Title	Visibly Stack Automata
Author(s)	Nguyen, Van Tang; Ogawa, Mizuhito
Citation	
Issue Date	2006-11-27
Type	Presentation
Text version	publisher
URL	http://hdl.handle.net/10119/8312
Rights	
Description	3rd VERITE : JAIST/TRUST-AIST/CVS joint workshop on VERIfication TEchnology , 2006 11 27 28 , JAIST

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Visibly Stack Automata

Nguyen Van Tang - Mizuhito Ogawa

School of Information Science
Japan Advanced Institute of Science and Technology

Talk Outline

❖ Automata-theoretic based verification

➢ Checking context-free specifications

➢ Obstacles

❖ Visibly Pushdown Automata

➢ Visibly Pushdown Languages

➢ Determinization

❖ Visibly Stack Automata

➢ Visibly Stack Languages

➢ Determinization

Automata-theoretic based verification

- To verify if a software system satisfies a regular specification

  ❖ System is modeled as a pushdown automaton M

  ❖ Requirement is specified as a finite automaton S

- M |= S iff:

  ❖ $L(M) \subseteq L(S)$

  ❖ $L(M) \cap L(S)^C = \varnothing$

- Model checking problems are reduced to decision problems of formal languages

- Model checker: SPIN, MAGIC,..

Checking Context-free Specifications

- When S is a context-free specification
- Checking M|= S becomes undecidable

- Obstacles
  - ➢ Context-free languages (CFL) are not closed under intersection, complementation
  - ➢ The inclusion problem of CFL is undecidable
- Goals: Find a class of non-regular languages
  - ➢ Enjoys closure properties, Inclusion problem is decidable
  - ➢ Robustness

Talk Outline

❖ Automata-theoretic based verification

  ➢ Checking context-free specifications

  ➢ Obstacles

❖ Visibly Pushdown Automata

  ➢ Visibly Pushdown Languages

  ➢ Determinization

❖ Visibly Stack Automata

  ➢ Visibly Stack Languages

  ➢ Determinization

5

Pushdown Automata (PDA)

- Def. PDA $(P,\Sigma,\Gamma,\Delta,p_0,Z_0,F)$ where
  - P : finite control locations
  - Γ : finite stack alphabet
  - Σ : finite input alphabet
  - $\Delta \subseteq (P \times \Gamma \times (\Sigma \cup \{\varepsilon\}) \times (P \times \Gamma^*))$: transition
  - p_0 : initial control location
  - Z_0: initial stack symbol
  - $F \subseteq P$: final control locations

- Accepted ⇔ run reaches some control location in F

- PDA are not determinizable. PDA are closed under union, but not closed under intersection, complementation

6

Visibly Pushdown Automata (VPA)

- VPA was introduced by J. Alur and P.Madhusudan in 2004, [p.202-211, ACM-STOC's 04]

- Pushdown alphabet: partitioned into 3 disjoint sets

$$\Sigma = \Sigma_{push} \cup \Sigma_{pop} \cup \Sigma_{local}$$

- A *visibly pushdown automaton* over a pushdown alphabet Σ is a pushdown automaton that

  ➢ pushes a symbol onto the stack on a symbol in Σ_{push}

  ➢ pops the stack on a symbol in Σ_{pop}

  ➢ cannot change the stack on a symbol in Σ_{local}

Key: Stack size at any time is determined by the input word but not control state or stack content

Visibly Pushdown Languages (VPL)

- A language L is a <u>VPL</u> over a pushdown alphabet Σ, if it is recognized by a VPA

- Examples
  - ➤ L = { $a^n b^n$ | $n \geq 1$, $a \in \Sigma_{push}$, $b \in \Sigma_{pop}$ }
  - ➤ Every regular language L is a VPL

- VPLs are closed under:
  - ➤ Union:
  - ➤ Intersection: Product construction works !
  - ➤ Complementation (see later)

