CS244: Theory of Computation

Fu Song
ShanghaiTech University

Fall 2022

Outline

Visibly pushdown automata (VPA)

Closure properties

Visibly pushdown grammar (VPG)

Logical characterization
- Equivalence of NFA and MSO
- Equivalence of VPA and MSO_μ

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

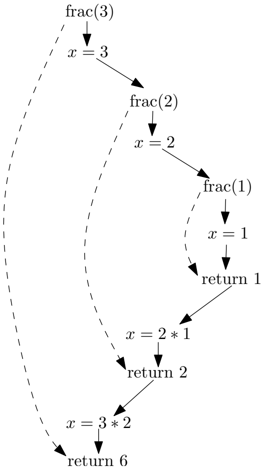
XML documents

⟨library⟩
⟨catalog⟩
⟨book⟩
⟨title⟩
Computational Complexity
⟨/title⟩
⟨author⟩
Fu Song
⟨/author⟩
⟨book⟩
⟨catalog⟩
⟨library⟩

Motivation

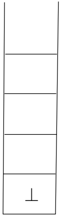
Recursive function calls and returns

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



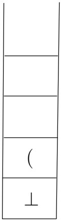
Motivation

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



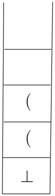
Motivation

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



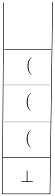
Motivation

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



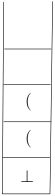
Motivation

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



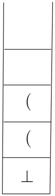
Motivation

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



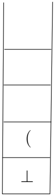
Motivation

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



Motivation

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



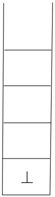
Motivation

Stack operations determined by the input symbol

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

(: Push

): Pop



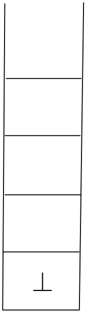
Motivation

Stack operations determined by the input symbol

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

{: Push

}: Pop



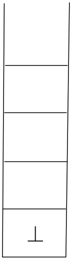
Motivation

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$\rangle$

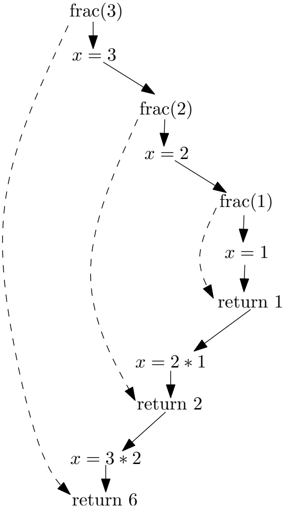
$\langle tag \rangle$: Push

$\langle / tag \rangle$: Pop

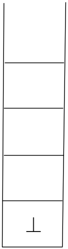


Motivation

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.

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 7-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,
- $\delta \subseteq Q \times \Sigma_c \times Q \times (\Gamma \setminus \{\perp\}) \;\cup\; Q \times \Sigma_r \times \Gamma \times Q \;\cup\; Q \times \Sigma_l \times Q$,
- $q_0 \in Q$ is the initial state,
- $\perp$ 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 $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$

- 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 7-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,
- $\delta \subseteq Q \times \Sigma_c \times Q \times (\Gamma \setminus \{\bot\}) \;\cup\; Q \times \Sigma_r \times \Gamma \times Q \;\cup\; Q \times \Sigma_l \times Q$,
- $q_0 \in Q$ 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 deterministic VPA is a VPA $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ such that

Call: for every $(q, a) \in Q \times \Sigma_c$, there is at most one pair $(q', \gamma) \in Q \times (\Gamma \setminus \{\perp\})$ such that $(q, a, q', \gamma) \in \delta$,

Visibly pushdown automata (VPA)

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

Call: for every $(q, a) \in Q \times \Sigma_c$, there is at most one pair $(q', \gamma) \in Q \times (\Gamma \setminus \{\bot\})$ such that $(q, a, q', \gamma) \in \delta$,

Return: for every $(q, a, \gamma) \in Q \times \Sigma_r \times \Gamma$, there is at most one $q' \in Q$ such that $(q, a, \gamma, q') \in \delta$,

Visibly pushdown automata (VPA)

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

Call: for every $(q, a) \in Q \times \Sigma_c$, there is at most one pair $(q', \gamma) \in Q \times (\Gamma \setminus \{\perp\})$ such that $(q, a, q', \gamma) \in \delta$,

Return: for every $(q, a, \gamma) \in Q \times \Sigma_r \times \Gamma$, there is at most one $q' \in Q$ such that $(q, a, \gamma, q') \in \delta$,

Local: for every $(q, a) \in Q \times \Sigma_l$, there is at most one $q' \in Q$ such that $(q, a, q') \in \delta$.

Visibly pushdown automata (VPA)

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

Call: for every $(q, a) \in Q \times \Sigma_c$, there is at most one pair $(q', \gamma) \in Q \times (\Gamma \setminus \{\perp\})$ such that $(q, a, q', \gamma) \in \delta$,

Return: for every $(q, a, \gamma) \in Q \times \Sigma_r \times \Gamma$, there is at most one $q' \in Q$ such that $(q, a, \gamma, q') \in \delta$,

Local: 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 $(q_0, \alpha_0)(q_1, \alpha_1) \dots (q_n, \alpha_n)$ *of configurations*

Visibly pushdown automata (VPA): continued

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

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

- $\forall i . q_i \in Q$,

Visibly pushdown automata (VPA): continued

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

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

- $\forall i . q_i \in Q$,
- $\alpha_0 = \perp$,

Visibly pushdown automata (VPA): continued

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

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

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

Visibly pushdown automata (VPA): continued

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

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

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

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

Visibly pushdown automata (VPA): continued

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

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

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

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

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

Visibly pushdown automata (VPA): continued

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

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

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

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

Return: $a_i \in \Sigma_r$,

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

Visibly pushdown automata (VPA): continued

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

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

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

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

- $\exists \gamma \in \Gamma \setminus \{\perp\}.(q_i, a_i, \gamma, q_{i+1}) \in \delta$, $\alpha_i = \gamma \alpha_{i+1}$,
- 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 $w = a_1 \dots a_n$ is

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

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

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

Return: $a_i \in \Sigma_r$,

- $\exists \gamma \in \Gamma \setminus \{\perp\}.(q_i, a_i, \gamma, q_{i+1}) \in \delta$, $\alpha_i = \gamma\alpha_{i+1}$,
- 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 $(q_0, \alpha_0) \dots (q_n, \alpha_n)$ is accepting if $q_n \in F$.

Visibly pushdown automata (VPA): continued

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

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

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

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

Return: $a_i \in \Sigma_r$,
- $\exists \gamma \in \Gamma \setminus \{\perp\}.(q_i, a_i, \gamma, q_{i+1}) \in \delta$, $\alpha_i = \gamma\alpha_{i+1}$,
- 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 $(q_0, \alpha_0) \dots (q_n, \alpha_n)$ 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.

Visibly pushdown automata (VPA): continued

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

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

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

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

Return: $a_i \in \Sigma_r$,

- $\exists \gamma \in \Gamma \setminus \{\perp\}.(q_i, a_i, \gamma, q_{i+1}) \in \delta$, $\alpha_i = \gamma\alpha_{i+1}$,
- 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 $(q_0, \alpha_0) \dots (q_n, \alpha_n)$ 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 $\mathcal{L}(\mathcal{A})$.

Visibly pushdown automata (VPA): continued

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

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

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

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

Return: $a_i \in \Sigma_r$,

- $\exists \gamma \in \Gamma \setminus \{\perp\}. (q_i, a_i, \gamma, q_{i+1}) \in \delta$, $\alpha_i = \gamma \alpha_{i+1}$,
- 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 $(q_0, \alpha_0) \dots (q_n, \alpha_n)$ 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,

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,

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

  $w = aw'$ or $w = w'a$ such that $a \in \Sigma_l$ is well-matched.

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
  $w = aw'$ *or* $w = w'a$ *such that* $a \in \Sigma_l$ *is well-matched.*
- if w' is well-matched, then
  $w = aw'b$ *such that* $a \in \Sigma_c, b \in \Sigma_r$ *is well-matched.*

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

  $w = aw'$ *or* $w = w'a$ *such that* $a \in \Sigma_l$ *is well-matched.*
- if w' is well-matched, then

  $w = aw'b$ *such that* $a \in \Sigma_c, b \in \Sigma_r$ *is well-matched.*
- if w' and w'' are well-matched, then

  $w = w'w''$ *is well-matched.*

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

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

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
  $w = aw'$ *or* $w = w'a$ *such that* $a \in \Sigma_l$ *is well-matched.*
- if w' is well-matched, then
  $w = aw'b$ *such that* $a \in \Sigma_c, b \in \Sigma_r$ *is well-matched.*
- if w' and w'' are well-matched, then
  $w = w'w''$ *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* $\mathcal{L}(\mathcal{A}) = L$.

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 $\{a^n b^n \mid n \geq 1\}$ *is a VPL with respect to* $\widetilde{\Sigma} = \langle \{a\}, \{b\}, \emptyset \rangle$.

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 $\{a^n b^n \mid n \geq 1\}$ *is a VPL with respect to* $\widetilde{\Sigma} = \langle \{a\}, \{b\}, \emptyset \rangle$*. But,* $\{b^n a^n \mid n \geq 1\}$ *is not a VPL with respect to* $\widetilde{\Sigma} = \langle \{a\}, \{b\}, \emptyset \rangle$*.*

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 $\{a^n b^n \mid n \geq 1\}$ is a VPL with respect to $\widetilde{\Sigma} = \langle \{a\}, \{b\}, \emptyset \rangle$. But, $\{b^n a^n \mid n \geq 1\}$ is not a VPL with respect to $\widetilde{\Sigma} = \langle \{a\}, \{b\}, \emptyset \rangle$.

Theorem

VPL $\subsetneq$ CFL.

Embedding of CFL as VPLs

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

Embedding of CFL as VPLs

Proposition. For every CFL $L \subseteq \Sigma^*$, there are a VPL $L' \subseteq (\Sigma')^*$ with respect to some $\widetilde{\Sigma'}$ 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).

