Visibly Pushdown Automata: Universality and Inclusion via Antichains

Marc Ducobu - UMONS

Joint work with : Véronique Bruyère (UMONS), Olivier Gauwin (LaBRI)

LATA13 - 4 April 2013

Outline

Introduction

Words with nesting structure
(XML, traces of programs)

VPAs

Introduction

Words with nesting structure
(XML, traces of programs)

$$\text{VPAs} \xleftarrow[]{\text{(Non-deterministic VPA)}} \text{Logics}$$

Translation

(XPath, CaRet)

Introduction

Words with nesting structure
(XML, traces of programs)

$$\text{VPAs} \xleftarrow[]{\substack{\text{Translation}\\ \text{(Non-deterministic VPA)}}} \text{Logics}$$

(XPath, CaRet)

Testing inclusion of VPAs $\rightarrow$ XML type checking, model checking of programs (EXPTIME-complete)

Introduction

Words with nesting structure
(XML, traces of programs)

$$\text{VPAs} \xleftarrow[\text{(Non-deterministic VPA)}]{\text{Translation}} \text{Logics}$$

(XPath, CaRet)

Testing inclusion of VPAs $\rightarrow$ XML type checking, model checking of programs (EXPTIME-complete)

Related work :

- universality - classical method ($L(\mathcal{A}^c) = \emptyset$ test)
  - OpenNWA (by E. Driscoll, A. Thakur and T. Reps (CAV'12))
- universality - on the fly
  - VPAchecker (by T. Van Nguyen and H. Ohsaki (CONCUR'09))
- universality and inclusion - Ramsey
  - FADecider (by O. Friedmann, F. Klaedtke and M. Lange (preprint 2012))

Our approach : antichains

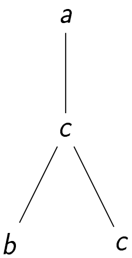
Our approach : antichains

Universality checking with antichain very efficient for

- finite word automata (De Wulf, Doyen, Henzinger and Raskin (CAV'06))
- non-deterministic finite ranked tree automata (Bouajjani, Habermehl, Holik, Touili, and Vojnar (CIAA'08))
- ...

Unranked tree and linearization

Unranked tree labeled by Σ

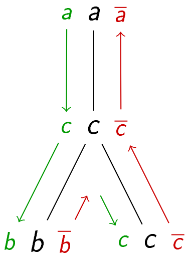


T_Σ : set of all (unranked) trees over Σ

H_Σ : set of all hedges over Σ

Unranked tree and linearization

Unranked tree labeled by Σ



T_Σ : set of all (unranked) trees over Σ

H_Σ : set of all hedges over Σ

Linearization

$a\ c\ b\ \overline{b}\ c\ \overline{c}\ \overline{c}\ \overline{a}$

Well nested word over $\Sigma \cup \overline{\Sigma}$

$[T_\Sigma]$ set of all linearizations of trees over Σ

$[H_\Sigma]$ set of all linearizations of hedges over Σ

Visibly Pushdown Automata (VPA) (on linearizations of unranked trees)

Definition (Alur and Madhusudan, Visibly pushdown languages, 2004)

A *VPA* over alphabet $\Sigma \cup \overline{\Sigma}$ is a tuple $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, \Gamma, Q_i, Q_f, \Delta)$ where Q finite set of states, $Q_i \subseteq Q$ initial states, $Q_f \subseteq Q$ final states, Γ finite set of stack symbols, and Δ finite set of rules of two types :

- $q_1 \xrightarrow{a,\gamma^+} q_2$, for an opening letter $a \in \Sigma$ and $\gamma \in \Gamma$,
- $q_1 \xrightarrow{\overline{a},\gamma^-} q_2$, for a closing letter $\overline{a} \in \Sigma$, $\gamma \in \Gamma$.

Visibly Pushdown Automata (VPA) (on linearizations of unranked trees)

Definition (Alur and Madhusudan, Visibly pushdown languages, 2004)

A *VPA* over alphabet $\Sigma \cup \overline{\Sigma}$ is a tuple $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, \Gamma, Q_i, Q_f, \Delta)$ where Q finite set of states, $Q_i \subseteq Q$ initial states, $Q_f \subseteq Q$ final states, Γ finite set of stack symbols, and Δ finite set of rules of two types :

- $q_1 \xrightarrow{a,\gamma^+} q_2$, for an opening letter $a \in \Sigma$ and $\gamma \in \Gamma$,
- $q_1 \xrightarrow{\overline{a},\gamma^-} q_2$, for a closing letter $\overline{a} \in \Sigma$, $\gamma \in \Gamma$.