M_1

α_2
α_1
$\perp$

M_2

γ_2
γ_1
$\perp$

→

M

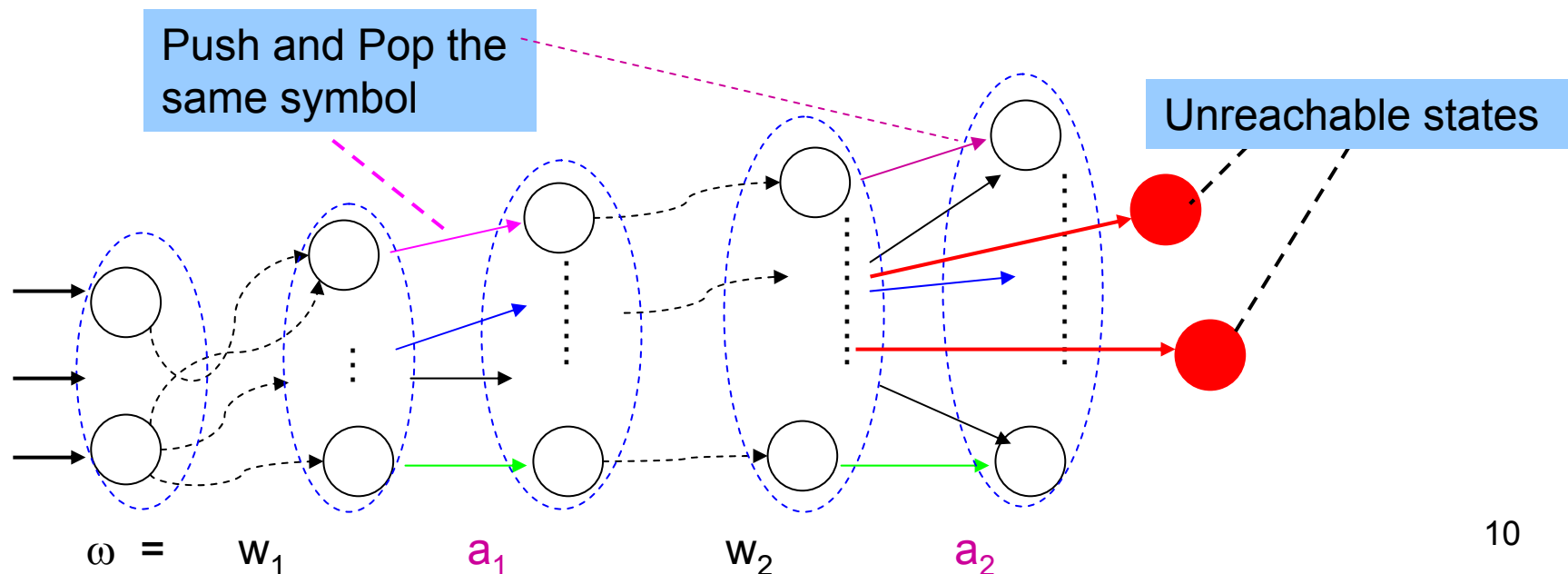| (α_2, γ_2) |
|:---:|
| (α_1, γ_1) |
| $\perp$ |

8
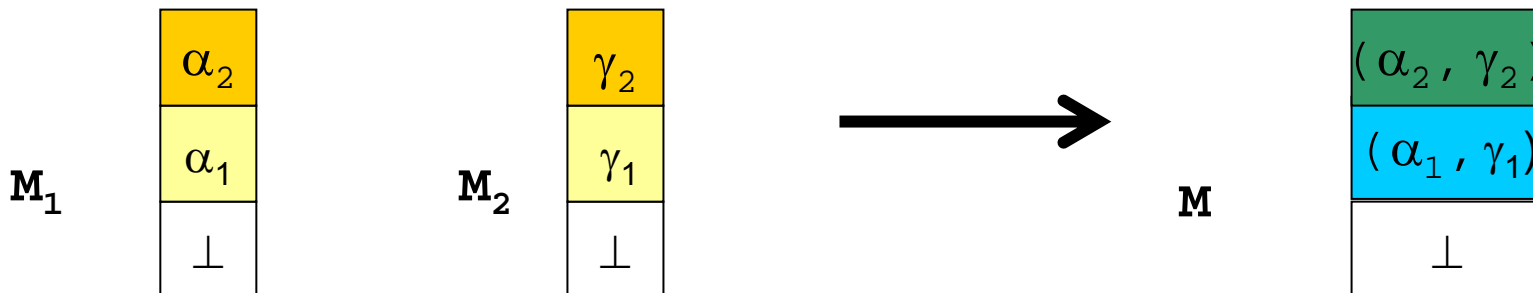
Decision Problems

- Given a nondeterministic *A* (n states), construct an equivalent deterministic *B* in size $O(2^{n^2})$.

- Emptiness: Decidable in polynomial-time (cubic)

- Language inclusion: $L(A) \subseteq L(B)$?