Embedding of CFL as VPLs

Proposition. For every CFL $L \subseteq \Sigma^*$, there are a VPL $L' \subseteq (\Sigma')^*$ with respect to some $\widetilde{\Sigma'}$ 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 $(q, a, X, q', \alpha) \in \delta$ satisfies that

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

Embedding of CFL as VPLs

Proposition. For every CFL $L \subseteq \Sigma^*$, there are a VPL $L' \subseteq (\Sigma')^*$ with respect to some $\widetilde{\Sigma'}$ 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 $(q, a, X, q', \alpha) \in \delta$ satisfies that

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

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

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

Embedding of CFL as VPLs

Proposition. For every CFL $L \subseteq \Sigma^*$, there are a VPL $L' \subseteq (\Sigma')^*$ with respect to some $\widetilde{\Sigma'}$ 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 $(q, a, X, q', \alpha) \in \delta$ satisfies that

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

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

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

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

Embedding of CFL as VPLs

Proposition. For every CFL $L \subseteq \Sigma^*$, there are a VPL $L' \subseteq (\Sigma')^*$ with respect to some $\widetilde{\Sigma'}$ 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 $(q, a, X, q', \alpha) \in \delta$ satisfies that

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

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

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

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

- if $(q, a, X, q', \varepsilon) \in \delta$, then $[Return](q, (a, r), X, q') \in \delta'$,

Embedding of CFL as VPLs

Proposition. For every CFL $L \subseteq \Sigma^*$, there are a VPL $L' \subseteq (\Sigma')^*$ with respect to some $\widetilde{\Sigma'}$ 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 $(q, a, X, q', \alpha) \in \delta$ satisfies that

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

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

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

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

- if $(q, a, X, q', \varepsilon) \in \delta$, then $[Return](q, (a, r), X, q') \in \delta'$,
- if $(q, a, X, q', X) \in \delta$, then add a new state q_1,
  $[Return](q, (a, r), X, q_1), [Call](q_1, (\sigma_\varepsilon, c), q', X) \in \delta'$.

Embedding of CFL as VPLs

Proposition. For every CFL $L \subseteq \Sigma^*$, there are a VPL $L' \subseteq (\Sigma')^*$ with respect to some $\widetilde{\Sigma'}$ and a homomorphism $h : (\Sigma')^* \rightarrow \Sigma^*$ 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 $(q, a, X, q', \alpha) \in \delta$ satisfies that

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

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

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

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

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

Embedding of CFL as VPLs

Proposition. For every CFL $L \subseteq \Sigma^*$, there are a VPL $L' \subseteq (\Sigma')^*$ with respect to some $\widetilde{\Sigma'}$ 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 $(q, a, X, q', \alpha) \in \delta$ satisfies that

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

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

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

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

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

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

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

Embedding of CFL as VPLs

Proposition. For every CFL $L \subseteq \Sigma^*$, there are a VPL $L' \subseteq (\Sigma')^*$ with respect to some $\widetilde{\Sigma'}$ 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 $(q, a, X, q', \alpha) \in \delta$ satisfies that

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

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

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

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

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

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

Visibly pushdown automata (VPA)

Closure properties

Visibly pushdown grammar (VPG)

Logical characterization

Equivalence of NFA and MSO

Equivalence of VPA and MSO_μ

Decision problems

Union and intersection

Proposition. VPLs with respect to $\widetilde{\Sigma}$ are closed under union and intersection.

Union and intersection

Proposition. VPLs with respect to $\widetilde{\Sigma}$ are closed under union and intersection.
Let $\mathcal{A}_1 = (Q_1, \widetilde{\Sigma}, \Gamma_1, \delta_1, q_0^1, \perp_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \widetilde{\Sigma}, \Gamma_2, \delta_2, q_0^2, \perp_2, F_2)$ be two VPAs.

Union and intersection

Proposition. VPLs with respect to $\widetilde{\Sigma}$ are closed under union and intersection.
Let $\mathcal{A}_1 = (Q_1, \widetilde{\Sigma}, \Gamma_1, \delta_1, q_0^1, \perp_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \widetilde{\Sigma}, \Gamma_2, \delta_2, q_0^2, \perp_2, F_2)$ be two VPAs.

Union.

Without loss of generality, suppose $\perp_1 = \perp_2 = \perp$.

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

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

Union and intersection

Proposition. VPLs with respect to $\widetilde{\Sigma}$ are closed under union and intersection. Let $\mathcal{A}_1 = (Q_1, \widetilde{\Sigma}, \Gamma_1, \delta_1, q_0^1, \perp_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \widetilde{\Sigma}, \Gamma_2, \delta_2, q_0^2, \perp_2, F_2)$ be two VPAs.

Union.
Without loss of generality, suppose $\perp_1 = \perp_2 = \perp$.
The VPA $\mathcal{A} = (Q_1 \cup Q_2 \cup \{q_0\}, \widetilde{\Sigma}, \Gamma_1 \cup \Gamma_2, \delta, q_0, \perp, F_1 \cup F_2)$ such that

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

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

Union and intersection

Proposition. VPLs with respect to $\widetilde{\Sigma}$ are closed under union and intersection.
Let $\mathcal{A}_1 = (Q_1, \widetilde{\Sigma}, \Gamma_1, \delta_1, q_0^1, \perp_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \widetilde{\Sigma}, \Gamma_2, \delta_2, q_0^2, \perp_2, F_2)$ be two VPAs.

Union.
Without loss of generality, suppose $\perp_1 = \perp_2 = \perp$.
The VPA $\mathcal{A} = (Q_1 \cup Q_2 \cup \{q_0\}, \widetilde{\Sigma}, \Gamma_1 \cup \Gamma_2, \delta, q_0, \perp, F_1 \cup F_2)$ such that

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

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

Intersection. Using the fact that $\mathcal{A}_1$ and $\mathcal{A}_2$, being VPAs, synchronize on the push and pop operations on the stack.

The VPA $\mathcal{A} = (Q_1 \times Q_2, \widetilde{\Sigma}, \Gamma_1 \times \Gamma_2, \delta, (q_0^1, q_0^2), (\perp_1, \perp_2), F_1 \times F_2)$ such that

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

Union and intersection

Proposition. VPLs with respect to $\widetilde{\Sigma}$ are closed under union and intersection.
Let $\mathcal{A}_1 = (Q_1, \widetilde{\Sigma}, \Gamma_1, \delta_1, q_0^1, \perp_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \widetilde{\Sigma}, \Gamma_2, \delta_2, q_0^2, \perp_2, F_2)$ be two VPAs.

Union.
Without loss of generality, suppose $\perp_1 = \perp_2 = \perp$.
The VPA $\mathcal{A} = (Q_1 \cup Q_2 \cup \{q_0\}, \widetilde{\Sigma}, \Gamma_1 \cup \Gamma_2, \delta, q_0, \perp, F_1 \cup F_2)$ such that

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

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

Intersection. Using the fact that $\mathcal{A}_1$ and $\mathcal{A}_2$, being VPAs, synchronize on the push and pop operations on the stack.

The VPA $\mathcal{A} = (Q_1 \times Q_2, \widetilde{\Sigma}, \Gamma_1 \times \Gamma_2, \delta, (q_0^1, q_0^2), (\perp_1, \perp_2), F_1 \times F_2)$ such that

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

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

Note that CFL are not closed under intersection

Determinization

For every VPA $\mathcal{A}$, can we construct a deterministic VPA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$?

Determinization

For every VPA $\mathcal{A}$, can we construct a deterministic VPA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$?

Theorem. For every VPA $\mathcal{A}$, a deterministic VPA $\mathcal{A}'$ can be constructed such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. Moreover, if $\mathcal{A}$ has n states, we can construct $\mathcal{A}'$ with $O(2^{n^2})$ states and with stack alphabet of size $O(2^{n^2} \cdot |\Sigma_c|)$.

Determinization

For every VPA $\mathcal{A}$, can we construct a deterministic VPA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$?

Theorem. For every VPA $\mathcal{A}$, a deterministic VPA $\mathcal{A}'$ can be constructed such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. Moreover, if $\mathcal{A}$ has n states, we can construct $\mathcal{A}'$ with $O(2^{n^2})$ states and with stack alphabet of size $O(2^{n^2} \cdot |\Sigma_c|)$.

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

Determinization

For every VPA $\mathcal{A}$, can we construct a deterministic VPA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$?

Theorem. For every VPA $\mathcal{A}$, a deterministic VPA $\mathcal{A}'$ can be constructed such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. Moreover, if $\mathcal{A}$ has n states, we can construct $\mathcal{A}'$ with $O(2^{n^2})$ states and with stack alphabet of size $O(2^{n^2} \cdot |\Sigma_c|)$.

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Do a subset construction but postpone handling the push-transitions that $\mathcal{A}$ does.

Determinization

For every VPA $\mathcal{A}$, can we construct a deterministic VPA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$?

Theorem. For every VPA $\mathcal{A}$, a deterministic VPA $\mathcal{A}'$ can be constructed such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. Moreover, if $\mathcal{A}$ has n states, we can construct $\mathcal{A}'$ with $O(2^{n^2})$ states and with stack alphabet of size $O(2^{n^2} \cdot |\Sigma_c|)$.

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Do a subset construction but postpone handling the push-transitions that $\mathcal{A}$ does.
- Instead, we store the call actions and simulate the push-transitions corresponding to them later, namely at the time of the corresponding pop-transition.

Determinization

For every VPA $\mathcal{A}$, can we construct a deterministic VPA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$?

Theorem. For every VPA $\mathcal{A}$, a deterministic VPA $\mathcal{A}'$ can be constructed such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. Moreover, if $\mathcal{A}$ has n states, we can construct $\mathcal{A}'$ with $O(2^{n^2})$ states and with stack alphabet of size $O(2^{n^2} \cdot |\Sigma_c|)$.

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Do a subset construction but postpone handling the push-transitions that $\mathcal{A}$ does.
- Instead, we store the call actions and simulate the push-transitions corresponding to them later, namely at the time of the corresponding pop-transition.
- The construction will have a component S that is a set of "summary" edges that keeps track of what state transitions are possible from a push-transition to the corresponding pop-transition.

Determinization

For every VPA $\mathcal{A}$, can we construct a deterministic VPA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$?

Theorem. For every VPA $\mathcal{A}$, a deterministic VPA $\mathcal{A}'$ can be constructed such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. Moreover, if $\mathcal{A}$ has n states, we can construct $\mathcal{A}'$ with $O(2^{n^2})$ states and with stack alphabet of size $O(2^{n^2} \cdot |\Sigma_c|)$.

