Automata theory and its applications

Lecture 7-8: Visibly pushdown languages

Zhilin Wu

State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences

November 21, 2012

Outline

1. Visibly pushdown automata (VPA)
2. Closure properties
3. Visibly pushdown grammar (VPG)
4. Logical characterization
   - Equivalence of NFA and MSO
   - Equivalence of VPA and MSO_μ
5. Decision problems

Motivation

Parenthesises in arithmetic expressions

$$(\; (\; (\; 5 + x \;) * y + z \;) * (\; u - v \;) \;) \; / \; w$$

Motivation

Curly brackets in C Programs

```
for(...)
{
  if (...)
  {
    for (...)
    {
      ...
    }
  }
  else
  {
    ...
  }
}
```

Motivation

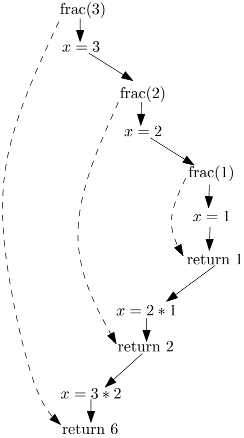
XML documents

⟨library⟩
⟨catalog⟩
⟨book⟩
⟨title⟩
Computational Complexity
⟨/title⟩
⟨author⟩
Christos H. Papadimitriou
⟨/author⟩
⟨book⟩
⟨catalog⟩
⟨library⟩

Motivation

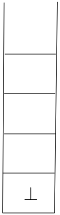
Recursive function calls and returns

```
frac(int y)
{
 int x = y;
 if (y >= 2)
 {
  x = y * frac(y − 1);
  return x;
 }
 else
 {
  return x;
 }
}
```



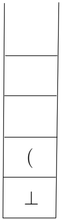
Motivation

$$(\ (\ (\ 5\ +\ x\ )\ *\ y\ +\ z\ )\ *\ (\ u\ -\ v\ )\ )\ /\ w$$



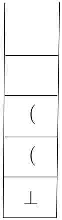
Motivation

$$(\; (\; (\; 5 \; + \; x \;) \; * \; y \; + \; z \;) \; * \; (\; u \; - \; v \;) \;) \; / \; w$$



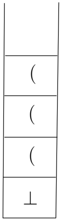
Motivation

$(\ (\ (\ 5 \ + \ x \ ) \ * \ y \ + \ z \ ) \ * \ (\ u \ - \ v \ ) \ ) \ / \ w$



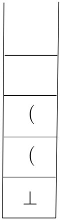
Motivation

$$(\; (\; (\; 5 \; + \; x \;) \; * \; y \; + \; z \;) \; * \; (\; u \; - \; v \;) \;) \; / \; w$$



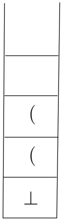
Motivation

$$(\; (\; (\; 5 \; + \; x \;) \; * \; y \; + \; z \;) \; * \; (\; u \; - \; v \;) \;) \; / \; w$$



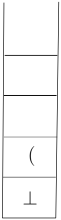
Motivation

$$(\; (\; (\; 5 \; + \; x \;) \; * \; y \; + \; z \;) \; * \; (\; u \; - \; v \;) \;) \; / \; w$$



Motivation

$$(\; (\; (\; 5 \; + \; x \;) \; * \; y \; + \; z \;) \; * \; (\; u \; - \; v \;) \;) \; / \; w$$



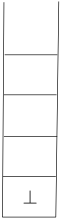
Motivation

Stack operations determined by the input symbol

$$(\; (\; (\; 5 \; + \; x \;) \; * \; y \; + \; z \;) \; * \; (\; u \; - \; v \;) \;) \; / \; w$$

(: Push

): Pop



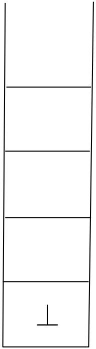
Motivation

Stack operations determined by the input symbol

```
for(...)
{
  if (...)
  {
    for (...)
    {
      ...
    }
  }
  else
  {
    ...
  }
}
```

{: Push

}: Pop



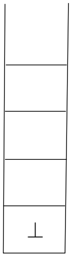
Motivation

Stack operations determined by the input symbol

$\langle$library$\rangle$
$\langle$catalog$\rangle$
$\langle$book$\rangle$
$\langle$title$\rangle$
Computational Complexity
$\langle$/title$\rangle$
$\langle$author$\rangle$
Christos H. Papadimitriou
$\langle$/author$\rangle$
$\langle$book$\rangle$
$\langle$catalog$\rangle$
$\langle$library$\rangle$

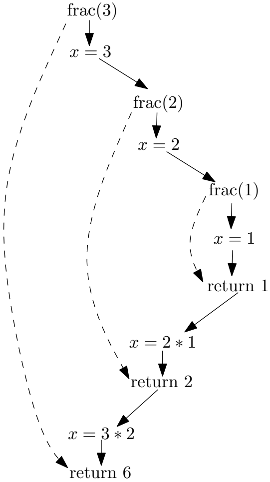
$\langle tag \rangle$: Push

$\langle /tag \rangle$: Pop

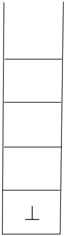


Motivation

Stack operations determined by the input symbol



frac: Push
return: Pop



Visibly pushdown automata (VPA)

The alphabet Σ is partitioned into $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$

- Σ_c: finite set of calls,
- Σ_r: finite set of returns,
- Σ_l: finite set of local actions.

A (nondeterministic) VPA $\mathcal{A}$ is a tuple $(Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \bot, F)$, where

- Q is a finite set of states,
- $\widetilde{\Sigma}$ is the input alphabet,
- Γ is the stack alphabet,
- $\delta \subseteq Q \times \Sigma_c \times Q \times (\Gamma \backslash \{\bot\}) \ \cup \ Q \times \Sigma_r \times \Gamma \times Q \ \cup \ Q \times \Sigma_l \times Q$,
- q_0 is the initial state,
- $\bot$ is the bottom symbol of the stack,
- $F \subseteq Q$ is the set of final states.

Visibly pushdown automata (VPA)

The alphabet Σ is partitioned into $\tilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$

- Σ_c: finite set of calls,
- Σ_r: finite set of returns,
- Σ_l: finite set of local actions.

A (nondeterministic) VPA $\mathcal{A}$ is a tuple $(Q, \tilde{\Sigma}, \Gamma, \delta, q_0, \bot, F)$, where

- Q is a finite set of states,
- $\tilde{\Sigma}$ is the input alphabet,
- Γ is the stack alphabet,
- $\delta \subseteq Q \times \Sigma_c \times Q \times (\Gamma \backslash \{\bot\}) \ \cup \ Q \times \Sigma_r \times \Gamma \times Q \ \cup \ Q \times \Sigma_l \times Q$,
- q_0 is the initial state,
- $\bot$ is the bottom symbol of the stack,
- $F \subseteq Q$ is the set of final states.

Remark:

- No ε-transitions,
- Exactly one symbol is pushed in each call transition.

Visibly pushdown automata (VPA)

A (nondeterministic) VPA $\mathcal{A}$ is a tuple $(Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$, where

- Q is a finite set of states,
- $\widetilde{\Sigma}$ is the input alphabet,
- Γ is the stack alphabet,
- $\delta \subseteq Q \times \Sigma_c \times Q \times (\Gamma \backslash \{\perp\}) \ \cup \ Q \times \Sigma_r \times \Gamma \times Q \ \cup \ Q \times \Sigma_l \times Q$,
- q_0 is the initial state,
- $\perp$ is the bottom symbol of the stack,
- $F \subseteq Q$ is the set of final states.

