Visibly pushdown languages

Saubj Kumar Jena

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

23 October 2013
Acknowledgment

I am thankful to Prof. Deepak D’Souza for assigning the said seminar topic. I would like to acknowledge Prof. Rajeev Alur and Prof. P. Madhusudan for their paper titled Visibly Pushdown Languages.
References

- Wikipedia
Outline

1. Visibly pushdown automata (VPA)
2. Closure properties
3. Visibly pushdown grammar (VPG)
4. Logical Characterisation
5. Decision Problems
6. Relation to Regular Tree Languages
7. Visibly pushdown $\omega$-languages
Visibly pushdown automata (VPA)

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

- $\Sigma_c$ : finite set of calls,
- $\Sigma_r$ : finite set of returns,
- $\Sigma_l$ : finite set of local actions.

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

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

Note: No $\varepsilon$-transitions, Exactly one symbol is pushed in each call transition.
Visibly pushdown automata (VPA)

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

- $\Sigma_c$ : finite set of calls,
- $\Sigma_r$ : finite set of returns,
- $\Sigma_l$ : finite set of local actions.

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

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

Note:

- No $\varepsilon$-transitions,
- Exactly one symbol is pushed in each call transition.
Visibly pushdown automata (VPA)

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

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

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

- for every $(q, a) \in Q \times \Sigma_c$, there is atmost one pair $(q', \gamma) \in Q \times (\Gamma \setminus \{\bot\})$ such that $(q, a, q', \gamma) \in \delta$
- for every $(q, a, \gamma) \in Q \times \Sigma_r \times \Gamma$, there is atmost one $q' \in Q$ such that $(q, a, \gamma, q') \in \delta$
- for every $(q, a) \in Q \times \Sigma_l$, there is atmost one $q' \in Q$ such that $(q, a, q') \in \delta$

A deterministic VPA is complete if atmost is replaced by exactly.
Visibly pushdown automata (VPA): continued

For a word $w = a_1....a_n$ in $\Sigma^*$, a run of a VPA $A$ over $w$ is a sequence $(q_0, \alpha_0)(q_1, \alpha_1)...(q_n, \alpha_n)$ s.t

- $\forall i. q_i \in Q$,
- $\forall i. \sigma_i \in St$, where $St = (\Gamma \setminus \{\bot\})^*\{\bot\}$ denotes the set of all stacks.
- $\alpha_0 = \bot$,
- $\forall i : 1 \leq i \leq n$, one of the following holds,
  - Call $a_i \in \Sigma_c$, $\exists \gamma \in \Gamma \setminus \{\bot\}$. s.t. $(q_i, a_i, q_{i+1}, \gamma) \in \delta$, $\alpha_{i+1} = \gamma\alpha_i$,
  - Return $a_i \in \Sigma_r$,
    - $\exists \gamma \in \Gamma \setminus \{\bot\}$. s.t. $(q_i, a_i, \gamma, q_{i+1}) \in \delta$, $\alpha_i = \gamma\alpha_{i+1}$,
    - or $(q_i, a_i, \bot, q_{i+1}) \in \delta$, and $\alpha_i = \alpha_{i+1} = \bot$,
  - Local $a_i \in \Sigma_l$, $(q_i, a_i, q_{i+1}) \in \delta$ and $\alpha_{i+1} = \alpha_i$. 