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Do a subset construction but postpone handling the push-transitions that $\mathcal{A}$ does.
- Instead, we store the call actions and simulate the push-transitions corresponding to them later, namely at the time of the corresponding pop-transition.
- The construction will have a component S that is a set of "summary" edges that keeps track of what state transitions are possible from a push-transition to the corresponding pop-transition.
- Using the summary information, the set of reachable states is updated.

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \bot, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \bot, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that
  - $a_1, a_2 \in \Sigma_c$

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that
  - $a_1, a_2 \in \Sigma_c$
  - $x \in \Sigma^*$ such that every call in x is matched by a return, but there may be unmatched returns

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that
  - $a_1, a_2 \in \Sigma_c$
  - $x \in \Sigma^*$ such that every call in x is matched by a return, but there may be unmatched returns
  - $y, z \in \Sigma^*$ such that all calls and returns are matched

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \bot, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \bot, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that
  - $a_1, a_2 \in \Sigma_c$
  - $x \in \Sigma^*$ such that every call in x is matched by a return, but there may be unmatched returns
  - $y, z \in \Sigma^*$ such that all calls and returns are matched
- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot\Big)$$

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q,q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that
  - $a_1, a_2 \in \Sigma_c$
  - $x \in \Sigma^*$ such that every call in x is matched by a return, but there may be unmatched returns
  - $y, z \in \Sigma^*$ such that all calls and returns are matched
- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\perp\Big)$$

  - $S_1 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{x} (q', \alpha)$ in $\mathcal{A}$

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \bot, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \bot, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that
  - $a_1, a_2 \in \Sigma_c$
  - $x \in \Sigma^*$ such that every call in x is matched by a return, but there may be unmatched returns
  - $y, z \in \Sigma^*$ such that all calls and returns are matched
- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot\Big)$$

  - $S_1 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{x} (q', \alpha)$ in $\mathcal{A}$
  - $S_2 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{y} (q', \alpha)$ in $\mathcal{A}$

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q,q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that
  - $a_1, a_2 \in \Sigma_c$
  - $x \in \Sigma^*$ such that every call in x is matched by a return, but there may be unmatched returns
  - $y, z \in \Sigma^*$ such that all calls and returns are matched
- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S,R), (S_2, R_2, a_2)(S_1, R_1, a_1)\perp\Big)$$

  - $S_1 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{x} (q', \alpha)$ in $\mathcal{A}$
  - $S_2 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{y} (q', \alpha)$ in $\mathcal{A}$
  - $S \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{z} (q', \alpha)$ in $\mathcal{A}$

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that
  - $a_1, a_2 \in \Sigma_c$
  - $x \in \Sigma^*$ such that every call in x is matched by a return, but there may be unmatched returns
  - $y, z \in \Sigma^*$ such that all calls and returns are matched
- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\perp\Big)$$

  - $S_1 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{x} (q', \alpha)$ in $\mathcal{A}$
  - $S_2 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{y} (q', \alpha)$ in $\mathcal{A}$
  - $S \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{z} (q', \alpha)$ in $\mathcal{A}$
  - R_1 is the set of states reachable by $\mathcal{A}$ from any initial state on x

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \bot, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \bot, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that
  - $a_1, a_2 \in \Sigma_c$
  - $x \in \Sigma^*$ such that every call in x is matched by a return, but there may be unmatched returns
  - $y, z \in \Sigma^*$ such that all calls and returns are matched
- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot\Big)$$

  - $S_1 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{x} (q', \alpha)$ in $\mathcal{A}$
  - $S_2 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{y} (q', \alpha)$ in $\mathcal{A}$
  - $S \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{z} (q', \alpha)$ in $\mathcal{A}$
  - R_1 is the set of states reachable by $\mathcal{A}$ from any initial state on x
  - R_2 is the set of states reachable by $\mathcal{A}$ from any initial state on xa_1y

Determinization

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA. we construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), \perp, F')$, where $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$

- Consider an input string $w = xa_1ya_2z$ such that
  - $a_1, a_2 \in \Sigma_c$
  - $x \in \Sigma^*$ such that every call in x is matched by a return, but there may be unmatched returns
  - $y, z \in \Sigma^*$ such that all calls and returns are matched
- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\perp\Big)$$

  - $S_1 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{x} (q', \alpha)$ in $\mathcal{A}$
  - $S_2 \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{y} (q', \alpha)$ in $\mathcal{A}$
  - $S \subseteq Q \times Q$ is the summary for all pairs of states (q, q') such that we have $(q, \alpha) \xrightarrow{z} (q', \alpha)$ in $\mathcal{A}$
  - R_1 is the set of states reachable by $\mathcal{A}$ from any initial state on x
  - R_2 is the set of states reachable by $\mathcal{A}$ from any initial state on xa_1y
  - R is the set of states reachable by $\mathcal{A}$ from any initial state on xa_1ya_2z

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 $\alpha = \alpha'$.

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.

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
  $$S_w = \{(q, q') \mid \exists q_1, q_2, \gamma.\ (q, a, q_1, \gamma) \in \delta, (q_1, q_2) \in S_{w'}, (q_2, b, \gamma, q') \in \delta\}.$$

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 $\alpha = \alpha'$.

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 III. Get inspirations from the subset construction for NFAs.

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 III. Get inspirations from the subset construction for NFAs.

Question:

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

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 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.

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 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 . . .

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 III. Get inspirations from the subset construction for NFAs.

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.

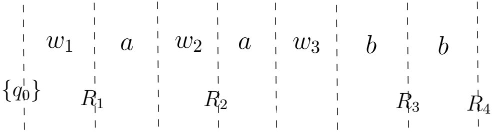


Illustration of the intuition of the proof of the Theorem

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 III. Get inspirations from the subset construction for NFAs.

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

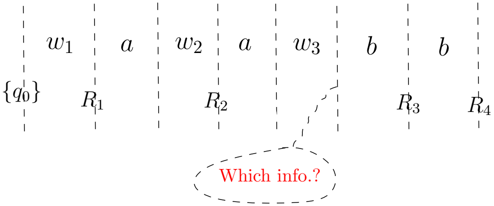


Illustration of the intuition of the proof of the Theorem

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 III. Get inspirations from the subset construction for NFAs.

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.

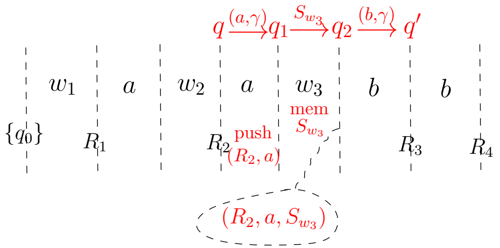


Illustration of the intuition of the proof of the Theorem

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 III. Get inspirations from the subset construction for NFAs.

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.

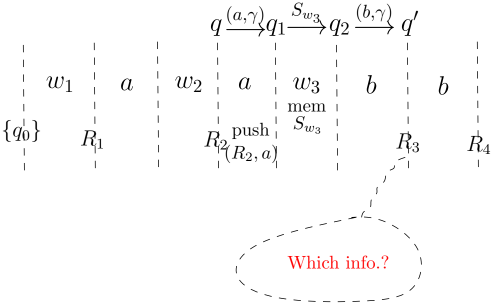


Illustration of the intuition of the proof of the Theorem

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 III. Get inspirations from the subset construction for NFAs.

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.

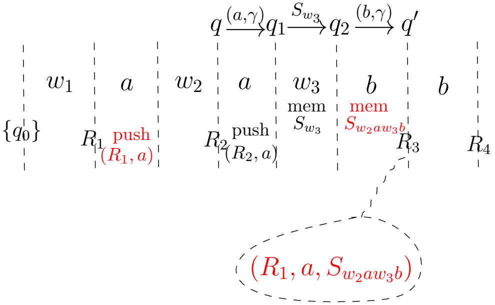


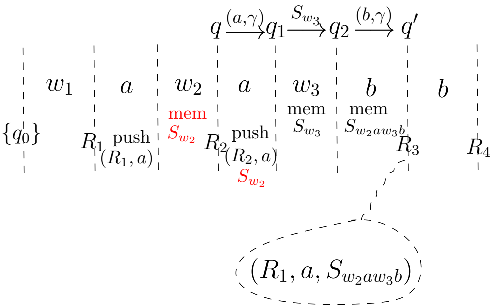
Illustration of the intuition of the proof of the Theorem

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 III. Get inspirations from the subset construction for NFAs.

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



Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration
$$\Big((S,R), (S_2,R_2,a_2)(S_1,R_1,a_1)\bot\Big)$$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot\Big)$$

- If a_3 is the current input at configuration $((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot)$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration
$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot\Big)$$
- If a_3 is the current input at configuration $((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot)$
- If $a_3 \in \Sigma_c$: $\mathcal{A}'$ goes to $((S', R'), (S, R, a_3)(S_2, R_2, a_2)(S_1, R_1, a_1)\bot)$, where
- $S' = \mathrm{Id}_Q$ (note $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$), the initialization of the summary;
- $R' = \{q' \mid \exists q \in R, \gamma \in \Gamma : (q, a_3, q', \gamma) \in \delta\}$, all the states reachable by $\mathcal{A}$ after reading a_3

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration
$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot\Big)$$
- If a_3 is the current input at configuration $((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot)$
- If $a_3 \in \Sigma_c$: $\mathcal{A}'$ goes to $((S', R'), (S, R, a_3)(S_2, R_2, a_2)(S_1, R_1, a_1)\bot)$, where
- $S' = \mathrm{Id}_Q$ (note $\mathrm{Id}_Q = \{(q, q) \mid q \in Q\}$), the initialization of the summary;
- $R' = \{q' \mid \exists q \in R, \gamma \in \Gamma : (q, a_3, q', \gamma) \in \delta\}$, all the states reachable by $\mathcal{A}$ after reading a_3

> If $a_3 \in \Sigma_c$, then $((S, R), a_3, (\mathrm{Id}_Q, R'), (S, R, a_3)) \in \delta'$,
> where $R' = \{q' \mid \exists q \in R, \gamma \in \Gamma.(q, a_3, q', \gamma) \in \delta\}$.