A *deterministic* VPA is a VPA $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, F)$ such that

- for every $(q, a) \in Q \times \Sigma_c$, there is at most one pair $(q', \gamma) \in Q \times (\Gamma \backslash \{\perp\})$ such that $(q, a, q', \gamma) \in \delta$,
- for every $(q, a, \gamma) \in Q \times \Sigma_r \times \Gamma$, there is at most one $q' \in Q$ such that $(q, a, \gamma, q') \in \delta$,
- for every $(q, a) \in Q \times \Sigma_l$, there is at most one $q' \in Q$ such that $(q, a, q') \in \delta$.

A deterministic VPA is *complete* if "at most" is replaced by "exactly".

Visibly pushdown automata (VPA): continued

A run of a VPA $\mathcal{A}$ over a word $w = a_1 \dots a_n$ is

a sequence $(q_0, \alpha_0)(q_1, \alpha_1) \dots (q_n, \alpha_n)$ s.t.

- $\forall i. q_i \in Q$,
- $\alpha_0 = \perp$,
- $\forall i : 1 \leqslant i < n$, one of the following holds,

Call $a_i \in \Sigma_c$, $\exists \gamma \in \Gamma \backslash \{\perp\}.(q_i, a_i, q_{i+1}, \gamma) \in \delta$, $\alpha_{i+1} = \gamma \alpha_i$,

Return $a_i \in \Sigma_r$,

- $\exists \gamma \in \Gamma \backslash \{\perp\}.(q_i, a_i, \gamma, q_{i+1}) \in \delta$, $\alpha_i = \gamma \alpha_{i+1}$,
- or $(q_i, a_i, \perp, q_{i+1}) \in \delta$ and $\alpha_i = \alpha_{i+1} = \perp$.

Local $a_i \in \Sigma_l$, $(q_i, a_i, q_{i+1}) \in \delta$ and $\alpha_{i+1} = \alpha_i$.

Visibly pushdown automata (VPA): continued

A run of a VPA $\mathcal{A}$ over a word $w = a_1 \ldots a_n$ is

a sequence $(q_0, \alpha_0)(q_1, \alpha_1) \ldots (q_n, \alpha_n)$ s.t.

- $\forall i. q_i \in Q$,
- $\alpha_0 = \perp$,
- $\forall i : 1 \leqslant i < n$, one of the following holds,

Call $a_i \in \Sigma_c$, $\exists \gamma \in \Gamma \backslash \{\perp\}.(q_i, a_i, q_{i+1}, \gamma) \in \delta$, $\alpha_{i+1} = \gamma\alpha_i$,

Return $a_i \in \Sigma_r$,

- $\exists \gamma \in \Gamma \backslash \{\perp\}.(q_i, a_i, \gamma, q_{i+1}) \in \delta$, $\alpha_i = \gamma\alpha_{i+1}$,
- or $(q_i, a_i, \perp, q_{i+1}) \in \delta$ and $\alpha_i = \alpha_{i+1} = \perp$.

Local $a_i \in \Sigma_l$, $(q_i, a_i, q_{i+1}) \in \delta$ and $\alpha_{i+1} = \alpha_i$.

A run $(q_0, \alpha_0) \ldots (q_n, \alpha_n)$ is *accepting* if $q_n \in F$.

A word w is accepted by a VPA $\mathcal{A}$ if $\exists$ an accepting run of $\mathcal{A}$ over w.

The set of words accepted by $\mathcal{A}$ is denoted by $\mathcal{L}(\mathcal{A})$.

Remark: Acceptance of VPAs are defined by final states, not by empty stack.

Well-matched words

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$.
The set of *well-matched* words $w \in \Sigma^*$ is defined inductively as follows,

- ε is well-matched,
- if w' is well matched, then
  $$w = aw' \textit{ or } w = w'a \textit{ such that } a \in \Sigma_l \textit{ is well-matched,}$$
- if w' is well-matched, then
  $$w = aw'b \textit{ such that } a \in \Sigma_c, b \in \Sigma_r \textit{ is well-matched.}$$
- if w' and w'' are well-matched, then
  $$w = w'w'' \textit{ is well-matched.}$$

Example: (())() is well-matched, while neither ()()) nor (() is.

Remark. As a result of the acceptance by final states,

VPAs over $\widetilde{\Sigma}$ may accept non-well-matched words.

Visibly pushdown languages (VPL)

A language $L \subseteq \Sigma^*$ is a *visibly pushdown language* with respect to $\widetilde{\Sigma}$ if *there is a VPA $\mathcal{A}$ over $\widetilde{\Sigma}$, satisfying that $\mathcal{L}(\mathcal{A}) = L$.*

Example:

The language $\{a^n b^n \mid n \geqslant 1\}$ is a VPL with respect to $\widetilde{\Sigma} = \langle \{a\}, \{b\}, \varnothing \rangle$.

Homework

Let $L \subseteq \{a, b\}^*$ be the set of words with "equal number of a's and b's".

Prove that L is not a VPL with respect to any partition of $\Sigma = \{a, b\}$.

Embedding of CFL as VPLs

Proposition. For every CFL $L \subseteq \Sigma^*$, there are a VPL $L' \subseteq (\Sigma')^*$ with respect to some $\tilde{\Sigma}'$ and a homomorphism $h : (\Sigma')^* \to \Sigma^*$ such that $L = h(L')$.

Let L be a CFL defined by a PDA $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ (accept. by final states).

W.l.o.g, suppose that each $(q, a, X, \alpha) \in \delta$ satisfies that

$\alpha = \varepsilon$ (pop) or $\alpha = X$ (stable) or $\alpha = YX$ (push).

Let $\Sigma' = (\Sigma \cup \{\sigma_\varepsilon\}) \times \{c, r, l\}$ and

$\tilde{\Sigma}' = \langle(\Sigma \cup \{\sigma_\varepsilon\}) \times \{c\}, (\Sigma \cup \{\sigma_\varepsilon\}) \times \{r\}, (\Sigma \cup \{\sigma_\varepsilon\}) \times \{l\}\rangle$

From $\mathcal{A}$, define a VPA $\mathcal{A}' = (Q, \tilde{\Sigma}', \Gamma, \delta', q_0, Z_0, F)$ over $\tilde{\Sigma}'$, where δ' is defined by the following rules,

- if $(q, a, X, q', \varepsilon) \in \delta$, then $(q, (a, r), X, q') \in \delta'$,
- if $(q, a, X, q', X) \in \delta$, then add a new state q_1,
  $(q, (a, r), X, q_1), (q_1, (\sigma_\varepsilon, c), q_2, X) \in \delta'$.
- if $(q, a, X, q', YX) \in \delta$, then add two new states q_1, q_2, and
  $(q, (a, r), X, q_1), (q_1, (\sigma_\varepsilon, c), q_2, X), (q_2, (\sigma_\varepsilon, c), q', Y) \in \delta'$.

Let $h : (\Sigma')^* \to \Sigma^*$ be a homomorphism defined by

$\forall a \in \Sigma, s \in \{c, r, l\}.\ h((a, s)) = a, h(\sigma_\varepsilon, s) = \varepsilon$.

Then $L = h(\mathcal{L}(\mathcal{A}'))$.

Outline

Union and intersection

Proposition. VPLs with respect to $\widetilde{\Sigma}$ are closed under union and intersection.
Let $\mathcal{A}_1 = (Q_1, \widetilde{\Sigma}, \Gamma_1, \delta_1, q_0^1, \bot_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \widetilde{\Sigma}, \Gamma_2, \delta_2, q_0^2, \bot_2, F_2)$ be two VPAs.

Union.

Without loss of generality, suppose $\bot_1 = \bot_2 = \bot$.