Note: Acceptance by VPAs are defined by final states, not by empty stack.
Visibly pushdown automata (VPA): continued

For a word \( w = a_1 \ldots a_n \) in \( \Sigma^* \), a run of a VPA \( \mathcal{A} \) over \( w \) is a sequence \( (q_0, \alpha_0)(q_1, \alpha_1) \ldots (q_n, \alpha_n) \) s.t

- \( \forall i. \ q_i \in Q, \)
- \( \forall i. \ \sigma_i \in St \), where \( St = (\Gamma \setminus \{\bot\})^* \{\bot\} \) denotes the set of all stacks.
- \( \alpha_0 = \bot, \)
- \( \forall i : 1 \leq i \leq n, \) one of the following holds,
  - Call \( a_i \in \Sigma_c, \ \exists \gamma \in \Gamma \setminus \{\bot\} \) s.t. \( (q_i, a_i, q_{i+1}, \gamma) \in \delta, \alpha_{i+1} = \gamma \alpha_i, \)
  - Return \( a_i \in \Sigma_r, \)
    - \( \exists \gamma \in \Gamma \setminus \{\bot\} \) s.t. \( (q_i, a_i, \gamma, q_{i+1}) \in \delta, \alpha_i = \gamma \alpha_{i+1}, \)
    - or \( (q_i, a_i, \bot, q_{i+1}) \in \delta, \) and \( \alpha_i = \alpha_{i+1} = \bot, \)
  - Local \( a_i \in \Sigma_l, (q_i, a_i, q_{i+1}) \in \delta \) and \( \alpha_{i+1} = \alpha_i. \)

A run \( (q_0, \alpha_0)(q_1, \alpha_1) \ldots (q_n, \alpha_n) \) is accepting if \( q_n \in F. \)

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

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

Note: Acceptance by VPAs are defined by final states, not by empty stack.
Let $\tilde{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$.

The set of well-matched words $w \in \Sigma^*$ is defined inductively as follows,

- $\epsilon$ 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 $w''$ is well matched, then
  $w = w'w''$ is well matched.

**Example:** $(()())$ is well matched, while neither $()())$ nor $(())$ is.
A language $L \subseteq \Sigma^*$ is a *visibly pushdown language with respect to* $\bar{\Sigma}$ (a $\bar{\Sigma}$ -- VPL) if there is a VPA $A$ over $\bar{\Sigma}$, satisfying that $L(A) = L$.

**Example 1:**

The language $\{a^n b^n | n \geq 1\}$ is a VPL with respect to $\bar{\Sigma} = \langle \{a\}, \{b\}, \Phi \rangle$
A language $L \subseteq \Sigma^*$ is a *visibly pushdown language with respect to* $\tilde{\Sigma}$ (a $\tilde{\Sigma}$ – VPL) if there is a VPA $\mathcal{A}$ over $\tilde{\Sigma}$, satisfying that $L(\mathcal{A}) = L$.

**Example 1:**

The language $\{a^n b^n | n \geq 1\}$ is a VPL with respect to $\tilde{\Sigma} = \langle \{a\}, \{b\}, \Phi \rangle$.

Is every CFL a VPL?
Visibly pushdown languages (VPL)

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

**Example 1:**

The language $\{a^n b^n | n \geq 1\}$ is a VPL with respect to $\tilde{\Sigma} = \langle \{a\}, \{b\}, \Phi \rangle$

**Is every CFL a VPL?**

**Example 2:**

The CFL $\{a^n ba^n | n \geq 1\}$ is not a VPL with respect to any partition $\tilde{\Sigma}$ of the alphabet $\Sigma = \{a, b\}$
A language $L \subseteq \Sigma^*$ is a **visibly pushdown language with respect to** $\tilde{\Sigma}$ (a $\tilde{\Sigma}$–VPL) if there is a VPA $A$ over $\tilde{\Sigma}$, satisfying that $L(A) = L$.

**Example 1:**

The language $\{a^n b^n | n \geq 1\}$ is a VPL with respect to $\tilde{\Sigma} = \langle \{a\}, \{b\}, \Phi \rangle$

Is every CFL a VPL?

**Example 2:**

The CFL $\{a^n ba^n | n \geq 1\}$ is not a VPL with respect to any partition $\tilde{\Sigma}$ of the alphabet $\Sigma = \{a, b\}$

*The class of VPLs is a strictly subclass of the class of CFLs.*
Visibly pushdown languages (VPL)

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

Example 1:

The language $\{a^n b^n | n \geq 1\}$ is a VPL with respect to $\tilde{\Sigma} = \langle\{a\}, \{b\}, \Phi\rangle$

Is every CFL a VPL?