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot\Big)$$

- If a_3 is the current input at configuration $((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot\Big)$$

- If a_3 is the current input at configuration $((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot)$
- If $a_3 \in \Sigma_r$: $\mathcal{A}'$ goes to $((S', R'), (S_1, R_1, a_1)\bot)$, where S' and R' are defined below.

$$\boxed{\begin{array}{l}
((S,R), a_3, (S_2, R_2, a_2), (S', R')) \in \delta', \text{ where} \\
S' = \left\{ (q, q') \;\middle|\; \begin{array}{c} \exists q_1, q_2, q_3, \gamma \in \Gamma : (q, q_1) \in S_2, (q_2, q_3) \in S, \\ (q_1, a_2, q_2, \gamma) \in \delta, (q_3, a_3, \gamma, q') \in \delta \end{array} \right\}, \\
R' = \left\{ q' \;\middle|\; \begin{array}{c} \exists q_1, q_2, q_3, \gamma \in \Gamma : q_1 \in R_2, (q_2, q_3) \in S, \\ (q_1, a_2, q_2, \gamma) \in \delta, (q_3, a_3, \gamma, q') \in \delta \end{array} \right\}
\end{array}}$$

$(q, q_1) \in S_2$: $(q, \alpha) \stackrel{y}{\Longrightarrow}^* (q_1, \alpha)$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration
$$\Big((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot\Big)$$
- If a_3 is the current input at configuration $((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$
- If $a_3 \in \Sigma_r$: $\mathcal{A}'$ goes to $((S',R'),(S_1,R_1,a_1)\bot)$, where S' and R' are defined below.

$$\boxed{\begin{array}{l} ((S,R),a_3,(S_2,R_2,a_2),(S',R')) \in \delta', \text{ where} \\ S' = \left\{ (q,q') \;\middle|\; \begin{array}{c} \exists q_1,q_2,q_3,\gamma \in \Gamma : (q,q_1) \in S_2, (q_2,q_3) \in S, \\ (q_1,a_2,q_2,\gamma) \in \delta, (q_3,a_3,\gamma,q') \in \delta \end{array} \right\}, \\ R' = \left\{ q' \;\middle|\; \begin{array}{c} \exists q_1,q_2,q_3,\gamma \in \Gamma : q_1 \in R_2, (q_2,q_3) \in S, \\ (q_1,a_2,q_2,\gamma) \in \delta, (q_3,a_3,\gamma,q') \in \delta \end{array} \right\} \end{array}}$$

$(q,q_1) \in S_2$: $(q,\alpha) \xRightarrow{y}{}^{*} (q_1,\alpha)$ and $(q_1,a_2,q_2,\gamma) \in \delta$: $(q_1,\alpha) \xRightarrow{a_2}{}^{*} (q_2,\gamma\alpha)$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot\Big)$$

- If a_3 is the current input at configuration $((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot)$
- If $a_3 \in \Sigma_r$: $\mathcal{A}'$ goes to $((S', R'), (S_1, R_1, a_1)\bot)$, where S' and R' are defined below.

$$\boxed{\begin{array}{l}
((S, R), a_3, (S_2, R_2, a_2), (S', R')) \in \delta', \text{ where} \\
S' = \left\{ (q, q') \;\middle|\; \begin{array}{c} \exists q_1, q_2, q_3, \gamma \in \Gamma : (q, q_1) \in S_2, (q_2, q_3) \in S, \\ (q_1, a_2, q_2, \gamma) \in \delta, (q_3, a_3, \gamma, q') \in \delta \end{array} \right\}, \\
R' = \left\{ q' \;\middle|\; \begin{array}{c} \exists q_1, q_2, q_3, \gamma \in \Gamma : q_1 \in R_2, (q_2, q_3) \in S, \\ (q_1, a_2, q_2, \gamma) \in \delta, (q_3, a_3, \gamma, q') \in \delta \end{array} \right\}
\end{array}}$$

$(q, q_1) \in S_2$: $(q, \alpha) \overset{y}{\Longrightarrow}{}^* (q_1, \alpha)$ and $(q_1, a_2, q_2, \gamma) \in \delta$: $(q_1, \alpha) \overset{a_2}{\Longrightarrow}{}^* (q_2, \gamma\alpha)$

$(q_2, q_3) \in S$: $(q_2, \gamma\alpha) \overset{z}{\Longrightarrow}{}^* (q_3, \gamma\alpha)$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot\Big)$$

- If a_3 is the current input at configuration $((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot)$
- If $a_3 \in \Sigma_r$: $\mathcal{A}'$ goes to $((S', R'), (S_1, R_1, a_1)\bot)$, where S' and R' are defined below.

$((S, R), a_3, (S_2, R_2, a_2), (S', R')) \in \delta'$, where

$$S' = \left\{ (q, q') \;\middle|\; \begin{array}{c} \exists q_1, q_2, q_3, \gamma \in \Gamma : (q, q_1) \in S_2, (q_2, q_3) \in S, \\ (q_1, a_2, q_2, \gamma) \in \delta, (q_3, a_3, \gamma, q') \in \delta \end{array} \right\},$$

$$R' = \left\{ q' \;\middle|\; \begin{array}{c} \exists q_1, q_2, q_3, \gamma \in \Gamma : q_1 \in R_2, (q_2, q_3) \in S, \\ (q_1, a_2, q_2, \gamma) \in \delta, (q_3, a_3, \gamma, q') \in \delta \end{array} \right\}$$

$(q, q_1) \in S_2$: $(q, \alpha) \stackrel{y}{\Longrightarrow}^* (q_1, \alpha)$ and $(q_1, a_2, q_2, \gamma) \in \delta$: $(q_1, \alpha) \stackrel{a_2}{\Longrightarrow}^* (q_2, \gamma\alpha)$

$(q_2, q_3) \in S$: $(q_2, \gamma\alpha) \stackrel{z}{\Longrightarrow}^* (q_3, \gamma\alpha)$ and $(q_3, a_3, \gamma, q') \in \delta$: $(q_3, \gamma\alpha) \stackrel{a_3}{\Longrightarrow}^* (q', \alpha)$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration
$$\Big((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot\Big)$$
- If a_3 is the current input at configuration $((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$
- If $a_3 \in \Sigma_r$: $\mathcal{A}'$ goes to $((S',R'),(S_1,R_1,a_1)\bot)$, where S' and R' are defined below.

$((S,R),a_3,(S_2,R_2,a_2),(S',R')) \in \delta'$, where

$$S' = \left\{(q,q') \;\middle|\; \begin{array}{c} \exists q_1,q_2,q_3,\gamma \in \Gamma : (q,q_1) \in S_2, (q_2,q_3) \in S, \\ (q_1,a_2,q_2,\gamma) \in \delta, (q_3,a_3,\gamma,q') \in \delta \end{array} \right\},$$

$$R' = \left\{q' \;\middle|\; \begin{array}{c} \exists q_1,q_2,q_3,\gamma \in \Gamma : q_1 \in R_2, (q_2,q_3) \in S, \\ (q_1,a_2,q_2,\gamma) \in \delta, (q_3,a_3,\gamma,q') \in \delta \end{array} \right\}$$

$(q,q_1) \in S_2$: $(q,\alpha) \stackrel{y}{\Longrightarrow}{}^* (q_1,\alpha)$ and $(q_1,a_2,q_2,\gamma) \in \delta$: $(q_1,\alpha) \stackrel{a_2}{\Longrightarrow}{}^* (q_2,\gamma\alpha)$

$(q_2,q_3) \in S$: $(q_2,\gamma\alpha) \stackrel{z}{\Longrightarrow}{}^* (q_3,\gamma\alpha)$ and $(q_3,a_3,\gamma,q') \in \delta$: $(q_3,\gamma\alpha) \stackrel{a_3}{\Longrightarrow}{}^* (q',\alpha)$

$$(q,\alpha) \stackrel{ya_2za_3}{\Longrightarrow}{}^* (q',\alpha)$$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration

$$\Big((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot\Big)$$

- If a_3 is the current input at configuration $((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$
- If $a_3 \in \Sigma_r$: $\mathcal{A}'$ goes to $((S',R'),(S_1,R_1,a_1)\bot)$, where S' and R' are defined below.

$((S,R),a_3,(S_2,R_2,a_2),(S',R')) \in \delta'$, where

$$S' = \left\{ (q,q') \;\middle|\; \begin{array}{c} \exists q_1,q_2,q_3,\gamma \in \Gamma : (q,q_1) \in S_2, (q_2,q_3) \in S, \\ (q_1,a_2,q_2,\gamma) \in \delta, (q_3,a_3,\gamma,q') \in \delta \end{array} \right\},$$

$$R' = \left\{ q' \;\middle|\; \begin{array}{c} \exists q_1,q_2,q_3,\gamma \in \Gamma : q_1 \in R_2, (q_2,q_3) \in S, \\ (q_1,a_2,q_2,\gamma) \in \delta, (q_3,a_3,\gamma,q') \in \delta \end{array} \right\},$$

or

$((S,R),a_3,\bot,(S',R')) \in \delta'$, where
$S' = \{(q,q') \mid \exists q''.(q,q'') \in S, (q'',a_3,\bot,q') \in \delta\}$,
$R' = \{q' \mid \exists q \in R.(q,a_3,\bot,q') \in \delta\}$.

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration
$$\Big((S, R), (S_2, R_2, a_2)(S_1, R_1, a_1)\bot\Big)$$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration
$$\Big((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot\Big)$$
- If a_3 is the current input at configuration $((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration
$$\big((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot\big)$$
- If a_3 is the current input at configuration $((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$
- If $a_3 \in \Sigma_I$: $\mathcal{A}'$ goes to $((S',R'),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$, where
  - $S' = \{(q,q') \mid \exists q'' : (q,q'') \in S, (q'',a_3,q') \in \delta\}$, the initialization of the summary;
  - $R' = \{q' \mid \exists q \in R : (q,a_3,q') \in \delta\}$

> If $a_3 \in \Sigma_I$, then $((S,R),a_3,(S',R')) \in \delta'$,
> where $S' = \{(q,q') \mid \exists q'' : (q,q'') \in S, (q'',a_3,q') \in \delta\}$ and
> $R' = \{q' \mid \exists q \in R : (q,a_3,q') \in \delta\}$.

$(q,q'') \in S$: $(q,\alpha) \overset{z}{\Longrightarrow}^* (q'',\alpha)$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration
$$\Big((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot\Big)$$
- If a_3 is the current input at configuration $((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$
- If $a_3 \in \Sigma_l$: $\mathcal{A}'$ goes to $((S',R'),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$, where
  - $S' = \{(q,q') \mid \exists q'' : (q,q'') \in S, (q'',a_3,q') \in \delta\}$, the initialization of the summary;
  - $R' = \{q' \mid \exists q \in R : (q,a_3,q') \in \delta\}$

> If $a_3 \in \Sigma_l$, then $((S,R),a_3,(S',R')) \in \delta'$,
> where $S' = \{(q,q') \mid \exists q'' : (q,q'') \in S, (q'',a_3,q') \in \delta\}$ and $R' = \{q' \mid \exists q \in R : (q,a_3,q') \in \delta\}$.