  - Determinize *B*, take its complement, take product with *A*, and test for emptiness

  - Exponential-time complete

- VPLs is a subclass of DCFLs (languages defined by deterministic PDAs)

  - DCFLs not closed under union, intersection

  - Equivalence problem for DCFLs decidable, but complex

Determinization: Key ideas

- **Def. A word u is *well-matched* if**
  - For each prefix u' of u, the number of Σ_{push} symbols in u' is at most the number of Σ_{pop} symbols in u'.
  - For each suffix u'' of u, the number of Σ_{pop} symbols in u'' is at most the number of Σ_{push} symbols in u''.

Push and Pop the same symbol

Unreachable states

ω = w_1 a_1 w_2 a_2

Determinization: Sketch of the construction

- Idea: a well-matched word preserves stack; thus regarded as internal transition (expressed as summaries S_i). Transitions by extra Σ_{push} symbols are postponed until corresponding Σ_{pop} symbols will be read.

- Determinized VPA will consist of :
  - Control locations : { (S,R) | S: summary, R: reachables}
  - Stack alphabet : { (S,R,a) | S, R; a$\in\Sigma_{push}$}
  - The initial state (Id, P_{init}), where Id = {(q,q)| q$\in$ P}
  - Final states { (S,R) | $R\cap$F$\neq\varphi$},
  - where
    - R ($\subseteq$ P)   = { all states reachable after a word w }
    - S ($\subseteq$ P$\times$P) = { all summaries on a well-marched word w }
                (i.e., (q,q') $\in$ S, if ($q,\perp$) can reach to ($q',\perp$)).

Determinization: Sketch of transitions

- Let $w = w_1 c_1 w_2 c_2 \ldots c_n w_{n+1}$, where c_i's are in Σ_{push}, w_i's are well matched words, let a be the next input.

  - Stack is $(S_n, R_n, c_n) \ldots (S_1, R_1, c_1) \perp$
  - Control location is (S_{n+1}, R_{n+1}),
  - If $a \in \Sigma_{\text{local}}$: (S_{n+1}, R_{n+1}) is combined with transitions by a.
  - If $a \in \Sigma_{\text{push}}$: push $(S_{n+1}, R_{n+1}, c_{n+1})$ and control location is (id, { q' | reachable from q$\in R_{n+1}$ by a})
  - If $a \in \Sigma_{\text{pop}}$: let *Update* be combination of transitions by c_n (push $\gamma \in \Gamma$), S_{n+1}, and transitions by a (pop same γ). *NewS* is S_n combined with *Update*, and *NewR* is R_n combined with *Update* (R_{n+1} is discarded).
  - where
  - R_i ($\subseteq$ P) = Set of all states reachable after $w_1 c_1 \ldots w_i$
  - S_i ($\subseteq$ P×P) = Set of all summaries on w_i

Talk Outline

❖ Automata-theoretic based verification

  ➢ Checking context-free specifications

  ➢ Obstacles

❖ Visibly Pushdown Automata

  ➢ Visibly Pushdown Languages

  ➢ Determinization

❖ Visibly Stack Automata

  ➢ Visibly Stack Languages

  ➢ Determinization

13

Stack Automata

- Stack automata: introduced by Ginsburg, Greibach and Harrison [JACM, No 2, Vol 14, pp.389-418, 1967]

- Stack automata = PDA + "read inside stack".

- More powerful than PDA. For instance, $\{a^n b^n c^n | n \geq 1\}$, $\{a^n b^{n^2} | n \geq 1\}$

- Def. Stack alphabet: $\Sigma = \Sigma_{push} \cup \Sigma_{pop} \cup \Sigma_{local} \cup \Sigma_{up} \cup \Sigma_{down}$

14

Visibly Stack Automata (VSA) (1/2)

- Def. A VSA A over stack alphabet Σ: A = $\langle$ P, P_{in}, Γ, $\uparrow$, δ, F $\rangle$
  - P : finite set of control locations
  - $P_{in} \subseteq$ P: set of initial control locations
  - Γ : finite stack alphabet, special symbols $\perp$, T
  - $\uparrow$: stack pointer
  - F $\subseteq$ P : set of final control locations
  - δ is a set of transitions $\langle \delta_{puah}, \delta_{pop}, \delta_{local}, \delta_{up}, \delta_{down} \rangle$,

$\delta_{push} \subseteq$ P$\times\Sigma_{push}\times$P$\times \Gamma\backslash\{\perp,T\}$; $\delta_{pop} \subseteq$ P$\times\Sigma_{pop}\times\Gamma\times$P;