The VPA $\mathcal{A} = (Q_1 \cup Q_2 \cup \{q_0\}, \widetilde{\Sigma}, \Gamma_1 \cup \Gamma_2, \delta, q_0, \bot, F_1 \cup F_2)$ such that

$$\delta = \begin{array}{l} \qquad\qquad\qquad \delta_1 \cup \delta_2 \cup \\ \{(q_0, a, q', \gamma) \mid (q_0^1, a, q', \gamma) \in \delta_1 \text{ or } (q_0^2, a, q', \gamma) \in \delta_2\} \cup \\ \quad \{(q_0, a, \gamma, q') \mid (q_0^1, a, \gamma, q') \in \delta_1 \text{ or } (q_0^2, a, \gamma, q') \in \delta_2\} \end{array}$$

defines $\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$.

Intersection.

The VPA $\mathcal{A} = (Q_1 \times Q_2, \widetilde{\Sigma}, \Gamma_1 \times \Gamma_2, \delta, (q_0^1, q_0^2), (\bot_1, \bot_2), F_1 \times F_2)$ such that

$$\delta = \begin{array}{l} \{((q_1, q_2), a, (q_1', q_2'), (\gamma_1, \gamma_2)) \mid (q_1, a, q_1', \gamma_1) \in \delta_1, (q_2, a, q_2', \gamma_2) \in \delta_2\} \cup \\ \quad \{((q_1, q_2), a, (\gamma_1, \gamma_2), (q_1', q_2')) \mid (q_1, a, \gamma_1, q_1') \in \delta_1, (q_2, a, \gamma_2, q_2') \in \delta_2\} \end{array}$$

defines $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

Complementation

Theorem. For every VPA $\mathcal{A}$, a deterministic VPA $\mathcal{A}'$ can be constructed such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

Corollary. VPLs with respect to $\widetilde{\Sigma}$ are closed under complementation.

Proof.

Suppose L is defined by a complete deterministic VPA $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$.

Then $\mathcal{A}' = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, Q \backslash F)$ defines $\Sigma^* \backslash \mathcal{L}(\mathcal{A})$. $\square$

Complementation

Illustration of the intuition of the proof of the Theorem.

In an obviously way, we can define $(q, \alpha) \xrightarrow{w} (q', \alpha')$:

the reachability of the config. (q', α') *from* (q, α) *by reading* w.

Observation. Suppose $(q, \alpha) \xrightarrow{w} (q', \alpha')$ and w is well-matched, then $\alpha = \alpha'$.

Point I.

A well-matched word w *can be seen as a relation* $S_w \subseteq Q \times Q$,

without changing the content of the stack.

Complementation

Illustration of the intuition of the proof of the Theorem.

In an obviously way, we can define $(q, \alpha) \xrightarrow{w} (q', \alpha')$:

the reachability of the config. (q', α') from (q, α) by reading w.

Observation. Suppose $(q, \alpha) \xrightarrow{w} (q', \alpha')$ and w is well-matched, then $\alpha = \alpha'$.

Point I.

A well-matched word w can be seen as a relation $S_w \subseteq Q \times Q$, without changing the content of the stack.

Point II.

Suppose w is well-matched.

- $S_\varepsilon = \mathrm{Id}_Q$.
- If $w = aw'$ with $a \in \Sigma_l$, then
$$S_w = \{(q, q') \mid \exists q''.(q, a, q'') \in \delta, (q'', q') \in S_{w'}\}.$$
Similarly for $w = w'a$.
- If $w = aw'b$ with $a \in \Sigma_c$ and $b \in \Sigma_r$, then
$S_w = \{(q, q') \mid \exists q_1, q_2, \gamma.\ (q, a, q_1, \gamma) \in \delta, (q_1, q_2) \in S_{w'}, (q_2, b, \gamma, q') \in \delta\}.$

Complementation

Illustration of the intuition of the proof of the Theorem.

Point III. Get inspirations from the subset construction for NFAs.

Question:

What info. should be remembered after reading a word w in a NFA?

Complementation

Illustration of the intuition of the proof of the Theorem.

Point III. Get inspirations from the subset construction for NFAs.

Question:

What info. should be remembered after reading a word w in a NFA?

Answer:

The set of states reachable from q_0 after reading w.

Complementation

Illustration of the intuition of the proof of the Theorem.

Point III. Get inspirations from the subset construction for NFAs.

Question:

What info. should be remembered
after reading a word w in a nondeterministic VPA?

Complementation

Illustration of the intuition of the proof of the Theorem.

Point III. Get inspirations from the subset construction for NFAs.

Question:

What info. should be remembered
after reading a word w in a nondeterministic VPA?

Answer:

Let me think for a while . . .

Complementation

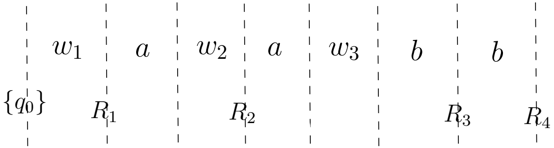
Illustration of the intuition of the proof of the Theorem.

Point III. Get inspirations from the subset construction for NFAs.

Question:

What info. should be remembered
after reading a word w in a nondeterministic VPA?

Consider $w_1 a w_2 a w_3 bb$ s.t. $a \in \Sigma_c, b \in \Sigma_r$ and w_1, w_2, w_3 are well-matched.



Complementation

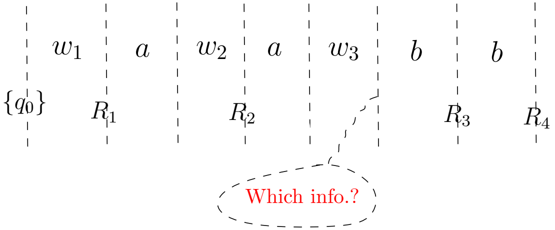
Illustration of the intuition of the proof of the Theorem.

Point III. Get inspirations from the subset construction for NFAs.

Question:

What info. should be remembered
after reading a word w in a nondeterministic VPA?

Consider $w_1 a w_2 a w_3 bb$ s.t. $a \in \Sigma_c, b \in \Sigma_r$ and w_1, w_2, w_3 are well-matched.



Complementation

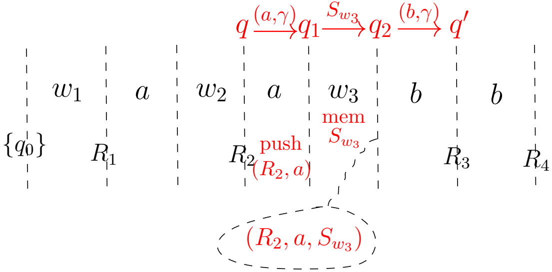
Illustration of the intuition of the proof of the Theorem.

Point III. Get inspirations from the subset construction for NFAs.

Question:

What info. should be remembered
after reading a word w in a nondeterministic VPA?

Consider $w_1 a w_2 a w_3 bb$ s.t. $a \in \Sigma_c, b \in \Sigma_r$ and w_1, w_2, w_3 are well-matched.



Complementation

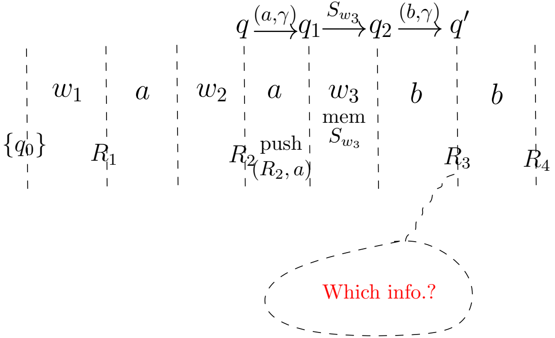
Illustration of the intuition of the proof of the Theorem.

Point III. Get inspirations from the subset construction for NFAs.

Question:

What info. should be remembered
after reading a word w in a nondeterministic VPA?

Consider $w_1 a w_2 a w_3 bb$ s.t. $a \in \Sigma_c, b \in \Sigma_r$ and w_1, w_2, w_3 are well-matched.