Example 2:

The CFL $\{a^n b a^n | n \geq 1\}$ is not a VPL with respect to any partition $\tilde{\Sigma}$ of the alphabet $\Sigma = \{a, b\}$

The class of VPLs is a strictly subclass of the class of CFLs.

But, for every CFL we can associate a VPL over a different alphabet.
Embedding of CFL as VPLs

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

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

W.l.o.g, suppose that each $(q, a, X, \alpha) \in \delta$ satisfies that $\alpha = \epsilon$ (pop) or $\alpha = X$ (stable) or $\alpha = YX$ (push).

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

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

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

Embedding of CFL as VPLs continued

A word \( w = a_1 a_2 ... a_n \) is accepted by PDA \( \mathcal{A} \) iff there is some augmentation \( w' \) of \( w \), \( w' = (a'_1, b_1)(a'_2, b_2).....(a'_m, b_m) \) where each \( b_i \in \{c, r, l\} \) and each \( a'_i \in \Sigma \cup \{\sigma_\epsilon\} \), such that \( w' \) is accepted by \( \mathcal{A}' \).

Let \( h : (\Sigma')^* \rightarrow \Sigma^* \) be a homomorphism defined by \( \forall a \in \Sigma, s \in \{c, r, l\}. \) s.t. \( h((a, s)) = a, h((\sigma_\epsilon, s)) = \epsilon. \) Then \( L = h(L(\mathcal{A}')). \)
Outline

1. Visibly pushdown automata (VPA)
2. Closure properties
3. Visibly pushdown grammar (VPG)
4. Logical Characterisation
5. Decision Problems
6. Relation to Regular Tree Languages
7. Visibly pushdown \( \omega \)-languages
Union and intersection

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

Let $\mathcal{A}_1 = (Q_1, \tilde{\Sigma}, \Gamma_1, \delta_1, q_0^1, \bot_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \tilde{\Sigma}, \Gamma_2, \delta_2, q_0^2, \bot_2, F_2)$ be two VPAs.

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

The VPA $\mathcal{A} = (Q_1 \cup Q_2 \cup q_0, \tilde{\Sigma}, \Gamma_1 \cup \Gamma_2, \delta, q_0, \bot, F_1 \cup F_2)$ s.t.

\[ \delta = \delta_1 \cup \delta_2 \cup \{(q_0, a, q', \gamma) | (q_0^1, a, q', \gamma) \in \delta_1 \text{ or } (q_0^2, a, q', \gamma) \in \delta_2\} \cup \{(q_0, a, \gamma, q') | (q_0^1, a, \gamma, q') \in \delta_1 \text{ or } (q_0^2, a, \gamma, q') \in \delta_2\} \]

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

**Intersection.**

The VPA $\mathcal{A} = (Q_1 \times Q_2, \tilde{\Sigma}, \Gamma_1 \times \Gamma_2, \delta, (q_0^1, q_0^2), (\bot_1, \bot_2), F_1 \times F_2)$ s.t.

\[ \delta = \{((q_1, q_2), a, (q_1', q_2'), (\gamma_1, \gamma_2)) | (q_1, a, q_1', \gamma_1) \in \delta_1, (q_2, a, q_2', \gamma_2) \in \delta_2\} \cup \{((q_1, q_2), a, (\gamma_1, \gamma_2), (q_1', q_2')) | (q_1, a, \gamma_1, q_1') \in \delta_1, (q_2, a, \gamma_2, q_2') \in \delta_2\} \]

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

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

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

**Proof.**

Suppose $L$ is defined by a complete deterministic VPA

$\mathcal{A} = (Q, \tilde{\Sigma}, \Gamma, \delta, q_0, \bot, F)$.