$\delta_{local} \subseteq$P$\times\Sigma_{local}\times$P ;$\delta_{down} \subseteq$ P$\times\Sigma_{down}\times\Gamma\times$P; $\delta_{up} \subseteq$ P$\times\Sigma_{up}\times\Gamma\times$P;

- $(q,a,\gamma,q') \in \delta_{up}$ $(\gamma\neq\perp,T)$ $\Leftrightarrow$ $(q,a,\gamma',q') \in \delta_{up}$ $(\gamma'\neq \gamma, \perp,T)$
- $(q,a,\gamma,q') \in \delta_{down}$ $(\gamma\neq\perp,T)$ $\Leftrightarrow$ $(q,a,\gamma',q') \in \delta_{down}$ $(\gamma'\neq \gamma,$ 15,T)

Visibly Stack Automata (2/2)

- Properties:
  - Stack has form $T\gamma_n...\gamma_i\uparrow\gamma_{i-1}...\gamma_1\perp$
  - VSA can only push, pop when stack pointer at the top
  - When stack is empty, pop is read but not popped
  - Stack pointer cannot go beyond T or below $\perp$
  - When pointer is reading $\perp$ (T), down (up) is read but pointer does not move down (up)

- Def. A language L is a visibly stack language (VSL) if it is accepted by a VSA.

- Example: $L=\{\underline{a^n b^n c^n}|n\geq 1, a\in\sum_{push}, b\in\sum_d, c\in\sum_u\}$

- VSL class is a proper extension of VPL class

Closure properties & Decision problems

- Similar to VPLs, VSLs are closed under:
  - Union, intersection, complementation
  - Determinizable (more complicated, see later)

- Emptiness for stack automata is decidable!
  D. Harel, Information and Computation, Vol.113, No.2, 278-299, 1994

- Inclusion problem is decidable for visibly stack languages.

Determinization: Key ideas

- Matching condition between push and pop symbols

- Read the same symbol, whenever pointer goes up, or goes down at the same position
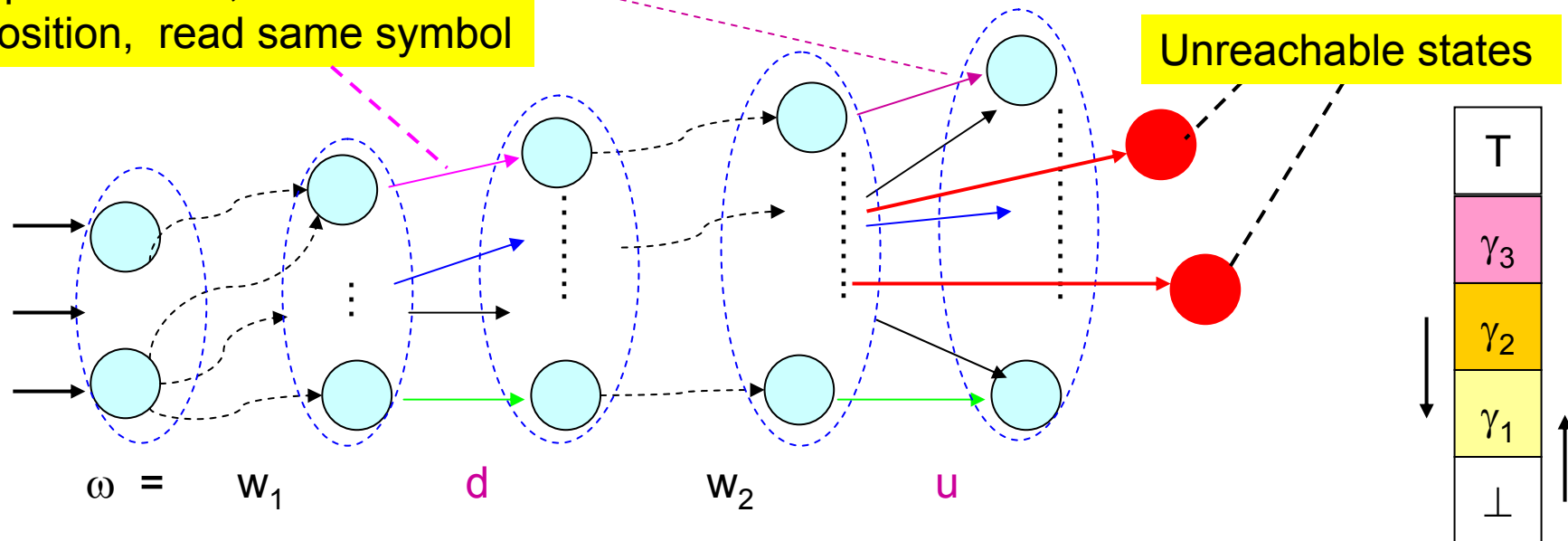
Determinization: Some definitions