Complementation

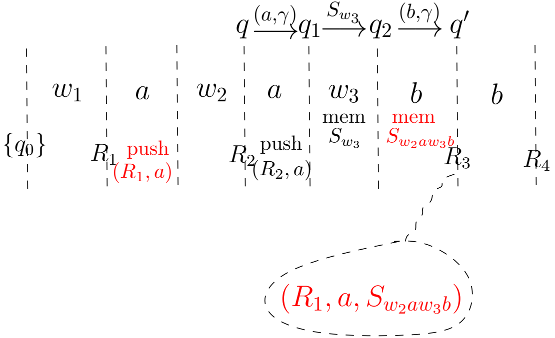
Illustration of the intuition of the proof of the Theorem.

Point III. Get inspirations from the subset construction for NFAs.

Question:

What info. should be remembered
after reading a word w in a nondeterministic VPA?

Consider $w_1aw_2aw_3bb$ s.t. $a \in \Sigma_c, b \in \Sigma_r$ and w_1, w_2, w_3 are well-matched.



Complementation

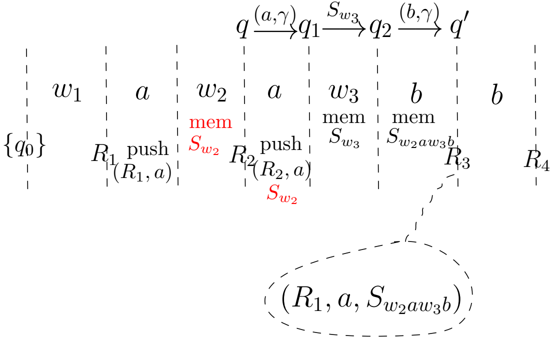
Illustration of the intuition of the proof of the Theorem.

Point III. Get inspirations from the subset construction for NFAs.

Question:

What info. should be remembered
after reading a word w in a nondeterministic VPA?

Consider $w_1 a w_2 a w_3 bb$ s.t. $a \in \Sigma_c, b \in \Sigma_r$ and w_1, w_2, w_3 are well-matched.



Complementation: continued

The construction for determinization $\mathcal{A}' = (Q', \tilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), F')$:

- Q': (S, R) such that $S \subseteq Q \times Q, R \subseteq Q$,
- Γ': letters (S, R, a) such that $S \subseteq Q \times Q, R \subseteq Q, a \in \Sigma_c$,
- $F' = \{(S, R) \mid R \cap F \neq \emptyset\}$,
- δ':

Local if $a \in \Sigma_l$, then $((S, R), a, (S', R')) \in \delta'$, where
$R' = \{q' \mid \exists q \in R.(q, a, q') \in \delta\}$, $S' = \{(q, q') \mid \exists q_1.(q, q_1) \in S, (q_1, a, q') \in \delta\}$.

Call if $a \in \Sigma_c$, then $((S, R), a, (\mathrm{Id}_Q, R'), (S, R, a)) \in \delta'$, where
$R' = \{q' \mid \exists q \in R, \gamma \in \Gamma.(q, a, q', \gamma) \in \delta\}$.

Return if $a \in \Sigma_r$, then $((S, R), a, (S'', R'', a'), (S', R')) \in \delta'$, where

$$S' = \left\{ (q, q') \;\middle|\; \begin{array}{c} \exists q_1, q_2, q_3, \gamma \in \Gamma. \\ (q, q_1) \in S'', (q_1, a', q_2, \gamma) \in \delta, (q_2, q_3) \in S, (q_3, a, \gamma, q') \in \delta \end{array} \right\},$$

$$R' = \left\{ q' \;\middle|\; \begin{array}{c} \exists q_1, q_2, q_3, \gamma \in \Gamma. \\ q_1 \in R'', (q_1, a', q_2, \gamma) \in \delta, (q_2, q_3) \in S, (q_3, a, \gamma, q') \in \delta \end{array} \right\},$$

or $((S, R), a, \perp, (S', R')) \in \delta'$, where
$S' = \{(q, q') \mid \exists q''.(q, q'') \in S, (q'', a, \perp, q') \in \delta\}$,
$R' = \{q' \mid \exists q \in R.(q, a, \perp, q') \in \delta\}$.

Outline

1. Visibly pushdown automata (VPA)
2. Closure properties
3. Visibly pushdown grammar (VPG)
4. Logical characterization
   - Equivalence of NFA and MSO
   - Equivalence of VPA and MSO_{μ}
5. Decision problems

Visibly pushdown grammar (VPG)

A CFG $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ is a VPG over $\widetilde{\Sigma}$ if $\mathcal{N}$ can be partitioned into $\mathcal{N}_0$ and $\mathcal{N}_1$, and each rule in $\mathcal{P}$ is of the following forms,

- $X \to \varepsilon$,
- $X \to aY$ such that if $X \in \mathcal{N}_0$, then $a \in \Sigma_l, Y \in \mathcal{N}_0$,
- $X \to aYbZ$ such that $a \in \Sigma_c$, $b \in \Sigma_r$, $Y \in \mathcal{N}_0$, and if $X \in \mathcal{N}_0$, then $Z \in \mathcal{N}_0$.

Example: Let $\widetilde{\Sigma} = (\{a\}, \{b\}, \emptyset)$. Then the VPG

$$S \to aSbC \mid aTbC, T \to \varepsilon, C \to \varepsilon,$$

such that $\mathcal{N}_0 = \{S, T, C\}$ defines $\{a^n b^n \mid n \geqslant 1\}$.

Equivalence of VPA and VPG

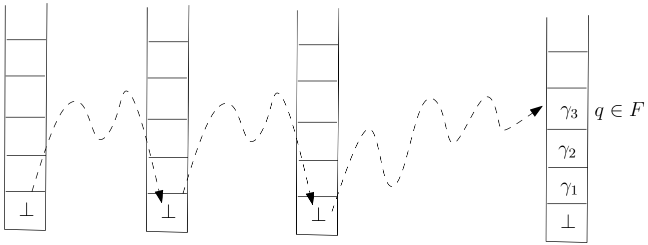
Theorem. VPA $\equiv$ VPG.

From VPA to VPG.

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA.

The intuition: Utilizing the nonterminals $[q, \gamma, p]$ with the meaning

the top symbol of the stack is γ,
and from state q,
by reading a well-matched word,
state p can be reached.



Equivalence of VPA and VPG

Theorem. VPA $\equiv$ VPG.

From VPA to VPG.

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA.

Construct a VPG $(\mathcal{N}_0, \mathcal{N}_1, \widetilde{\Sigma}, \mathcal{P}, S)$ as follows.

- $Q = \{(q, \perp) \mid q \in Q\} \cup \{q \mid q \in Q\} \cup \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \backslash \{\perp\}\}$,
  - $(q, \perp)$: the state is q and the stack is empty,
  - q: the state is q and the stack is nonempty.
- $\mathcal{N}_0 = \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \backslash \{\perp\}\}$, $S = (q_0, \perp)$,
- $\mathcal{P}$ is defined by the following rules,
  - if $(q, a, q') \in \delta$ s.t. $a \in \Sigma_l$, then
    $(q, \perp) \to a(q', \perp)$, $q \to aq'$, $[q, \gamma, p] \to a[q', \gamma, p]$.
  - if $(q, a, q', \gamma), (p', b, \gamma, p) \in \delta$ s.t. $a \in \Sigma_c, b \in \Sigma_r$, then
    $[q, \gamma_1, r] \to a[q', \gamma, p']b[p, \gamma_1, r]$, $(q, \perp) \to a(q', \gamma, p')b(p, \perp)$,
    $q \to a(q', \gamma, p')bp$.
  - if $(q, a, q', \gamma) \in \delta$ s.t. $a \in \Sigma_c$, then
    $(q, \perp) \to aq'$, $q \to aq'$, $(q, \perp) \to a[q', \gamma, p]$, $q \to a[q', \gamma, p]$.
  - if $(q, a, \perp, q') \in \delta$ s.t. $a \in \Sigma_r$, then $(q, \perp) \to a(q', \perp)$.
  - $\forall q \in Q. [q, \gamma, q] \to \varepsilon$,
  - $\forall q \in F. q \to \varepsilon, (q, \perp) \to \varepsilon$.