$(q,q'') \in S$: $(q,\alpha) \overset{z}{\Longrightarrow}{}^* (q'',\alpha)$ and $(q'',a_3,q') \in \delta$: $(q'',\alpha) \overset{a_3}{\Longrightarrow}{}^* (q',\alpha)$

Determinization

- After reading $w = xa_1ya_2z$, the VPA $\mathcal{A}'$ we construct will reach the configuration
$$\Big((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot\Big)$$
- If a_3 is the current input at configuration $((S,R),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$
- If $a_3 \in \Sigma_I$: $\mathcal{A}'$ goes to $((S',R'),(S_2,R_2,a_2)(S_1,R_1,a_1)\bot)$, where
  - $S' = \{(q,q') \mid \exists q'' : (q,q'') \in S, (q'',a_3,q') \in \delta\}$, the initialization of the summary;
  - $R' = \{q' \mid \exists q \in R : (q,a_3,q') \in \delta\}$

> If $a_3 \in \Sigma_I$, then $((S,R),a_3,(S',R')) \in \delta'$,
> where $S' = \{(q,q') \mid \exists q'' : (q,q'') \in S, (q'',a_3,q') \in \delta\}$ and $R' = \{q' \mid \exists q \in R : (q,a_3,q') \in \delta\}$.

$(q,q'') \in S$: $(q,\alpha) \overset{z}{\Longrightarrow}{}^* (q'',\alpha)$ and $(q'',a_3,q') \in \delta$: $(q'',\alpha) \overset{a_3}{\Longrightarrow}{}^* (q',\alpha)$

$$(q,\alpha) \overset{za_3}{\Longrightarrow}{}^* (q',\alpha)$$

Determinization

We construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), F')$:

Determinization

We construct $\mathcal{A}' = (Q', \widetilde{\Sigma}, \Gamma', \delta', (\mathrm{Id}_Q, \{q_0\}), F')$:

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

Complementation

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

Complementation

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

For a VPA $\mathcal{A}$, first construct a deterministic VPA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. Then, complement the set of final states.

Outline

Visibly pushdown automata (VPA)

Closure properties

Visibly pushdown grammar (VPG)

Logical characterization
- Equivalence of NFA and MSO
- Equivalence of VPA and MSO_{μ}

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 $Z \in \mathcal{N}_0$.

The non-terminals in $\mathcal{N}_0$ derive only well-matched words where there is a one-to-one correspondence between calls and returns.
The non-terminals in $\mathcal{N}_1$ derive words that can contain unmatched calls as well as unmatched returns.

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 $Z \in \mathcal{N}_0$.

The non-terminals in $\mathcal{N}_0$ derive only well-matched words where there is a one-to-one correspondence between calls and returns.
The non-terminals in $\mathcal{N}_1$ derive words that can contain unmatched calls as well as unmatched returns.

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 $\{a^n b^n \mid n \geq 1\}$.

Equivalence of VPA and VPG

Theorem. $\mathrm{VPA} \equiv \mathrm{VPG}$.

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.

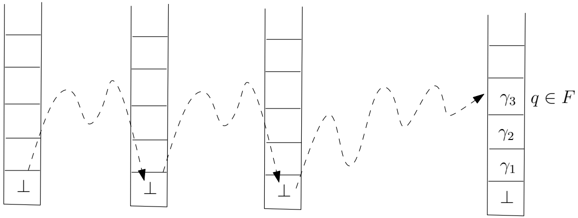
Equivalence of VPA and VPG

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

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

Equivalence of VPA and 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.

Equivalence of VPA and 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.

- $S = (q_0, \perp)$,

Equivalence of VPA and 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.

- $S = (q_0, \perp)$,
- $\mathcal{N}_0 = \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \setminus \{\perp\}\}$,

Equivalence of VPA and 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.

- $S = (q_0, \perp)$,
- $\mathcal{N}_0 = \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \setminus \{\perp\}\}$,
- $\mathcal{N}_1 = \{(q, \perp) \mid q \in Q\} \cup \{q \mid q \in Q\}$,

Equivalence of VPA and 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.

- $S = (q_0, \perp)$,
- $\mathcal{N}_0 = \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \setminus \{\perp\}\}$,
- $\mathcal{N}_1 = \{(q, \perp) \mid q \in Q\} \cup \{q \mid q \in Q\}$,
  - $(q, \perp)$: the state is q and the stack is empty,
  - q: the state is q and the stack is nonempty.

Equivalence of VPA and 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.

- $S = (q_0, \perp)$,
- $\mathcal{N}_0 = \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \setminus \{\perp\}\}$,
- $\mathcal{N}_1 = \{(q, \perp) \mid q \in Q\} \cup \{q \mid q \in Q\}$,
  - $(q, \perp)$: the state is q and the stack is empty,
  - q: the state is q and the stack is nonempty.
- $\mathcal{P}$ is defined by the following rules,

Equivalence of VPA and 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.

- $S = (q_0, \perp)$,
- $\mathcal{N}_0 = \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \setminus \{\perp\}\}$,
- $\mathcal{N}_1 = \{(q, \perp) \mid q \in Q\} \cup \{q \mid q \in Q\}$,
  - $(q, \perp)$: the state is q and the stack is empty,
  - q: the state is q and the stack is nonempty.
- $\mathcal{P}$ is defined by the following rules,
  - if $(q, a, q') \in \delta$ s.t. $a \in \Sigma_I$, then
    $(q, \perp) \to a(q', \perp)$, $q \to aq'$, $[q, \gamma, p] \to a[q', \gamma, p]$.

Equivalence of VPA and 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.

- $S = (q_0, \perp)$,
- $\mathcal{N}_0 = \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \setminus \{\perp\}\}$,
- $\mathcal{N}_1 = \{(q, \perp) \mid q \in Q\} \cup \{q \mid q \in Q\}$,
  - $(q, \perp)$: the state is q and the stack is empty,
  - q: the state is q and the stack is nonempty.
- $\mathcal{P}$ 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
    $[q, \gamma_1, r] \to a[q', \gamma, p']b[p, \gamma_1, r]$, $(q, \perp) \to a[q', \gamma, p']b(p, \perp)$,
    $q \to a[q', \gamma, p']bp$.

Equivalence of VPA and 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.

- $S = (q_0, \perp)$,
- $\mathcal{N}_0 = \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \setminus \{\perp\}\}$,
- $\mathcal{N}_1 = \{(q, \perp) \mid q \in Q\} \cup \{q \mid q \in Q\}$,
  - $(q, \perp)$: the state is q and the stack is empty,
  - q: the state is q and the stack is nonempty.
- $\mathcal{P}$ is defined by the following rules,
  - if $(q, a, q') \in \delta$ s.t. $a \in \Sigma_l$, then
    $(q, \perp) \rightarrow a(q', \perp)$, $q \rightarrow aq'$, $[q, \gamma, p] \rightarrow a[q', \gamma, p]$.
  - if $(q, a, q', \gamma), (p', b, \gamma, p) \in \delta$ s.t. $a \in \Sigma_c, b \in \Sigma_r$, then
    $[q, \gamma_1, r] \rightarrow a[q', \gamma, p']b[p, \gamma_1, r]$, $(q, \perp) \rightarrow a[q', \gamma, p']b(p, \perp)$,
    $q \rightarrow a[q', \gamma, p']bp$.
  - if $(q, a, q', \gamma) \in \delta$ s.t. $a \in \Sigma_c$, then
    $(q, \perp) \rightarrow aq'$, $q \rightarrow aq'$, $(q, \perp) \rightarrow a[q', \gamma, p]$, $q \rightarrow a[q', \gamma, p]$.

Equivalence of VPA and 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.

- $S = (q_0, \perp)$,
- $\mathcal{N}_0 = \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \setminus \{\perp\}\}$,
- $\mathcal{N}_1 = \{(q, \perp) \mid q \in Q\} \cup \{q \mid q \in Q\}$,
  - $(q, \perp)$: the state is q and the stack is empty,
  - q: the state is q and the stack is nonempty.
- $\mathcal{P}$ 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 $(q, \perp) \to a(q', \perp)$.

Equivalence of VPA and 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.

- $S = (q_0, \perp)$,
- $\mathcal{N}_0 = \{[q, \gamma, p] \mid q, p \in Q, \gamma \in \Gamma \setminus \{\perp\}\}$,
- $\mathcal{N}_1 = \{(q, \perp) \mid q \in Q\} \cup \{q \mid q \in Q\}$,
    - $(q, \perp)$: the state is q and the stack is empty,
    - q: the state is q and the stack is nonempty.
- $\mathcal{P}$ 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 $\mathcal{A} = (\mathcal{N}, \widetilde{\Sigma}, \Sigma_r \times \mathcal{N} \cup \{\perp, \$\}, \delta, S, F)$ as follows.

- δ is defined by the following rules,
  - if $X \to aY$ s.t. $a \in \Sigma_l$, then $(X, a, Y) \in \delta$,

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 $(X, a, Y, \$) \in \delta$,

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 $(X, a, \perp, Y) \in \delta$,

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 $(X, a, Y, (b, Z)) \in \delta$,

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 $(X, b, (b, Y), Y) \in \delta$.

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 $\perp$.

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.

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 $\forall \gamma.((X, \gamma), a, (Y, \gamma)) \in \delta'$,

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 $\forall\gamma.((X, \gamma), a, (Y, \$), (\$, \gamma)) \in \delta'$,

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 $\bot$.

  Adapt $\mathcal{A}$ *into* $\mathcal{A}' = (\mathcal{N} \times \Gamma, \widetilde{\Sigma}, \Gamma, \delta', (S, \bot), \{(X, \gamma) \mid X \to \varepsilon, \gamma = \$, \bot\})$
  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
    $\forall\gamma.((X, \gamma), a, \bot, (Y, \bot)) \in \delta'$ and $\forall\gamma.((X, \$), a, (\$, \gamma), (Y, \gamma)) \in \delta'$,

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 $\bot$.