Then $\mathcal{A} = (Q, \tilde{\Sigma}, \Gamma, \delta, q_0, \bot, Q \setminus F)$ defines $\Sigma^* \setminus L(\mathcal{A})$. □
Determinisation of VPA

The construction of the deterministic VPA $\mathcal{A'} = (Q', \tilde{\Sigma}, \Gamma', \delta', q_0, \bot, F')$.

....NOT COMPLETE....
## Summary of Closure Properties

<table>
<thead>
<tr>
<th></th>
<th>Closed Under</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\cup$</td>
</tr>
<tr>
<td>Regular</td>
<td>YES</td>
</tr>
<tr>
<td>CFL</td>
<td>YES</td>
</tr>
<tr>
<td>DCFL</td>
<td>NO</td>
</tr>
<tr>
<td>VPL</td>
<td>YES</td>
</tr>
</tbody>
</table>
Outline

1. Visibly pushdown automata (VPA)
2. Closure properties
3. Visibly pushdown grammar (VPG)
4. Logical Characterisation
5. Decision Problems
6. Relation to Regular Tree Languages
7. Visibly pushdown $\omega$-languages
A CFG $G = (N, \Sigma, P, S)$ is a VPG over $\tilde{\Sigma}$ if $N$ can be partitioned into $N_0$ and $N_1$, and each rule in $P$ is of the following forms,

- $X \rightarrow \epsilon$,
- $X \rightarrow aY$ such that if $X \in N_0$, then $a \in \Sigma_l$, $Y \in N_0$,
- $X \rightarrow aYbZ$ such that $a \in \Sigma_c$, $b \in \Sigma_r$ $Y \in N_0$ and if $X \in N_0$, then $Z \in N_0$.

**Example.** Let $\tilde{\Sigma} = (\{a\}, \{b\}, \Phi)$. Then the VPG

$$S \rightarrow aSbC | aTbC, T \rightarrow \epsilon, C \rightarrow \epsilon,$$

such that $N_0 = \{S, T, C\}$ defines $\{a^nb^n | n \geq 1\}$. 

---

Visibly pushdown grammar (VPG)

---

Visibly pushdown automata (VPA)  Closure properties  Visibly pushdown grammar (VPG)  Logical Characterisation  Decision Problems  Relation to Regular Tree Languages
Equivalence of VPA and VPG

**Theorem.** $\text{VPA} \equiv \text{VPG}$.

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

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

*the top symbol of the stack is $\gamma$, and from state $q$, by reading a well matched word, state $p$ can be reached*
Equivalence of VPA and VPG

**Theorem.** \( \text{VPA} \equiv \text{VPG} \).

From VPA to VPG.

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

Construct a VPG \( (N_0, N_1, \tilde{\Sigma}, P, S) \) as follows.

\[ N = \{(q, \bot)|q \in Q\} \cup \{q|q \in Q\} \cup \{[q, \gamma, p]|q, p \in Q, \gamma \in \Gamma \setminus \{\bot\}\}, \]

- \((q, \bot)\): the state is \( q \) and the stack is empty,
- \( q \): the state is \( q \) and the stack is nonempty.

\[ N_0 = \{[q, \gamma, p]|q, p \in Q, \gamma \in \Gamma \setminus \{\bot\}\}, S = (q_0, \bot), \]