Equivalence of VPA and VPG: continued

From VPG to VPA.
Let $G = (\mathcal{N}_0, \mathcal{N}_1, \widetilde{\Sigma}, \mathcal{P}, S)$ be a VPG.
Construct VPA $\mathcal{A} = (\mathcal{N}, \widetilde{\Sigma}, \Sigma_r \times \mathcal{N} \cup \{\bot, \$\}, \delta, S, F)$ as follows.

- δ is defined by the following rules,
  - if $X \to aY$ s.t. $a \in \Sigma_l$, then $(X, a, Y) \in \delta$,
  - if $X \to aY$ s,t. $a \in \Sigma_c$, then $(X, a, Y, \$) \in \delta$,
  - if $X \to aY$ s.t. $a \in \Sigma_r$, then $(X, a, \$, Y) \in \delta$ and $(X, a, \bot, Y) \in \delta$,
  - if $X \to aYbZ$, then $(X, a, Y, (b, Z)) \in \delta$,
  - if $X \to \varepsilon$ and $X \in \mathcal{N}_0$, then $(X, b, (b, Y), Y) \in \delta$.
- $\mathcal{A}$ accepts if the state is in X s.t. $X \to \varepsilon$ and the top symbol is \$ or $\bot$.

Equivalence of VPA and VPG: continued

From VPG to VPA.
Let $G = (\mathcal{N}_0, \mathcal{N}_1, \widetilde{\Sigma}, \mathcal{P}, S)$ be a VPG.
Construct VPA $\mathcal{A} = (\mathcal{N}, \widetilde{\Sigma}, \Sigma_r \times \mathcal{N} \cup \{\perp, \$\}, \delta, S, F)$ as follows.

- δ is defined by the following rules,
  - if $X \to aY$ s.t. $a \in \Sigma_l$, then $(X, a, Y) \in \delta$,
  - if $X \to aY$ s,t. $a \in \Sigma_c$, then $(X, a, Y, \$) \in \delta$,
  - if $X \to aY$ s.t. $a \in \Sigma_r$, then $(X, a, \$, Y) \in \delta$ and $(X, a, \perp, Y) \in \delta$,
  - if $X \to aYbZ$, then $(X, a, Y, (b, Z)) \in \delta$,
  - if $X \to \varepsilon$ and $X \in \mathcal{N}_0$, then $(X, b, (b, Y), Y) \in \delta$.
- $\mathcal{A}$ accepts if the state is in X s.t. $X \to \varepsilon$ and the top symbol is \$ or $\perp$.

  Adapt $\mathcal{A}$ *into* $\mathcal{A}' = (\mathcal{N} \times \Gamma, \widetilde{\Sigma}, \Gamma, \delta', (S, \perp), \{(X, \gamma) \mid X \to \varepsilon, \gamma = \$, \perp\})$
  by adding the top symbol of the stack into the states.

  - if $X \to aY$ s.t. $a \in \Sigma_l$, then $\forall \gamma.((X, \gamma), a, (Y, \gamma)) \in \delta'$,
  - if $X \to aY$ s,t. $a \in \Sigma_c$, then $\forall \gamma.((X, \gamma), a, (Y, \$), (\$, \gamma)) \in \delta'$,
  - if $X \to aY$ s.t. $a \in \Sigma_r$, then
    $\forall \gamma.((X, \gamma), a, \perp, (Y, \perp)) \in \delta$ and $\forall \gamma.((X, \$), a, (\$, \gamma), (Y, \gamma)) \in \delta'$,
  - if $X \to aYbZ$, then $\forall \gamma.((X, \gamma), a, (Y, (b, Z)), ((b, Z), \gamma)) \in \delta'$,
  - if $X \to \varepsilon$ and $X \in \mathcal{N}_0$, then $\forall \gamma.((X, (b, Z)), b, ((b, Z), \gamma), (Z, \gamma)) \in \delta'$.

Outline

1. Visibly pushdown automata (VPA)
2. Closure properties
3. Visibly pushdown grammar (VPG)
4. Logical characterization
   - Equivalence of NFA and MSO
   - Equivalence of VPA and MSO_{μ}
5. Decision problems

Outline

Monadic Second-Order Logic (MSO)

Syntax.

$\varphi := P_\sigma(x) \mid x = y \mid \mathrm{suc}(x, y) \mid X(x) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists x \varphi_1 \mid \exists X \varphi_1$,

where $\sigma \in \Sigma$.

An MSO *sentence* is a MSO formula without free variables.

Semantics.

A *structure* $\mathcal{S}$ over Σ is

- a domain $S = \{1, \ldots, n\}$,
- an interpretation of all the unary predicates $P_\sigma \in \Sigma$ over S, denoted by $(P_\sigma)^{\mathcal{S}}$.

Example. Let $\Sigma = \{a, b\}$. Then $\mathcal{S} = (\{1, 2, 3\}, (P_a)^{\mathcal{S}} = \{1\}, (P_b)^{\mathcal{S}} = \{2, 3\})$ is a structure over Σ.

A word $w = a_1 \ldots a_n$ can be seen as a structure $\mathcal{S}_w$ over Σ,

- the domain of $\mathcal{S}_w$, denoted by S_w, is $\{1, \ldots, n\}$,
- the interpretation of every $P_\sigma \in \Sigma$ is the set of positions with the letter σ in w.

Monadic Second-Order Logic (MSO)

Syntax.

$\varphi := P_\sigma(x) \mid x = y \mid \mathrm{suc}(x, y) \mid X(x) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists x \varphi_1 \mid \exists X \varphi_1$,

where $\sigma \in \Sigma$.

An MSO *sentence* is a MSO formula without free variables.

Semantics.

Given a MSO formula φ, a *valuation* of $\mathrm{free}(\varphi)$ over a structure $\mathcal{S}$ is a mapping $\mathcal{I}$ such that

- for every $x \in \mathrm{free}(\varphi)$, $\mathcal{I}(x) \in S$,
- for every $X \in \mathrm{free}(\varphi)$, $\mathcal{I}(X) \subseteq S$.

Monadic Second-Order Logic (MSO)

Syntax.

$\varphi := P_\sigma(x) \mid x = y \mid \mathrm{suc}(x, y) \mid X(x) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists x \varphi_1 \mid \exists X \varphi_1$,

where $\sigma \in \Sigma$.

An MSO *sentence* is a MSO formula without free variables.

Semantics.

A MSO formula φ is satisfied over a word $w = a_1 \dots a_n$, with a valuation $\mathcal{I}$ of free(φ) over $\mathcal{S}_w$, denoted by $(w, \mathcal{I}) \models \varphi$, is defined as follows,

- $(w, \mathcal{I}) \models P_\sigma(x)$ iff $a_{\mathcal{I}(x)} = \sigma$,
- $(w, \mathcal{I}) \models x = y$ iff $\mathcal{I}(x) = \mathcal{I}(y)$,
- $(w, \mathcal{I}) \models \mathrm{suc}(x, y)$ iff $\mathcal{I}(x) + 1 = \mathcal{I}(y)$,
- $(w, \mathcal{I}) \models X(x)$ iff $\mathcal{I}(x) \in \mathcal{I}(X)$,
- $(w, \mathcal{I}) \models \varphi_1 \vee \varphi_2$ iff $(w, \mathcal{I}) \models \varphi_1$ or $(w, \mathcal{I}) \models \varphi_2$,
- $(w, \mathcal{I}) \models \neg\varphi_1$ iff not $(w, \mathcal{I}) \models \varphi_1$,
- $(w, \mathcal{I}) \models \exists x \varphi_1$ iff there is $j \in S_w$ such that $(w, \mathcal{I}[x \to j]) \models \varphi_1$,
- $(w, \mathcal{I}) \models \exists X \varphi_1$ iff there is $J \subseteq S_w$ such that $(w, \mathcal{I}[X \to J]) \models \varphi_1$.