Adapt $\mathcal{A}$ *into* $\mathcal{A}' = (\mathcal{N} \times \Gamma, \widetilde{\Sigma}, \Gamma, \delta', (S, \bot), \{(X, \gamma) \mid X \to \varepsilon, \gamma = \$, \bot\})$
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
  $\forall \gamma.((X, \gamma), a, \bot, (Y, \bot)) \in \delta'$ and $\forall \gamma.((X, \$), a, (\$, \gamma), (Y, \gamma)) \in \delta'$,
- if $X \to aYbZ$, then $\forall \gamma.((X, \gamma), a, (Y, (b, Z)), ((b, Z), \gamma)) \in \delta'$,

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

Visibly pushdown automata (VPA)

Closure properties

Visibly pushdown grammar (VPG)

Logical characterization

- Equivalence of NFA and MSO
- Equivalence of VPA and MSO_μ

Decision problems

Outline

Visibly pushdown automata (VPA)

Closure properties

Visibly pushdown grammar (VPG)

Logical characterization

- Equivalence of NFA and MSO
- Equivalence of VPA and MSO_μ

Decision problems

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 $\sigma \in \Sigma$, x, y are position (first-order) variables, X is a set (second-order) variables

An MSO sentence is a MSO formula without free variables.

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 $\sigma \in \Sigma$, x, y are position (first-order) variables, X is a set (second-order) variables

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 $(P_\sigma)^{\mathcal{S}}$.

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 $\sigma \in \Sigma$, x, y are position (first-order) variables, X is a set (second-order) variables

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.

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 $\sigma \in \Sigma$, x, y are position (first-order) variables, X is a set (second-order) variables

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.

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

where $\sigma \in \Sigma$, x, y are position (first-order) variables, X is a set (second-order) variables

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 $\sigma \in \Sigma$, x, y are position (first-order) variables, X is a set (second-order) variables

An MSO sentence is a MSO formula without free variables.

Semantics. A MSO formula φ is satisfied over a word $w = a_1 \ldots a_n$, with a valuation $\mathcal{I}$ of $\mathrm{free}(\varphi)$ 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 $\sigma \in \Sigma$, x, y are position (first-order) variables, X is a set (second-order) variables

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.

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

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.

$$\neg\exists x \ \mathrm{first}(x), \ \text{i.e.,} \ \varepsilon$$

Monadic Second-Order Logic (continued)

Abbreviations.

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

Example.

$$\neg\exists x \; \mathrm{first}(x), \text{ i.e., } \varepsilon$$

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

i.e., all the words that has two positions where b occurs later than a

Monadic Second-Order Logic (continued)

Abbreviations.

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

Example.

$$\neg\exists x\ \mathrm{first}(x),\ \text{i.e., } \varepsilon$$

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

i.e., all the words that has two positions where b occurs later than a

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

i.e., non-empty words and all the even positions have a

From NFA to MSO
Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA.

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,

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

where

- X_q stands for the positions where the run is in state q,
- $\varphi_{unique} = \bigwedge_{q \neq q'} \forall x \neg (X_q(x) \wedge X_{q'}(x))$
- $\varphi_{init} = \exists x (\mathrm{first}(x) \wedge \bigvee\limits_{(q_0, a, q) \in \delta} (P_a(x) \wedge X_q(x))),$
- $\varphi_{trans} = \forall x \forall y (\mathrm{suc}(x, y) \rightarrow \bigvee\limits_{(q, a, q') \in \delta} X_q(x) \wedge P_a(y) \wedge X_{q'}(y)),$
- $\varphi_{final} = \exists x (\mathrm{last}(x) \wedge \bigvee\limits_{q \in F} X_q(x)).$

Then $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A})$ if $q_0 \notin F$ and $\mathcal{L}(\varphi \vee \forall x (\neg \mathrm{first}(x))) = \mathcal{L}(\mathcal{A})$ if $q_0 \in F$.

From MSO to NFA.

A normal form for MSO formulas

New modalities,

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

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

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

From MSO to NFA.

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

Let $\varphi(X_1, \ldots, X_k)$ be a MSO formula in the prefix normal form.
We construct a NFA $\mathcal{A} = (Q, \Sigma \times \{0,1\}^k, \delta, q_0, F)$ as follows.
Consider $\varphi(X_1, \cdots, X_k)$ with $X_1, \cdots, X_k$ free variables, $(a, b_1 \cdots b_k)$ at position p such that $a \in \Sigma$ and $b_i \in \{0,1\}$ denotes that

- a is the symbol at position p,
- $b_i = 1$ denotes that $p \in X_i$
- $b_i = 0$ denotes that $p \notin X_i$

From MSO to NFA.

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

Let $\varphi(X_1, \ldots, X_k)$ be a MSO formula in the prefix normal form.
We construct a NFA $\mathcal{A} = (Q, \Sigma \times \{0,1\}^k, \delta, q_0, F)$ as follows.
Consider $\varphi(X_1, \cdots, X_k)$ with $X_1, \cdots, X_k$ free variables, $(a, b_1 \cdots b_k)$ at position p such that $a \in \Sigma$ and $b_i \in \{0,1\}$ denotes that

- a is the symbol at position p,
- $b_i = 1$ denotes that $p \in X_i$
- $b_i = 0$ denotes that $p \notin X_i$

input	a_1	a_2	a_3	a_4	a_5	$a_6 \cdots$
X_1	1	0	1	0	1	$0 \cdots$
X_2	0	0	1	1	0	$1 \cdots$

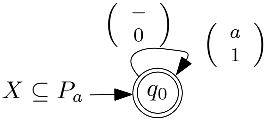
X_1: even positions
X_2: prime positions

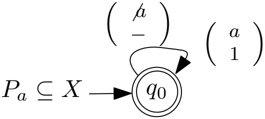
NFA≡MSO

From MSO to NFA.

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

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



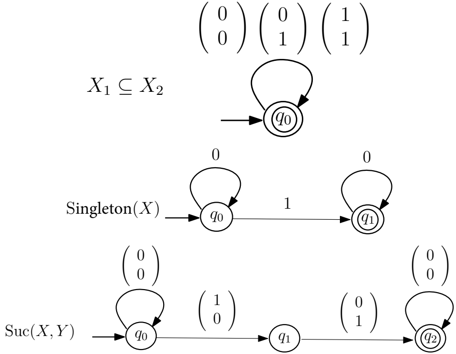


NFA≡MSO

From MSO to NFA.

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

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



NFA≡MSO

From MSO to NFA.

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

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,
- $\varphi = \exists X_1 \varphi_1$
  NFAs are closed under projection (a special case of homomorphisms), e.g. $(b_1, \dots, b_k) \to (b_2, \dots, b_k)$.

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

Outline

Visibly pushdown automata (VPA)

Closure properties

Visibly pushdown grammar (VPG)

Logical characterization

- Equivalence of NFA and MSO
- Equivalence of VPA and MSO_μ

Decision problems

MSO_μ

Fix $\widetilde{\Sigma}$.

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

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

MSO$_\mu$

Fix $\widetilde{\Sigma}$.
Given a word $w = a_1 \ldots a_n \in \Sigma^*$, 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.

MSO_μ

Fix $\widetilde{\Sigma}$.
Given a word $w = a_1 \ldots a_n \in \Sigma^*$, 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 $\widetilde{\Sigma}$.

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

where $\sigma \in \Sigma$.

MSO_μ

Fix $\widetilde{\Sigma}$.
Given a word $w = a_1 \ldots a_n \in \Sigma^*$, a binary relation $\mu(x, y)$ can be defined such that

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

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

Syntax of MSO_μ **over** $\widetilde{\Sigma}$.

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

where $\sigma \in \Sigma$.

Semantics of $\mathbf{MSO}_\mu$ **over** $\widetilde{\Sigma}$.

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

MSO$_\mu$

Fix $\widetilde{\Sigma}$.
Given a word $w = a_1 \ldots a_n \in \Sigma^*$, 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$_\mu$* over $\widetilde{\Sigma}$.

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

where $\sigma \in \Sigma$.

Semantics of MSO$_\mu$ over $\widetilde{\Sigma}$.

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

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

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

From VPA to MSO$_\mu$

Let $\mathcal{A} = (Q, \widetilde{\Sigma}, \Gamma, \delta, q_0, \perp, F)$ be a VPA, $Q = \{q_0, \ldots, q_n\}$, $\Gamma = \{\gamma_1, \ldots, \gamma_k\}$.

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 $\varphi := \exists X_{q_0} \ldots X_{q_n} P_{\gamma_1} \ldots P_{\gamma_k} (\varphi_{unique} \wedge \varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final})$ as follows,

From VPA to MSO_μ

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 $\varphi := \exists X_{q_0} \ldots X_{q_n} P_{\gamma_1} \ldots P_{\gamma_k} (\varphi_{unique} \wedge \varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final})$ as follows,

- X_q stands for the positions where the run is in state q

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 $\varphi := \exists X_{q_0} \ldots X_{q_n} P_{\gamma_1} \ldots P_{\gamma_k} (\varphi_{unique} \wedge \varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final})$ as follows,

- X_q stands for the positions where the run is in state q
- P_γ stands for the positions where r is pushed/popped

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 $\varphi := \exists X_{q_0} \ldots X_{q_n} P_{\gamma_1} \ldots P_{\gamma_k} (\varphi_{unique} \wedge \varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final})$ as follows,

- X_q stands for the positions where the run is in state q
- P_γ stands for the positions where r is pushed/popped
- $\varphi_{unique} = \bigwedge_{q \neq q'} \forall x \neg (X_q(x) \wedge X_{q'}(x)) \wedge \bigwedge_{\gamma \neq \gamma'} \forall x \neg (P_\gamma(x) \wedge P_{\gamma'}(x))$

From VPA to MSO_μ

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 $\varphi := \exists X_{q_0} \ldots X_{q_n} P_{\gamma_1} \ldots P_{\gamma_k} (\varphi_{unique} \wedge \varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final})$ as follows,

- X_q stands for the positions where the run is in state q
- P_γ stands for the positions where r is pushed/popped
- $\varphi_{unique} = \bigwedge_{q \neq q'} \forall x \neg (X_q(x) \wedge X_{q'}(x)) \wedge \bigwedge_{\gamma \neq \gamma'} \forall x \neg (P_\gamma(x) \wedge P_{\gamma'}(x))$
- $$\varphi_{init} = \exists x \left(\mathrm{first}(x) \wedge \left(\begin{array}{c} \bigvee_{(q_0,a,q) \in \delta} (P_a(x) \wedge X_q(x)) \bigvee \\ \bigvee_{(q_0,a,q,\gamma) \in \delta} (P_a(x) \wedge X_q(x) \wedge P_\gamma(x)) \bigvee \\ \bigvee_{(q_0,a,\perp,q) \in \delta} (P_a(x) \wedge X_q(x) \wedge P_\perp(x)) \end{array} \right) \right),$$