A *configuration* : (q, σ) where $q \in Q$ is a state and $\sigma \in \Gamma^*$ is a stack.

Visibly Pushdown Automata (VPA) (on linearizations of unranked trees)

Definition (Alur and Madhusudan, Visibly pushdown languages, 2004)

A *VPA* over alphabet $\Sigma \cup \overline{\Sigma}$ is a tuple $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, \Gamma, Q_i, Q_f, \Delta)$ where Q finite set of states, $Q_i \subseteq Q$ initial states, $Q_f \subseteq Q$ final states, Γ finite set of stack symbols, and Δ finite set of rules of two types :

- $q_1 \xrightarrow{a,\gamma^+} q_2$, for an opening letter $a \in \Sigma$ and $\gamma \in \Gamma$,
- $q_1 \xrightarrow{\overline{a},\gamma^-} q_2$, for a closing letter $\overline{a} \in \Sigma$, $\gamma \in \Gamma$.

A *configuration* : (q, σ) where $q \in Q$ is a state and $\sigma \in \Gamma^*$ is a stack.

A *run* on a tree linearization $a_1 a_2 a_3 \dots a_n \in [T_\Sigma]$:
$(q_i, \epsilon) \xrightarrow{a_1} (q_1, \sigma_1) \xrightarrow{a_2} (q_1, \sigma_2) \xrightarrow{a_3} \dots \xrightarrow{a_n} (q_n, \epsilon)$, with $q_i \in Q_i$.

A run is *accepting* if $q_n \in Q_f$. A tree is *accepted* if there is an accepting run on its linearization.

Outline

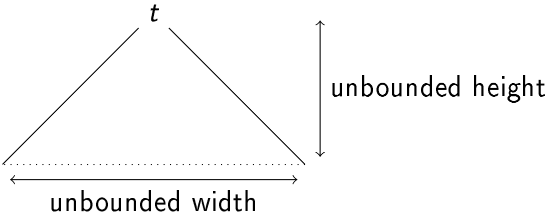
Universality and accessibility relation

A VPA $\mathcal{A}$ is said *universal* if it accepts all unranked trees, i.e. $L(\mathcal{A}) = T_\Sigma$.

Universality and accessibility relation

A VPA $\mathcal{A}$ is said *universal* if it accepts all unranked trees, i.e. $L(\mathcal{A}) = T_\Sigma$.

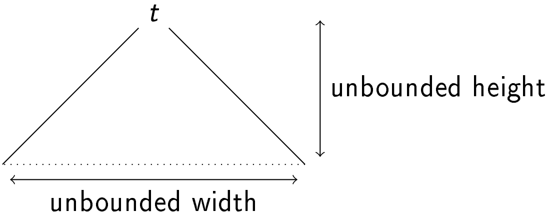
For all $t \in T_\Sigma$, we have to check that $t \in L(\mathcal{A})$.



Universality and accessibility relation

A VPA $\mathcal{A}$ is said *universal* if it accepts all unranked trees, i.e. $L(\mathcal{A}) = T_\Sigma$.

For all $t \in T_\Sigma$, we have to check that $t \in L(\mathcal{A})$.



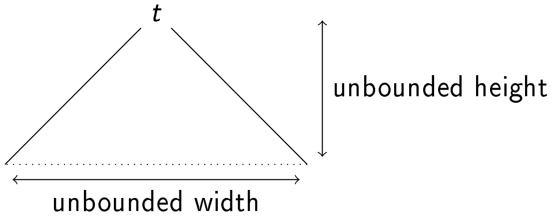
Accessibility relation :

$$Acc(t) = \{(q, q') \in Q \times Q \mid (q, \epsilon) \xrightarrow{[t]} (q', \epsilon)\} \subseteq Q \times Q,\ t \in T_\Sigma$$

Universality and accessibility relation

A VPA $\mathcal{A}$ is said *universal* if it accepts all unranked trees, i.e. $L(\mathcal{A}) = T_\Sigma$.

For all $t \in T_\Sigma$, we have to check that $t \in L(\mathcal{A})$.