Monadic Second-Order Logic (MSO)

Syntax.

$\varphi := P_\sigma(x) \mid x = y \mid \mathrm{suc}(x,y) \mid X(x) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists x \varphi_1 \mid \exists X \varphi_1$,

where $\sigma \in \Sigma$.

An MSO *sentence* is a MSO formula without free variables.

Semantics.

Let φ be a MSO sentence.
The language defined by φ, denoted $\mathcal{L}(\varphi)$: The set of words satisfying φ.

A language $L \subseteq \Sigma^*$ is *MSO-definable*
if
there is a MSO sentence φ such that $\mathcal{L}(\varphi) = L$.

Monadic Second-Order Logic (continued)

Abbreviations.

- $\varphi_1 \wedge \varphi_2 = \neg(\neg\varphi_1 \vee \neg\varphi_2)$,
- $\varphi_1 \rightarrow \varphi_2 = \neg\varphi_1 \vee \varphi_2$,
- $\forall x \varphi_1 = \neg\exists x(\neg\varphi_1)$,
- $x < y = \begin{array}{c} \neg x = y \wedge \\ \forall X \left(\left(X(x) \wedge \forall z_1 \forall z_2 (X(z_1) \wedge \mathrm{suc}(z_1, z_2) \rightarrow X(z_2))\right) \rightarrow X(y) \right) \end{array}$,
- $\mathrm{first}(x) = \forall y(x = y \vee x < y)$,
- $\mathrm{last}(x) = \forall y(x = y \vee y < x)$.

Example.

$$\neg\exists x\ \mathrm{first}(x),$$

$$\exists x \exists y (P_a(x) \wedge P_b(y) \wedge x < y),$$

$$\exists X \left(\begin{array}{c} \exists x(\mathrm{first}(x) \wedge X(x)) \wedge \\ \forall x \forall y \forall z(\mathrm{suc}(x, y) \wedge \mathrm{suc}(y, z) \wedge X(x) \rightarrow X(z)) \\ \wedge \forall x(X(x) \rightarrow P_a(x)) \end{array} \right).$$

NFA≡MSO

From NFA to MSO

Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA. Let $Q = \{q_0, q_1, \ldots, q_n\}$. Construct the MSO formula φ as follows,

$$\exists q_0 \ldots q_n (\varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final}),$$

where

- $\varphi_{init} = \exists x(\text{first}(x) \wedge \bigvee_{(q_0,a,q)\in\delta} (P_a(x) \wedge q(x))),$
- $\varphi_{trans} = \forall x \forall y(\text{suc}(x, y) \to \bigvee_{(q,a,q')\in\delta} q(x) \wedge P_a(y) \wedge q'(y)),$
- $\varphi_{final} = \exists x(\text{last}(x) \wedge \bigvee_{q\in F} q(x)).$

Then $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A})$.

NFA≡MSO

From MSO to NFA.

A normal form for MSO formulas

New modalities,

$$X \subseteq Y,\ \mathrm{Sing}(X),\ suc(X, Y).$$

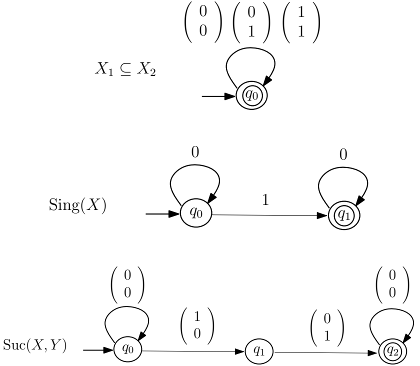
Then a MSO formula φ can be transformed into a normal form φ' by the following rules,

- if $\varphi = P_\sigma(x)$, then $\varphi' = \mathrm{Sing}(X) \wedge X \subseteq P_\sigma$,
- if $\varphi = x = y$, then $\varphi' = \mathrm{Sing}(X) \wedge \mathrm{Sing}(Y) \wedge X \subseteq Y \wedge Y \subseteq X$,
- if $\varphi = \mathrm{suc}(x, y)$, then $\varphi' = \mathrm{suc}(X, Y)$,
- if $\varphi = Z(x)$, then $\varphi' = \mathrm{Sing}(X) \wedge X \subseteq Z$,
- if $\varphi = \varphi_1 \vee \varphi_2$, then $\varphi' = \varphi_1' \vee \varphi_2'$,
- if $\varphi = \neg\varphi_1$, then $\varphi' = \neg\varphi_1'$,
- if $\varphi = \exists x \varphi_1$, then $\varphi' = \exists X(\mathrm{Sing}(X) \wedge \varphi_1')$,
- if $\varphi = \exists X \varphi_1$, then $\varphi' = \exists X \varphi_1'$.

From MSO to NFA.

$$\varphi := X \subseteq P_\sigma \mid P_\sigma \subseteq X \mid X \subseteq Y \mid \mathrm{Sing}(X) \mid \mathrm{suc}(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1.$$

Let $\varphi(X_1, \ldots, X_k)$ be a MSO formula in the normal form.
We construct a NFA $\mathcal{A} = (Q, \Sigma \times \{0,1\}^k, \delta, q_0, F)$ as follows.



NFA≡MSO

From MSO to NFA.

$$\varphi := X \subseteq P_\sigma \mid P_\sigma \subseteq X \mid X \subseteq Y \mid \mathrm{Sing}(X) \mid \mathrm{suc}(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1.$$

Let $\varphi(X_1, \dots, X_k)$ be a MSO formula in the normal form.
We construct a NFA $\mathcal{A} = (Q, \Sigma \times \{0,1\}^k, \delta, q_0, F)$ as follows.

- $\varphi = \varphi_1 \vee \varphi_2$

  NFAs are closed under union,

- $\varphi = \neg\varphi_1$

  NFAs are closed under complementation,

- $\varphi = \exists X \varphi_1$

  NFAs are closed under projection (a special case of homomorphisms), e.g. $(b_1, \dots, b_k) \to (b_2, \dots, b_k)$.

Outline

1. Visibly pushdown automata (VPA)
2. Closure properties
3. Visibly pushdown grammar (VPG)
4. Logical characterization
   - Equivalence of NFA and MSO
   - Equivalence of VPA and MSO_μ
5. Decision problems

MSO_μ

Fix $\tilde{\Sigma}$.

Given a word $w = a_1 \dots a_n \in \Sigma^*$, a binary relation $\mu(x, y)$ can be defined such that

$\mu(i,j)$ *iff* a_i *is a call and* a_j *is a matching return.*

Example. In the word "(( )) ( ) ((", $\mu(1,4), \mu(2,3), \mu(5,6)$ hold.

Syntax of MSO_μ over $\tilde{\Sigma}$.

$$\varphi := P_\sigma(x) \mid x = y \mid \mathrm{suc}(x,y) \mid X(x) \mid \mu(x,y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists x \varphi_1 \mid \exists X \varphi_1,$$

where $\sigma \in \Sigma$.

Semantics of MSO_μ over $\tilde{\Sigma}$.

- $(w, \mathcal{I}) \models \mu(x,y)$ iff $\mu(\mathcal{I}(x), \mathcal{I}(y))$ holds on w.

Example. Let $\tilde{\Sigma} = (\{a\}, \{b\}, \{c\})$

$$\forall x (P_a(x) \to \exists y \exists z (P_b(y) \wedge P_c(z) \wedge x < z \wedge z < y \wedge \mu(x,y)))$$

VPA $\equiv$ MSO$_\mu$

From VPA to MSO$_\mu$.