- ## Def. A word u is an *up-down* segment if:

1. For each prefix u' of u, the number of Σ_{down} symbols in u' is at most the number of Σ_{up} symbols in u'.
2. For each suffix u'' of u, the number of Σ_{up} symbols in u'' is at most the number of Σ_{down} symbols in u''.
3. There is no push, pop symbols in u

- ## Def. A word u is *well-matched* if:

1. For each prefix u' of u, the number of Σ_{push} symbols in u' is at most the number of Σ_{pop} symbols in u'.
2. For each suffix u'' of u, the number of Σ_{pop} symbols in u'' is at most the number of Σ_{push} symbols in u''.
3. For each up (down) symbol a of u, a must belongs to an *up-down* segment ud, ud is a subword of u.

Determinization: Sketch of the construction

- **Determinized VSA will consist of :**
  - ➢ Control locations : { (S,R) | S: summary, R: reachables}
  - ➢ Stack alphabet : { (S,R,a) | S, R; a∈Σ_{push}}
  - ➢ The initial state (Id,P_{init}), where Id = {(q,q)| q∈ P}
  - ➢ Final states { (S,R) | R∩F≠φ},

  - – where
  - – R (⊆ P) = {all states reachable after a word w}
  - – S (⊆ P×P) = {all summaries on a well-marched word w} (i.e., (q,q') ∈ S, if $(q,T{\uparrow}\perp)$ can reach to $(q',T{\uparrow}\perp)$).

Determinization: Sketch of transitions

- Let $w = w_1 c_1 w_2 c_2 \ldots c_n w_{n+1}$, where c_i's are in Σ_{push}, w_i's are well matched words, let a be the next input.

  - Stack is $T \uparrow (S_n, R_n, c_n) \ldots (S_1, R_1, c_1) \perp$
  - Control location is (S_{n+1}, R_{n+1}),
  - If $a \in \Sigma_{\text{local}}$: (S_{n+1}, R_{n+1}) is updated using subset construction with transitions by a.
  - If $a \in \Sigma_{\text{down}}$: (S_{n+1}, R_{n+1}) is updated with transition by a. Stack now is $T(S_n, R_n, c_n) \uparrow \ldots (S_1, R_1, c_1) \perp$
  - If $a \in \Sigma_{\text{up}}$: (S_{n+1}, R_{n+1}) is updated with transitions by a, the pointer cannot go up. Stack stays unchanged, $T' \uparrow (S_n, R_n, c_n) \ldots (S_1, R_1, c_1) \perp$

  - where
  - $R_i (\subseteq P)$ = Set of all states reachable after $w_1 c_1 \ldots w_i$
  - $S_i (\subseteq P \times P)$ = Set of all summaries on w_i

Determinization: Sketch of transitions

- If $a \in \Sigma_{push}$: push $(S_{n+1}, R_{n+1}, c_{n+1})$ and control location is (id, { q' | reachable from $q \in R_{n+1}$ by a}). Stack now is $T \uparrow (S_{n+1}, R_{n+1}, c_{n+1})(S_n, R_n, c_n)...(S_1, R_1, c_1) \perp$

- If $a \in \Sigma_{pop}$: let *Update* be combination of transitions by c_n (push $\gamma \in \Gamma$), S_{n+1}, and transitions by a (pop same γ). *NewS* is S_n combined with *Update*, and *NewR* is R_n combined with *Update* (R_{n+1} is discarded). Stack now is $T \uparrow (S_{n-1}, R_{n-1}, c_{n-1})...(S_1, R_1, c_1) \perp$
  - where
  - $R_i (\subseteq P)$ = Set of all states reachable after $w_1 c_1 ... w_i$
  - $S_i (\subseteq P \times P)$ = Set of all summaries on w_i

Conclusion

- Proposed the class of visibly stack languages recognized by visibly stack automata

- To our knowledge, to date, VSLs is the largest class which enjoys closure properties. All the decision problems are decidable for VSA.

- Infinite words:
  - Visibly Büchi pushdown automata [AM04]
  - Visibly Büchi stack automata
  - Closure properties, not determinizable, but language inclusion is still decidable!

Thank for your attention !