Accessibility relation :

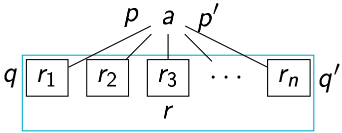
$$Acc(t) = \{(q, q') \in Q \times Q \mid (q, \epsilon) \xrightarrow{[t]} (q', \epsilon)\} \subseteq Q \times Q, \; t \in T_\Sigma$$

Algorithm : compute the set $\{Acc(t) \mid t \in T_\Sigma\} \subseteq \mathcal{P}(Q \times Q)$ and check if $\forall t \in T_\Sigma,\ Acc(t) \cap Q_i \times Q_f \neq \emptyset$

Recursive computation of $\{Acc(t) \mid t \in T_\Sigma\}$

Let $r \in Q \times Q$ be a relation over Q.

$$Post_a(r) = \{(p, p') \in Q \times Q \mid \exists (q, q') \in r, p \xrightarrow{a:\gamma} q \in \Delta,\ q' \xrightarrow{\bar{a}:\gamma} p' \in \Delta\}$$

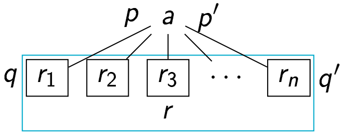


$$\begin{gathered} p \xrightarrow{a:\gamma} q \in \Delta, \\ q' \xrightarrow{\bar{a}:\gamma} p' \in \Delta, \\ (q, q') \in r = r_1 \circ r_2 \circ r_3 \circ \cdots \circ r_n \end{gathered}$$

Recursive computation of $\{Acc(t) \mid t \in T_\Sigma\}$

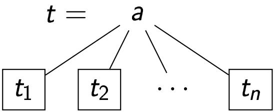
Let $r \in Q \times Q$ be a relation over Q.

$$Post_a(r) = \{(p, p') \in Q \times Q \mid \exists (q, q') \in r, p \xrightarrow{a:\gamma} q \in \Delta,\ q' \xrightarrow{\bar{a}:\gamma} p' \in \Delta\}$$

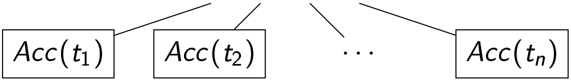


$$p \xrightarrow{a:\gamma} q \in \Delta,$$
$$q' \xrightarrow{\bar{a}:\gamma} p' \in \Delta,$$
$$(q, q') \in r = r_1 \circ r_2 \circ r_3 \circ \cdots \circ r_n$$

$t =$



$$Acc(t) = Post_a(Acc(t_1) \circ \cdots \circ Acc(t_n))$$



Recursive computation of $\{Acc(t) \mid t \in T_\Sigma\}$

Let $\mathscr{R} \subseteq \mathcal{P}(Q \times Q)$ be a set of relations over Q.

$\mathscr{R}^*$ denote the reflexive and transitive closure of $\mathscr{R}$ i.e.
$\{r_1 \circ r_2 \circ \cdots \circ r_n \mid n \geq 0 \text{ and } r_i \in \mathscr{R} \text{ for all } 1 \leq i \leq n\}$

$Post(\mathscr{R}) = \{Post_a(r) \mid a \in \Sigma,\ r \in \mathscr{R}^*\} \cup \mathscr{R}$

$Post^0(\mathscr{R}) = \mathscr{R}$, and for all $i > 0$, $Post^i(\mathscr{R}) = Post(Post^{i-1}(\mathscr{R}))$

$Post^*(\mathscr{R}) = \cup_{i \geq 0} Post^i(\mathscr{R})$

Recursive computation of $\{Acc(t) \mid t \in T_\Sigma\}$

Let $\mathscr{R} \subseteq \mathcal{P}(Q \times Q)$ be a set of relations over Q.

$\mathscr{R}^*$ denote the reflexive and transitive closure of $\mathscr{R}$ i.e.
$\{r_1 \circ r_2 \circ \cdots \circ r_n \mid n \geq 0 \text{ and } r_i \in \mathscr{R} \text{ for all } 1 \leq i \leq n\}$

$Post(\mathscr{R}) = \{Post_a(r) \mid a \in \Sigma,\, r \in \mathscr{R}^*\} \cup \mathscr{R}$

$Post^0(\mathscr{R}) = \mathscr{R}$, and for all $i > 0$, $Post^i(\mathscr{R}) = Post(Post^{i-1}(\mathscr{R}))$

$Post^*(\mathscr{R}) = \cup_{i \geq 0} Post^i(\mathscr{R})$

$$Post^*(\emptyset) = \{Acc(t) \mid t \in T_\Sigma\}$$

$\mathcal{A}$ is universal iff $\forall r \in Post^*(\emptyset),\ r \cap Q_i \times Q_f \neq \emptyset$