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA, $Q = \{q_0, \ldots, q_n\}$, $\Gamma = \{\gamma_1, \ldots, \gamma_k\}$.
Define $\varphi := \exists q_0 \ldots q_n P_{\gamma_1} \ldots P_{\gamma_k} (\varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final})$ as follows,

- $$\varphi_{init} = \exists x \left(\text{first}(x) \wedge \left(\begin{array}{c} \bigvee\limits_{(q_0,a,q)\in\delta} (P_a(x) \wedge q(x)) \bigvee \\ \bigvee\limits_{(q_0,a,q,\gamma)\in\delta} (P_a(x) \wedge q(x) \wedge P_\gamma(x)) \bigvee \\ \bigvee\limits_{(q_0,a,\perp,q)\in\delta} (P_a(x) \wedge q(x) \wedge P_\perp(x)) \end{array} \right)\right),$$
- $\varphi_{trans} = \forall x \forall y \, (\text{suc}(x, y) \rightarrow \psi_{call} \vee \psi_{return} \vee \psi_{local})$, where
  - $\psi_{call} = \bigvee\limits_{(q,a,q',\gamma)\in\delta} (q(x) \wedge P_a(y) \wedge q'(y) \wedge P_\gamma(y))$,
  - $$\psi_{return} = \begin{array}{c} \bigvee\limits_{(q,a,\gamma,q')\in\delta} (q(x) \wedge P_a(y) \wedge q'(y) \wedge P_\gamma(y) \wedge \exists z(\mu(z, y) \wedge P_\gamma(z))) \bigvee \\ \bigvee\limits_{(q,a,\perp,q')\in\delta} (q(x) \wedge P_a(y) \wedge q'(y) \wedge P_\perp(y) \wedge \neg\exists z(\mu(z, y))) \end{array},$$
  - $\psi_{local} = \bigvee\limits_{(q,a,q')\in\delta} (q(x) \wedge P_a(y) \wedge q'(y))$.
- $$\varphi_{final} = \exists x \left(\text{last}(x) \wedge \bigvee_{q\in F} q(x) \right).$$

From MSO$_\mu$ to VPA.

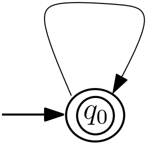
$$\varphi := \begin{array}{c} X \subseteq P_\sigma \mid P_\sigma \subseteq X \mid X \subseteq Y \mid \mathrm{Sing}(X) \mid \\ \mathrm{suc}(X,Y) \mid \mu(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1 \end{array},$$

$$\sigma', \sigma'' \neq \sigma : \quad (\sigma',0), \downarrow (\sigma',0) \;\; (\sigma',0), \uparrow (\sigma'',0), (\sigma,1), \perp$$

$$(\sigma',0)$$

$$(\sigma,1) \downarrow (\sigma,1) \;\; (\sigma,0), \downarrow (\sigma,0)$$

$X \subseteq P_\sigma : \sigma \in \Sigma_c$



VPA $\equiv$ MSO$_\mu$: continued

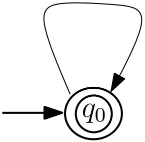
From MSO$_\mu$ to VPA.

$$\varphi := \begin{array}{c} X \subseteq P_\sigma \mid P_\sigma \subseteq X \mid X \subseteq Y \mid \mathrm{Sing}(X) \mid \\ \mathrm{suc}(X,Y) \mid \mu(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1 \end{array},$$

$\sigma', \sigma'' \neq \sigma :$

$$(\sigma',0), \downarrow (\sigma',0) \quad (\sigma',0), \uparrow (\sigma'',0), \perp$$
$$(\sigma',0)$$
$$(\sigma,1) \uparrow (\sigma',0), \perp \quad (\sigma,0) \uparrow (\sigma',0), \perp$$

$X \subseteq P_\sigma : \sigma \in \Sigma_r$

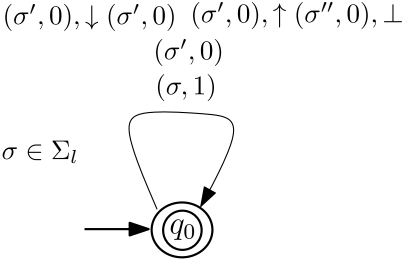


From MSO$_\mu$ to VPA.

$$\varphi := \begin{array}{c} X \subseteq P_\sigma \mid P_\sigma \subseteq X \mid X \subseteq Y \mid \mathrm{Sing}(X) \mid \\ \mathrm{suc}(X,Y) \mid \mu(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1 \end{array},$$

$\sigma', \sigma'' \neq \sigma$:

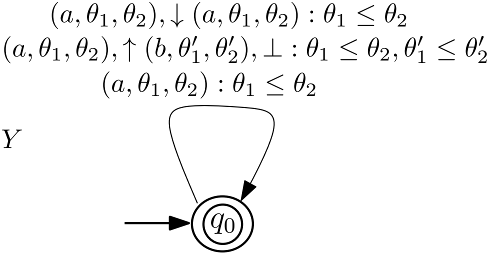
$X \subseteq P_\sigma : \sigma \in \Sigma_l$



From MSO$_\mu$ to VPA.

$$\varphi := \begin{array}{c} X \subseteq P_\sigma \mid P_\sigma \subseteq X \mid X \subseteq Y \mid \mathrm{Sing}(X) \mid \\ \mathrm{suc}(X,Y) \mid \mu(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1 \end{array},$$

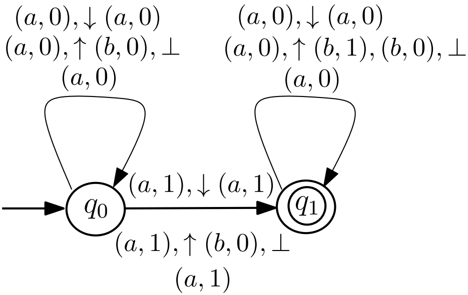
$X \subseteq Y$



From MSO$_\mu$ to VPA.

$$\varphi := \begin{array}{c} X \subseteq P_\sigma \mid P_\sigma \subseteq X \mid X \subseteq Y \mid \mathrm{Sing}(X) \mid \\ \mathrm{suc}(X,Y) \mid \mu(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1 \end{array},$$

Sing(X)

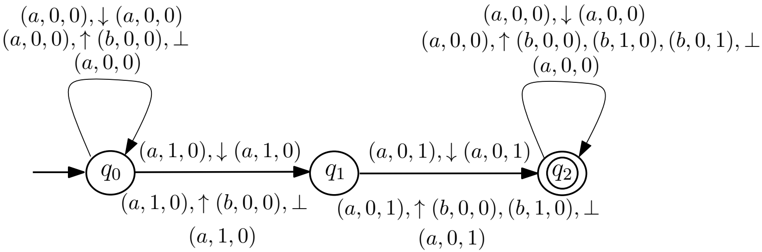


VPA $\equiv$ MSO$_\mu$: continued

From MSO$_\mu$ to VPA.

$$\varphi := \begin{array}{c} X \subseteq P_\sigma \mid P_\sigma \subseteq X \mid X \subseteq Y \mid \mathrm{Sing}(X) \mid \\ \mathrm{suc}(X,Y) \mid \mu(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1 \end{array},$$

suc(X,Y)

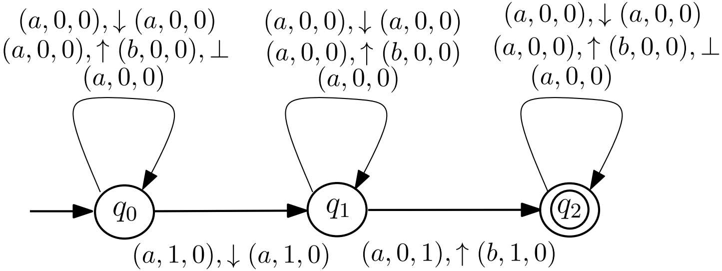


VPA $\equiv$ MSO$_\mu$: continued

From MSO$_\mu$ to VPA.

$$\varphi := \begin{array}{c} X \subseteq P_\sigma \mid P_\sigma \subseteq X \mid X \subseteq Y \mid \mathrm{Sing}(X) \mid \\ \mathrm{suc}(X,Y) \mid \mu(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1 \end{array},$$

$\mu(X,Y)$



Outline

1. Visibly pushdown automata (VPA)
2. Closure properties
3. Visibly pushdown grammar (VPG)
4. Logical characterization
   - Equivalence of NFA and MSO
   - Equivalence of VPA and MSO_μ
5. Decision problems

Nonemptiness

Theorem. The nonemptiness of VPA can be solved in $O(n^3)$ time.