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 $\varphi := \exists X_{q_0} \ldots X_{q_n} P_{\gamma_1} \ldots P_{\gamma_k} (\varphi_{unique} \wedge \varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final})$ as follows,

From VPA to MSO$_\mu$

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

- $\varphi_{trans} = \forall x \forall y \, (\mathrm{suc}(x, y) \to \psi_{call} \vee \psi_{return} \vee \psi_{local})$,

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 $\varphi := \exists X_{q_0} \ldots X_{q_n} P_{\gamma_1} \ldots P_{\gamma_k} (\varphi_{unique} \wedge \varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final})$ as follows,

- $\varphi_{trans} = \forall x \forall y \, (\mathrm{suc}(x, y) \rightarrow \psi_{call} \vee \psi_{return} \vee \psi_{local})$, where
  - $\psi_{call} = \bigvee_{(q,a,q',\gamma)\in\delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\gamma(y))$,

From VPA to MSO$_\mu$

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

- $\varphi_{trans} = \forall x \forall y \, (\mathrm{suc}(x, y) \to \psi_{call} \vee \psi_{return} \vee \psi_{local})$, where
  - $\psi_{call} = \bigvee_{(q,a,q',\gamma) \in \delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\gamma(y))$,
  - $\psi_{local} = \bigvee_{(q,a,q') \in \delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y))$,

From VPA to MSO$_\mu$

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

- $\varphi_{trans} = \forall x \forall y \, (\mathrm{suc}(x, y) \to \psi_{call} \vee \psi_{return} \vee \psi_{local})$, where
  - $\psi_{call} = \bigvee_{(q,a,q',\gamma)\in\delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\gamma(y))$,
  - $\psi_{local} = \bigvee_{(q,a,q')\in\delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y))$,
  - $\psi_{return} =$
  $$\bigvee_{(q,a,\gamma,q')\in\delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\gamma(y) \wedge \exists z(\mu(z, y) \wedge P_\gamma(z))) \bigvee$$
  $$\bigvee_{(q,a,\perp,q')\in\delta} (q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\perp(y) \wedge \neg\exists z(\mu(z, y))) .$$

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 $\varphi := \exists X_{q_0} \ldots X_{q_n} P_{\gamma_1} \ldots P_{\gamma_k} (\varphi_{unique} \wedge \varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final})$ as follows,

- $\varphi_{trans} = \forall x \forall y \, (\mathrm{suc}(x, y) \to \psi_{call} \vee \psi_{return} \vee \psi_{local})$, where
  - $\psi_{call} = \bigvee_{(q,a,q',\gamma)\in\delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\gamma(y))$,
  - $\psi_{local} = \bigvee_{(q,a,q')\in\delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y))$,
  - $\psi_{return} =$ $$\begin{array}{l} \bigvee_{(q,a,\gamma,q')\in\delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\gamma(y) \wedge \exists z(\mu(z, y) \wedge P_\gamma(z))) \bigvee \\ \bigvee_{(q,a,\perp,q')\in\delta} (q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\perp(y) \wedge \neg\exists z(\mu(z, y))) \end{array} .$$

- $\varphi_{final} = \exists x \left(\mathrm{last}(x) \wedge \bigvee_{q\in F} X_q(x) \right)$.

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 $\varphi := \exists X_{q_0} \ldots X_{q_n} P_{\gamma_1} \ldots P_{\gamma_k} (\varphi_{unique} \wedge \varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{final})$ as follows,

- $\varphi_{trans} = \forall x \forall y \, (\mathrm{suc}(x, y) \to \psi_{call} \vee \psi_{return} \vee \psi_{local})$, where
  - $\psi_{call} = \bigvee_{(q,a,q',\gamma)\in\delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\gamma(y))$,
  - $\psi_{local} = \bigvee_{(q,a,q')\in\delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y))$,
  - $\psi_{return} =$
$$\bigvee_{(q,a,\gamma,q')\in\delta} (X_q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\gamma(y) \wedge \exists z(\mu(z, y) \wedge P_\gamma(z))) \bigvee \bigvee_{(q,a,\perp,q')\in\delta} (q(x) \wedge P_a(y) \wedge X_{q'}(y) \wedge P_\perp(y) \wedge \neg\exists z(\mu(z, y))) .$$
- $\varphi_{final} = \exists x \left(\mathrm{last}(x) \wedge \bigvee_{q\in F} X_q(x) \right)$.

Then $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A})$ if $q_0 \notin F$ and $\mathcal{L}(\varphi \vee \forall x(\neg \mathrm{first}(x))) = \mathcal{L}(\mathcal{A})$ if $q_0 \in F$.

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

From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.

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

From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.

We construct the VPA by structural induction.

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

From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.
We construct the VPA by structural induction.
Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.

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

From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.
We construct the VPA by structural induction.
Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.
Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0,1\}^k$ for $s \in \{c, r, l\}$.

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

From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.

We construct the VPA by structural induction.

Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.

Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0,1\}^k$ for $s \in \{c, r, l\}$.

A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.

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

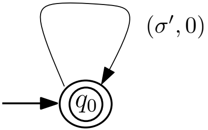
From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.
We construct the VPA by structural induction.
Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.
Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0,1\}^k$ for $s \in \{c, r, l\}$.
A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.

$$\sigma', \sigma'' \neq \sigma : \begin{array}{ll} (\sigma', 0), \downarrow (\sigma', 0) & (\sigma', 0), \uparrow (\sigma'', 0) \\ (\sigma', 0), \uparrow (\sigma, 1) & (\sigma', 0), \uparrow \perp (\sigma', 0), \uparrow (\sigma, 0) \\ (\sigma, 1), \downarrow (\sigma, 1) & (\sigma, 0), \downarrow (\sigma, 0) \end{array}$$

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



where $\downarrow$=push, $\uparrow$=pop. If push, $\sigma' \in \Sigma_c$; If pop, $\sigma' \in \Sigma_r$; otherwise $\sigma' \in \Sigma_l$; If σ'' in stack, $\sigma'' \in \Sigma_c$.

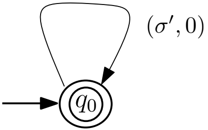
From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.
We construct the VPA by structural induction.
Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.
Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0,1\}^k$ for $s \in \{c, r, l\}$.
A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.

$$\sigma', \sigma'' \neq \sigma : \begin{array}{ll} (\sigma', 0), \downarrow (\sigma', 0) & (\sigma', 0), \uparrow (\sigma'', 0) \\ (\sigma', 0), \uparrow (\sigma, 1) & (\sigma', 0), \uparrow \perp (\sigma', 0), \uparrow (\sigma, 0) \\ (\sigma, 1), \downarrow (\sigma, 1) & (\sigma, 0), \downarrow (\sigma, 0) \end{array}$$

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



where $\downarrow$=push, $\uparrow$=pop. If push, $\sigma' \in \Sigma_c$; If pop, $\sigma' \in \Sigma_r$; otherwise $\sigma' \in \Sigma_l$; If σ'' in stack, $\sigma'' \in \Sigma_c$. $P_\sigma \subseteq X$ can be handled similarly!

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

From MSO$_\mu$ to VPA.

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

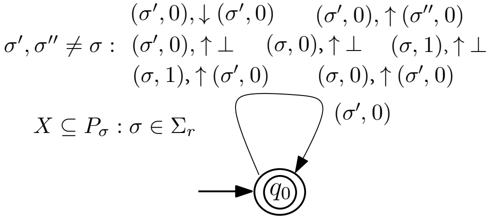
Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.

We construct the VPA by structural induction.

Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.

Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0,1\}^k$ for $s \in \{c, r, l\}$.

A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.



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

From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.

We construct the VPA by structural induction.

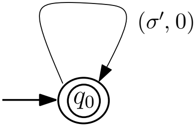
Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.

Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0,1\}^k$ for $s \in \{c, r, l\}$.

A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.

$$\sigma', \sigma'' \neq \sigma : \begin{array}{lll} (\sigma',0), \downarrow (\sigma',0) & (\sigma',0), \uparrow (\sigma'',0) & \\ (\sigma',0), \uparrow \perp & (\sigma,0), \uparrow \perp & (\sigma,1), \uparrow \perp \\ (\sigma,1), \uparrow (\sigma',0) & (\sigma,0), \uparrow (\sigma',0) & \end{array}$$

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



$P_\sigma \subseteq X$ can be handled similarly!

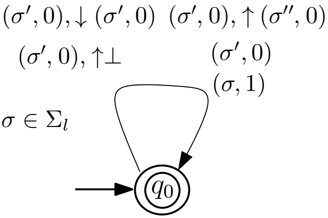
VPA $\equiv$ MSO$_\mu$: continued

From MSO$_\mu$ to VPA.
Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.
We construct the VPA by structural induction.
Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.
Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0,1\}^k$ for $s \in \{c, r, l\}$.
A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.

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

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

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



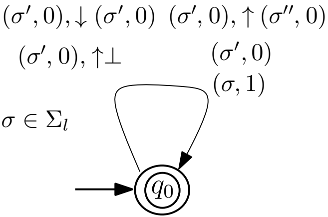
VPA $\equiv$ MSO$_\mu$: continued

From MSO$_\mu$ to VPA.
Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.
We construct the VPA by structural induction.
Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.
Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0,1\}^k$ for $s \in \{c, r, l\}$.
A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.

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

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

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



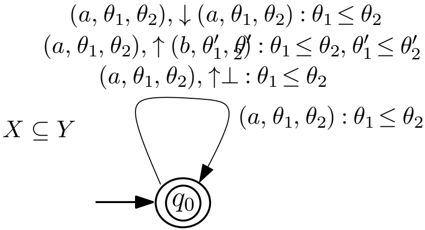
$P_\sigma \subseteq X$ can be handled similarly!

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

From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.
We construct the VPA by structural induction.
Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.
Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0, 1\}^k$ for $s \in \{c, r, l\}$.
A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.