The algorithm

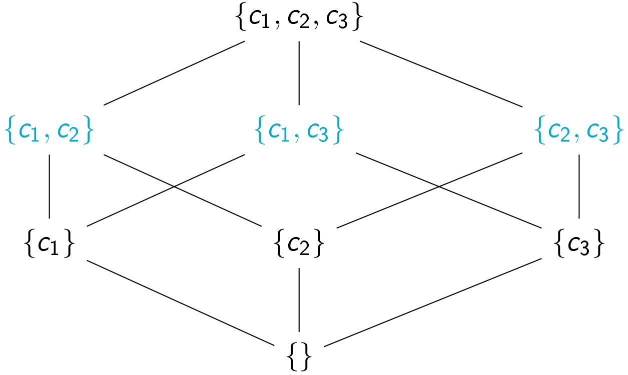
Universality($\mathcal{A}$) :

```
    ℛ ← ∅ ;
    ℛ* ← {id_Q} ;
    repeat
        ℛ_new ← {Post_a(r) | a ∈ Σ, r ∈ ℛ*} \ ℛ ;
        if ∃r ∈ ℛ_new : r ∩ Q_i × Q_f = ∅ then
            return False    // Not universal
        ℛ ← ℛ ∪ ℛ_new
        ℛ' ← ℛ_new \ ℛ*
        if ℛ' ≠ ∅ then
            ℛ* ← (ℛ* ∪ ℛ')*
    until ℛ' = ∅
    return True    // Universal
```

Antichains

Let S be a partially ordered set. An antichain A is a subset of S such that all the elements are pairwise incomparable.

For ex. Let $S = \{e \mid e \subseteq \{c_1, c_2, c_3\}\}$ with $\subseteq$ as order.



$\{\{c_1, c_2\}, \{c_1, c_3\}, \{c_2, c_3\}\}$ is an antichain

Antichain Optimization

We only need to consider minimal elements.

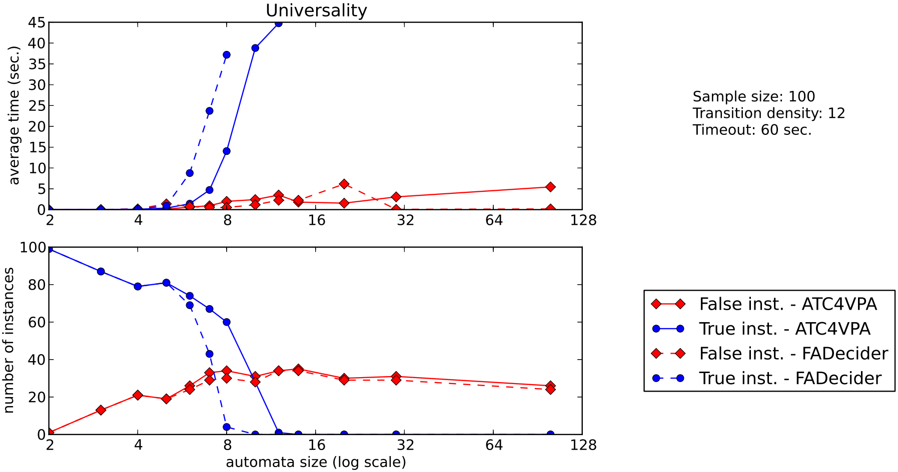
$\mathcal{A}$ is universal iff $\forall r \in \lfloor Post^*(\emptyset) \rfloor$, $r \cap Q_i \times Q_f \neq \emptyset$.

If $\exists r \in \lfloor Post^*(\emptyset) \rfloor : r \cap Q_i \times Q_f = \emptyset \Rightarrow$ VPA not universal.

If $\forall r \in \lfloor Post^*(\emptyset) \rfloor : r \cap Q_i \times Q_f \neq \emptyset \Rightarrow$ VPA universal as
$\forall r' \supseteq r$, $r' \cap Q_i \times Q_f \neq \emptyset$