A VPA can be transformed into an equivalent VPG in $O(n^3)$ *time.*

The emptiness of a CFG can be solved in linear time.

Language inclusion

Theorem. The language inclusion of VPA is EXPTIME-complete.

Upper bound.

Given two VPAs $\mathcal{A}_1$ and $\mathcal{A}_2$,

- determinize $\mathcal{A}_2$ into $\mathcal{A}_2'$,
- complement $\mathcal{A}_2'$ into $\mathcal{B}$,
- test whether $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{B}) = \emptyset$.

The determinization procedure can be fulfilled in EXPTIME.

Language inclusion

Theorem. The language inclusion of VPA is EXPTIME-complete.

Lower bound.

The universality of VPA is EXPTIME-hard.

Language inclusion

Theorem. The language inclusion of VPA is EXPTIME-complete.

Lower bound.

The universality of VPA is EXPTIME-hard.

Result from complexity theory: APSPACE = EXPTIME.

An *alternating* TM (ATM) is a TM $M = (Q_\vee, Q_\wedge, \Sigma, \Gamma, \delta, q_0, B, F)$ such that

- the state set is divided into two disjoint subsets, $Q_\vee$ ("or" state), $Q_\wedge$ ("and" state),
- for every $q \in Q$ and $a \in \Gamma$, $|\delta(q, a)| = 2$.

A *run* of an ATM M over an input $w \in \Sigma^*$ is a configuration tree s.t.

- the root of the tree is the initial configuration,
- for every node (configuration) $\alpha q \beta$ in the tree, if $q \in Q_\vee$, then $\alpha q \beta$ has one of its successor config. as its unique child in the tree,
- for every node (configuration) $\alpha q \beta$ in the tree, if $q \in Q_\wedge$, then the two successor config. of $\alpha q \beta$ are both its children in the tree.

APSPACE: The class of languages accepted by ATMs using polynomial space.

Language inclusion

Theorem. The language inclusion of VPA is EXPTIME-complete.

Lower bound.

The universality of VPA is EXPTIME-hard.

Reduction from

the membership problem of alternating TMs using polynomial space.

Let $M = (Q_\vee, Q_\wedge, \Sigma, \Gamma, \delta, q_0, B, F)$ be a ATM using linear space, say cn.
Let t be a accepting run of M over an input w.
Use C_x*'s (where* $x \in \{0,1\}^*$*) to denote the nodes of* t*,*
e.g. the root is C_ε, while the left child of the root is C_0, and so on.
Encode t by a word θ which is generated by a DFS traversal of t.

Language inclusion

Theorem. The language inclusion of VPA is EXPTIME-complete.

Lower bound.

The universality of VPA is EXPTIME-hard.

Reduction from

the membership problem of alternating TMs using polynomial space.

Let $M = (Q_\vee, Q_\wedge, \Sigma, \Gamma, \delta, q_0, B, F)$ be a ATM using linear space, say cn.
Let t be a accepting run of M over an input w.
Use C_x*'s (where* $x \in \{0,1\}^*$*) to denote the nodes of* t*,*
e.g. the root is C_ε, while the left child of the root is C_0, and so on.
Encode t by a word θ which is generated by a DFS traversal of t.
Initially set $\theta = \varepsilon$.

1. The traversal starts from the root C_ε.
2. When a node C_x is visited for the first time, then $\theta = \theta(fC_x)$,
3. When a node C_x is visited again by backtracking from its right-child, then $\theta = \theta(b\overline{C_x})^r$.

Language inclusion

Theorem. The language inclusion of VPA is EXPTIME-complete.

Lower bound.

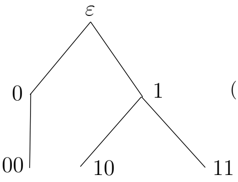
The universality of VPA is EXPTIME-hard.

Reduction from

the membership problem of alternating TMs using polynomial space.

Let $M = (Q_\vee, Q_\wedge, \Sigma, \Gamma, \delta, q_0, B, F)$ be a ATM using linear space, say cn.
Let t be a accepting run of M over an input w.
Use C_x*'s (where* $x \in \{0,1\}^*$*) to denote the nodes of* t*,*
e.g. the root is C_ε, while the left child of the root is C_0, and so on.
Encode t by a word θ which is generated by a DFS traversal of t.

Such a word θ *is called a successful computation of* M*.*



$$(fC_\varepsilon)(fC_0)(fC_{00})(b\overline{C_{00}})^r(b\overline{C_0})^r(fC_1)(fC_{10})(b\overline{C_{10}})^r(fC_{11})(b\overline{C_{11}})^r(b\overline{C_1})^r(b\overline{C_\varepsilon})^r$$

Language inclusion

Theorem. The language inclusion of VPA is EXPTIME-complete.

Lower bound.

The universality of VPA is EXPTIME-hard.

Reduction from

the membership problem of alternating TMs using polynomial space.

Let $M = (Q_\vee, Q_\wedge, \Sigma, \Gamma, \delta, q_0, B, F)$ be a ATM using linear space, say cn.
Let $\Gamma' = \Gamma \cup Q \cup \bar{\Gamma} \cup \bar{Q} \cup \{f, b\}$, $\tilde{\Gamma}' = \langle \Gamma \cup Q \cup \{f\}, \bar{\Gamma} \cup \bar{Q} \cup \{b\} \rangle$.
The format of a successful computation θ,
e.g. well-matched call-returns, except consistencies of consecutive config.
can be checked by a deterministic VPA $\mathcal{A}$.

A word $w \in (\Gamma')^*$ is a *unsuccessful* computation of M if one of the following conditions holds,

- w is not accepted by $\mathcal{A}$,
- there is a subword $fC_x fC_{x0}$ or $\overline{C^r_{x0}} b \overline{C^r_x} b$ or $\overline{C^r_{x1}} b \overline{C^r_x} b$, such that $C_x \not\vdash C_{x0}$, or $C_x \not\vdash C_{x1}$: Guess an index $i : 1 < i < cn + 1$, and check the relationship of the $(i-1, i, i+1)$-th symbol of C_x and the i-th symbol of C_{x0}, ...

Language inclusion

Theorem. The language inclusion of VPA is EXPTIME-complete.

Lower bound.

The universality of VPA is EXPTIME-hard.

Reduction from

the membership problem of alternating TMs using polynomial space.

Let $M = (Q_\vee, Q_\wedge, \Sigma, \Gamma, \delta, q_0, B, F)$ be a ATM using linear space, say cn.
Let $\Gamma' = \Gamma \cup Q \cup \bar{\Gamma} \cup \bar{Q} \cup \{f, b\}$, $\widetilde{\Gamma'} = \langle \Gamma \cup Q \cup \{f\}, \bar{\Gamma} \cup \bar{Q} \cup \{b\} \rangle$.

The set of unsuccessful computations of M

can be accepted by a nondeterministic VPA $\mathcal{B}$ of polynomial size.

$$M \textit{ does not accept } w \textit{ iff } \mathcal{L}(\mathcal{B}) = (\Gamma')^*.$$

Automata over infinite words