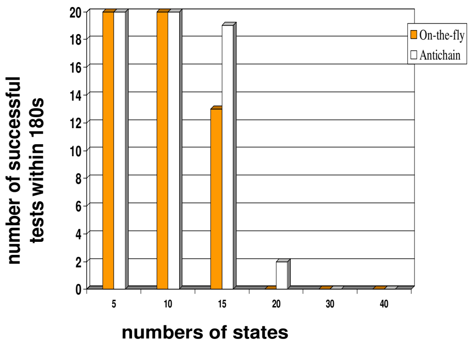


**Table 2.** Universality checking for 2-regular random VPAs

## 5 Conclusions

In this paper, we proposed an antichain algorithm for checking universality / inclusion of VPAs, by combining determinization procedure and a$\mathcal{P}$-automaton construction in an incremental way. We focused on antichains of transitions of a $\mathcal{P}$-automaton to minimize each step of the procedure. Although the current implementation is quite naive (e.g., set operations by a hashset library in Java) and restricts applications of the minimization steps ($\Longrightarrow_{[\mathsf{m}]}$), preliminary experimental results showed significant improvement compared to existing techniques. In the future, we would like to improve implementation by:

– full implementation of the minimization step ($\Longrightarrow_{[\mathsf{m}]}$),
– efficient data structure (e.g.,BDD) for set operations, and
– combining bounded model checking approach with the antichain algorithm.

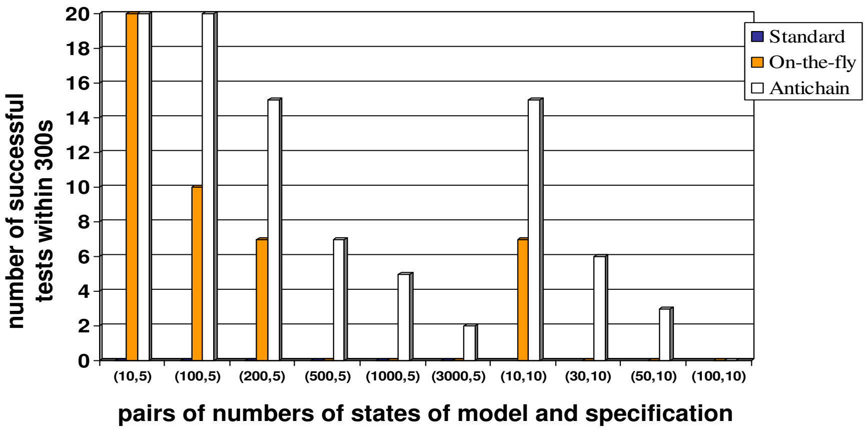


**Fig. 2.** Inclusion checking between pairs of 2-regular random VPAs

We also would like to investigate practical applications, e.g., XML-streaming analysis.

## References

1. Alur, R., Kumar, V., Madhusudan, P., Viswanathan, M.: Congruences for visibly pushdown languages. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1102–1114. Springer (2005)
2. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Babai, L. (ed.) STOC 2004. pp. 202–211. ACM (2004)
3. Bouajjani, A., Habermehl, P., Holík, L., Touili, T., Vojnar, T.: Antichain-based universality and inclusion testing over nondeterministic finite tree automata. In: Ibarra, O.H., Ravikumar, B. (eds.) CIAA 2008. LNCS, vol. 5148, pp. 57–67. Springer (2008)
4. Doyen, L., Raskin, J.F.: Improved algorithms for the automata-based approach to model-checking. In: TACAS. LNCS. pp. 451–465 (2007)
5. Esparza, J., Hansel, D., Rossmanith, P., Schwoon, S.: Efficient algorithms for model checking pushdown systems. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 232–247. Springer (2000)
6. Reps, T.W., Schwoon, S., Jha, S., Melski, D.: Weighted pushdown systems and their application to interprocedural dataflow analysis. Science of Computer Programming 58(1-2), 206–263 (2005)
7. Tabakov, D., Vardi, M.Y.: Experimental evaluation of classical automata constructions. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS, vol. 3835, pp. 396–411. Springer (2005)
8. Wulf, M.D., Doyen, L., Henzinger, T.A., Raskin, J.F.: Antichains: A new algorithm for checking universality of finite automata. In: CAV. LNCS. pp. 17–30 (2006)
9. Wulf, M.D., Doyen, L., Raskin, J.F., Maquet, N.: Alaska: Antichains of logic, automata and symbolic kripke structures analysis. In: ATVA. (2008). to appear