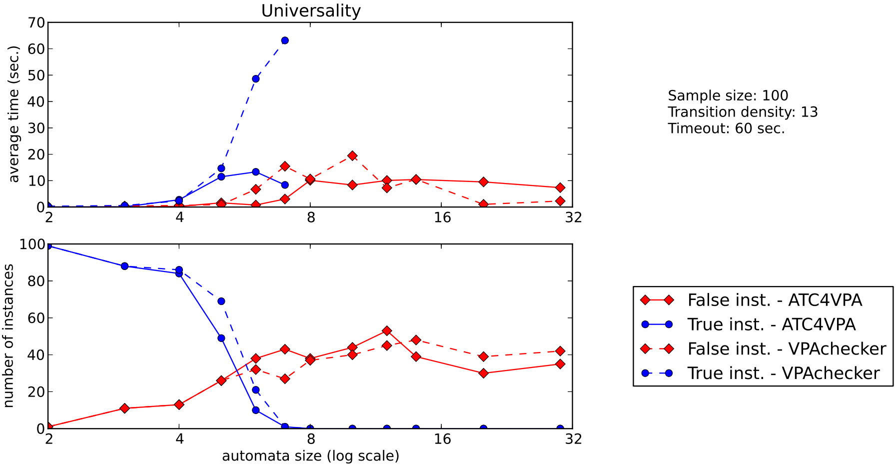
Post operator is monotonic.

Experiments



Sample size: 100
Transition density: 12
Timeout: 60 sec.

Experiments



Sample size: 100
Transition density: 13
Timeout: 60 sec.

Outline

Inclusion

$$\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, \Gamma_{\mathcal{A}}, Q_{i,\mathcal{A}}, Q_{f,\mathcal{A}}, \Delta_{\mathcal{A}}) \text{ and } \mathcal{B} = (Q_{\mathcal{B}}, \Sigma, \Gamma_{\mathcal{B}}, Q_{i,\mathcal{B}}, Q_{f,\mathcal{B}}, \Delta_{\mathcal{B}})$$

Question : $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$

Inclusion

$$\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, \Gamma_{\mathcal{A}}, Q_{i,\mathcal{A}}, Q_{f,\mathcal{A}}, \Delta_{\mathcal{A}}) \text{ and } \mathcal{B} = (Q_{\mathcal{B}}, \Sigma, \Gamma_{\mathcal{B}}, Q_{i,\mathcal{B}}, Q_{f,\mathcal{B}}, \Delta_{\mathcal{B}})$$

Question : $L(\mathcal{A}) \nsubseteq L(\mathcal{B})$

Our goal : find $t \in T_{\Sigma}$: $t \in L(\mathcal{A})$ and $t \notin L(\mathcal{B})$

Inclusion

$$\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, \Gamma_{\mathcal{A}}, Q_{i,\mathcal{A}}, Q_{f,\mathcal{A}}, \Delta_{\mathcal{A}}) \text{ and } \mathcal{B} = (Q_{\mathcal{B}}, \Sigma, \Gamma_{\mathcal{B}}, Q_{i,\mathcal{B}}, Q_{f,\mathcal{B}}, \Delta_{\mathcal{B}})$$

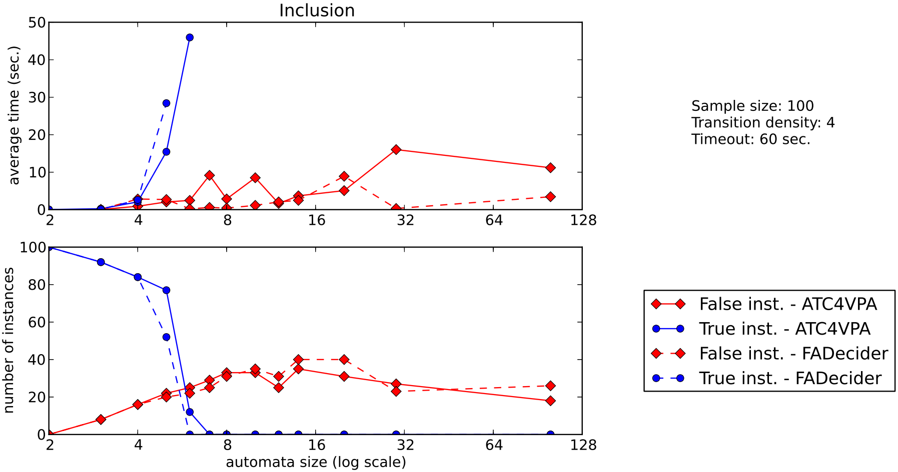
Question : $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$

Our goal : find $t \in T_\Sigma$: $t \in L(\mathcal{A})$ and $t \notin L(\mathcal{B})$

- $(q, q') \in Q_{\mathcal{A}} \times Q_{\mathcal{A}}$:
  - $(q, \epsilon) \xrightarrow{[t]} (q', \epsilon)$
  - $q \in Q_{i,\mathcal{A}}$ $q' \in Q_{f,\mathcal{A}}$
- $r \subseteq Q_{\mathcal{B}} \times Q_{\mathcal{B}}$:
  - $r = Acc_{\mathcal{B}}(t)$
  - $r \cap Q_{i,\mathcal{B}} \times Q_{f,\mathcal{B}} = \emptyset$

Easy adaptation of *Post* operator.