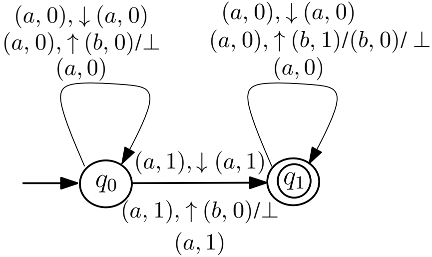
VPA $\equiv$ MSO$_\mu$: continued

From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.
We construct the VPA by structural induction.
Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.
Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0, 1\}^k$ for $s \in \{c, r, l\}$.
A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.

$\mathrm{Singleton}(X)$

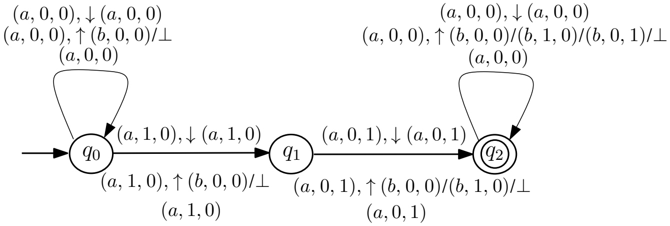


From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.
We construct the VPA by structural induction.
Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.
Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0,1\}^k$ for $s \in \{c, r, l\}$.
A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.

$\mathrm{suc}(X, Y)$



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

From MSO$_\mu$ to VPA.

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

Let $\widetilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ and assume that the MSO$_\mu$ formula φ is given in prefix normal form.

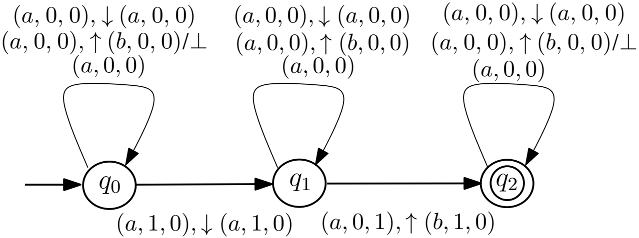
We construct the VPA by structural induction.

Consider a MSO$_\mu$ $\phi(X_1, \cdots, X_k)$ with free variables $X_1, \cdots, X_k$.

Define, $\widetilde{\Sigma'} = \langle \Sigma'_c, \Sigma'_r, \Sigma'_l \rangle$, where $\Sigma'_s = \Sigma_s \times \{0, 1\}^k$ for $s \in \{c, r, l\}$.

A word w' over Σ' encodes a word w over Σ with valuations of all X_i's.

$\mu(X, Y)$



Satisfiability of MSO and MSO_μ

- A MSO formula ϕ over finite words is satisfiable iff the NFA $\mathcal{A}_\phi$ is nonempty.

Satisfiability of MSO and MSO$_\mu$

- A MSO formula ϕ over finite words is satisfiable iff the NFA $\mathcal{A}_\phi$ is nonempty.
- A MSO$_\mu$ formula ϕ over visibly pushdown words is satisfiable iff the VPA $\mathcal{A}_\phi$ is nonempty.

Satisfiability of MSO and MSO$_\mu$

- A MSO formula ϕ over finite words is satisfiable iff the NFA $\mathcal{A}_\phi$ is nonempty.
- A MSO$_\mu$ formula ϕ over visibly pushdown words is satisfiable iff the VPA $\mathcal{A}_\phi$ is nonempty.
- Upper Bound [Büchi, Elgot, Trakhtenbrot, 1957-8 (independently)]: Nonelementary Growth $2^{\cdot^{\cdot^{\cdot^{2^n}}}}$, (tower of height $O(n)$), due to complementation
- Lower Bound [Stockmeyer, 1974]: Satisfiability of FO over finite words is nonelementary (no bounded height tower).

Outline

Visibly pushdown automata (VPA)

Closure properties

Visibly pushdown grammar (VPG)

Logical characterization

- Equivalence of NFA and MSO
- Equivalence of VPA and MSO_{μ}

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 and universality problems

Theorem. The language inclusion problem and universality problem 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 $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{B}) = \emptyset$.

Language inclusion and universality problems

Theorem. The language inclusion problem and universality problem 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 $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{B}) = \emptyset$.

$$\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2) \iff \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{B}) = \emptyset$$

$$\mathcal{L}(\mathcal{A}_2) = \widetilde{\Sigma}^* \iff \mathcal{L}(\mathcal{B}) = \emptyset$$

The determinization procedure can be fulfilled in EXPTIME.

Language inclusion and universality problems

Theorem. The language inclusion problem and universality problem 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 $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{B}) = \emptyset$.

$$\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2) \iff \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{B}) = \emptyset$$

$$\mathcal{L}(\mathcal{A}_2) = \widetilde{\Sigma}^* \iff \mathcal{L}(\mathcal{B}) = \emptyset$$

The determinization procedure can be fulfilled in EXPTIME. To show EXPTIME-hardness, we show that universality of VPA is EXPTIME-hard.

$$\mathcal{L}(\mathcal{A}) = \widetilde{\Sigma}^* \iff \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$$

where $\mathcal{L}(\mathcal{A}') = \widetilde{\Sigma}^*$

Language inclusion and universality problems

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

The universality of VPA is EXPTIME*-hard.*

Language inclusion and universality problems

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

The universality of VPA is EXPTIME*-hard.*

Result from complexity theory: **APSPACE**= EXPTIME.

Language inclusion and universality problems

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 $M = (Q_\exists, Q_\forall, \Sigma, \Gamma, \delta, q_0, B, F)$ such that

- the state set is divided into two disjoint subsets, $Q_\exists$ ("existential" state), $Q_\forall$ ("universal" 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,
- we assume that for every node (configuration) $\alpha q \beta$ in the tree, if $q \in Q_\exists$, 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 $q \in Q_\forall$, 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 and universality problems

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

The universality of VPA is EXPTIME-hard.

Language inclusion and universality problems

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 $M = (Q_\exists, Q_\forall, \Sigma, \Gamma, \delta, q_0, B, F)$ be an ATM using linear space, say cn.

Language inclusion and universality problems

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 $M = (Q_\exists, Q_\forall, \Sigma, \Gamma, \delta, q_0, B, F)$ be an ATM using linear space, say cn.

- We will construct a VPA $\mathcal{B}$ that accepts all the non-accepting computation histories of M over an input w, then

$$\mathcal{L}(\mathcal{B}) = \widetilde{\Sigma}^* \iff M \text{ does not accept } w$$

- To construct $\mathcal{B}$, we first construct a deterministic VPA $\mathcal{A}$ that accepts all the accepting computation histories of M over an input w, then complement $\mathcal{A}$

Language inclusion and universality problems

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 $M = (Q_\exists, Q_\forall, \Sigma, \Gamma, \delta, q_0, B, F)$ be an ATM using linear space, say cn.

Language inclusion and universality problems

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 $M = (Q_\exists, Q_\forall, \Sigma, \Gamma, \delta, q_0, B, F)$ be an ATM using linear space, say cn.

Let t be an accepting computation history of M over an input w.

Language inclusion and universality problems

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 $M = (Q_\exists, Q_\forall, \Sigma, \Gamma, \delta, q_0, B, F)$ be an ATM using linear space, say cn.

Let t be an accepting computation history 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.

Language inclusion and universality problems

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 $M = (Q_\exists, Q_\forall, \Sigma, \Gamma, \delta, q_0, B, F)$ be an ATM using linear space, say cn.

Let t be an accepting computation history 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.

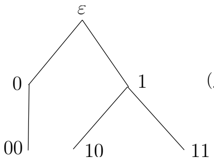
Language inclusion and universality problems

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

The universality of VPA is EXPTIME-hard.

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 $\theta = \theta(b\overline{C_x})^r$.
4. Each leaf is an accepting configuration.



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

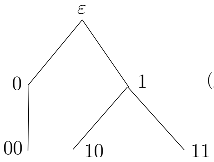
Language inclusion and universality problems

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

The universality of VPA is EXPTIME-hard.

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 $\theta = \theta(b\overline{C_x})^r$.
4. Each leaf is an accepting configuration.



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

Let $\Gamma' = \Gamma \cup Q \cup \overline{\Gamma} \cup \overline{Q} \cup \{f, b\}$, $\widetilde{\Gamma'} = \langle \Gamma \cup Q \cup \{f\}, \overline{\Gamma} \cup \overline{Q} \cup \{b\} \rangle$.

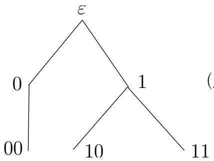
Language inclusion and universality problems

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

The universality of VPA is EXPTIME-hard.

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 $\theta = \theta(b\overline{C_x})^r$.
4. Each leaf is an accepting configuration.



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

Let $\Gamma' = \Gamma \cup Q \cup \overline{\Gamma} \cup \overline{Q} \cup \{f, b\}$, $\widetilde{\Gamma'} = \langle \Gamma \cup Q \cup \{f\}, \overline{\Gamma} \cup \overline{Q} \cup \{b\} \rangle$.
The format of a successful computation θ, *e.g. well-matched call-returns.*

Language inclusion and universality problems

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

Lower bound.

The universality of VPA is EXPTIME-hard.

The set of unsuccessful computations of M

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

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

Equivalence problem

Theorem. The equivalence of VPA is EXPTIME-complete.

Equivalence problem

Theorem. The equivalence of VPA is EXPTIME-complete.

Upper bound.

Invoke two times of inclusion testing.

Equivalence problem

Theorem. The equivalence of VPA is EXPTIME-complete.

Upper bound.

Invoke two times of inclusion testing.

Lower bound.

$$\mathcal{L}(\mathcal{A}) = \widetilde{\Sigma}^* \iff \mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$$

where $\mathcal{L}(\mathcal{A}') = \widetilde{\Sigma}^*$

Summary

Closure Properties

	Union	Intersection	Complement	Concatenation	Kleene-$*$
Regular	YES	YES	YES	YES	YES
CFL	YES	NO	NO	YES	YES
DCFL	NO	NO	YES	NO	NO
VPL	YES	YES	YES	YES	YES

Summary

Closure Properties

	Union	Intersection	Complement	Concatenation	Kleene-$*$
Regular	YES	YES	YES	YES	YES
CFL	YES	NO	NO	YES	YES
DCFL	NO	NO	YES	NO	NO
VPL	YES	YES	YES	YES	YES

Decision problems

	Emptiness	Universality/Equivalence	Inclusion
NFA	**NL**	PSPACE	PSPACE
PDA	**P**	Undecidable	Undecidable
DPDA	**P**	Decidable	Undecidable
VPA	**P**	EXPTIME	EXPTIME