Experiments



Sample size: 100
Transition density: 4
Timeout: 60 sec.

Hedge automata

Algorithm is easily adapted from VPAs to hedge automata.

Translating the Hedge automaton via a VPA.

Universality of General VPAs - Three Types of Symbols

Original VPA model : $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i$ where

- Σ_c set of call symbols
- Σ_r set of return symbols
- Σ_i set of internal symbols

Adaptation of initialization : $\mathscr{R}^*$ (resp. $\mathscr{R}^*_{min}$) gets $\{id_Q\} \cup \bigcup_{a \in \Sigma_i} \{(q, q') \mid q \xrightarrow{a} q' \in \Delta\}$ instead of $\{id_Q\}$ (internal symbols can appear at any place)

Adaptation of *Post* : $Post_{a,\overline{b}}(r) =$

$\{(p, p') \in Q \times Q \mid \exists (q, q') \in r,\ p \xrightarrow{a:\gamma} q \in \Delta,\ q' \xrightarrow{\overline{b}:\gamma} p' \in \Delta\}$ for all $a \in \Sigma_c$ and $\overline{b} \in \Sigma_r$.

Universality of General VPAs - Pending Calls and Returns

General shape word with pending calls or pending returns :

$$w = [h_0]\overline{b}_0[h_1]\overline{b}_1 \cdots [h_m]\overline{b}_m \quad [h] \quad a_1[h'_1]a_2[h'_2]\cdots a_n[h'_n]$$

where all h_i, h'_j, and h are hedges over Σ (H_Σ), and $\overline{b_i} \in \overline{\Sigma}$, $a_j \in \Sigma$ for all i,j.

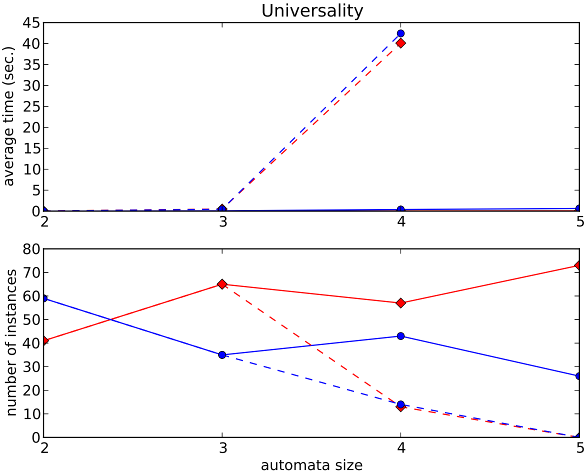
- w is a sequence of words $[h_i]\overline{b_i}$
- followed by the linearization of a hedge $[h]$
- followed by a sequence of words $a_j[h'_j]$

Adapt the algorithm to compute :

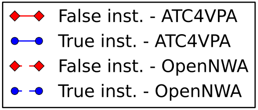
- $\mathscr{R}^* = \{Acc(h) \mid h \in H_\Sigma\}$
- $\mathscr{C}^*$, the closure by composition of $\mathscr{C} = \{Acc([h]\overline{b}) \mid h \in H_\Sigma,\ \overline{b} \in \overline{\Sigma}\}$.
- $\mathscr{O}^*$, the closure by composition of $\mathscr{O} = \{Acc(a[h]) \mid a \in \Sigma,\ h \in H_\Sigma\}$.

Check $\forall r_c \in \mathscr{C}^*$, $r_h \in \mathscr{R}^*$, $r_o \in \mathscr{O}^*$ if $r_c \circ r_h \circ r_o \cap Q_i \times Q_f \neq \emptyset$.

Experiments



Sample size: 100
Transition density: Σ_c:2.7/Σ_r:5/Σ_i:2
Timeout: 60 sec.



Future Work

- using the antichain universality finite ranked tree automata algorithms :
  - A. Bouajjani, P. Habermehl, L. Holik, T. Touili, and T. Vojnar (CIAA'08)
  - P. Abdulla, Y. Chen, L. Holik, R. Mayr and T. Vojnar (TACAS'10)
- p-universality checking
- simulation
- bisimulation up-to