\( P \) is defined by the following rules,

- if \((q, a, q') \in \delta \) s.t. \( a \in \Sigma _l \), then
  \((q, \bot) \rightarrow a(q', \bot), q \rightarrow aq', [q, \gamma, p] \rightarrow a[q', \gamma, p] \)
- if \((q, a, q', \gamma), (p', b, \gamma, p) \in \delta \) s.t. \( a \in \Sigma _c, b \in \Sigma _r \), then
  \([q, \gamma_1, r] \rightarrow a[q', \gamma, p']b[p, \gamma_1, r], (q, \bot) \rightarrow a(q', \gamma, p')b(p, \bot), q \rightarrow a(q', \gamma, p')bp.\)
- if \((q, a, q', \gamma) \in \delta \) s.t. \( a \in \Sigma _c \), then
  \((q, \bot) \rightarrow aq', q \rightarrow aq'(q, \bot) \rightarrow a[q', \gamma, p], q \rightarrow a[q', \gamma, p].\)
- if \((q, a, \bot, q') \in \delta \) s.t. \( a \in \Sigma _r \), then \((q, \bot) \rightarrow a(q', \bot).\)
- \( \forall q \in Q. \ [q, \gamma, q] \rightarrow \varepsilon, \)
- \( \forall q \in F. \ q \rightarrow \varepsilon, (q, \bot) \rightarrow \varepsilon, \)
Equivalence of VPA and VPG: continued

From VPG to VPA.
Let $G = (N_0, N_1, \tilde{\Sigma}, P, S)$ be a VPG.

Construct a VPA $A = (N, \tilde{\Sigma}, \Sigma_r \times N \cup \{\bot, \$\}, \delta, S, F)$ as follows.

- $\delta$ is defined by the following rules,
  - if $X \rightarrow aY$ s.t. $a \in \Sigma_l$, then $(X, a, Y) \in \delta$,
  - if $X \rightarrow aY$ s.t. $a \in \Sigma_c$, then $(X, a, Y, \$) \in \delta$,
  - if $X \rightarrow aY$ s.t. $a \in \Sigma_r$, then $(X, a, \$, Y) \in \delta$ and $(X, a, \bot, Y) \in \delta$,
  - if $X \rightarrow aYbZ$, then $(X, a, Y, (b, Z)) \in \delta$,
  - if $X \rightarrow \epsilon$ and $X \in N_0$, then $(X, b, (b, Y), Y) \in \delta$,

- $A$ accepts if the state is in $X$ s.t. $X \rightarrow \epsilon$ and the top symbol is $\$ or $\bot$. 
Equivalence of VPA and VPG: continued

From VPG to VPA.
Let $G = (N_0, N_1, \tilde{\Sigma}, P, S)$ be a VPG.

Construct a VPA $\mathcal{A} = (N, \tilde{\Sigma}, \Sigma_r \times N \cup \{\bot, \$\}, \delta, S, F)$ as follows.

- $\delta$ is defined by the following rules,
  - if $X \rightarrow aY$ s.t. $a \in \Sigma_l$, then $(X, a, Y) \in \delta$,
  - if $X \rightarrow aY$ s.t. $a \in \Sigma_c$, then $(X, a, Y, \$) \in \delta$,
  - if $X \rightarrow aY$ s.t. $a \in \Sigma_r$, then $(X, a, \$, Y) \in \delta$ and $(X, a, \bot, Y) \in \delta$,
  - if $X \rightarrow aYbZ$, then $(X, a, Y, (b, Z)) \in \delta$,
  - if $X \rightarrow \epsilon$ and $X \in N_0$, then $(X, b, (b, Y), Y) \in \delta$,

- $\mathcal{A}$ accepts if the state is in $X$ s.t. $X \rightarrow \epsilon$ and the top symbol is $\$ or $\bot$.

Adapt $\mathcal{A}$ into $\text{VPA}$

$\mathcal{A} = (N \times \Gamma, \tilde{\Sigma}, \Gamma, \delta', (S, \bot), \{(X, \gamma)|X \rightarrow \epsilon, \gamma = \$, \bot\})$ by adding the top symbol of the stack into the states.

- if $X \rightarrow aY$ s.t. $a \in \Sigma_l$, then $\forall \gamma$. s.t. $((X, \gamma), a, (Y, \gamma)) \in \delta'$,
- if $X \rightarrow aY$ s.t. $a \in \Sigma_c$, then $\forall \gamma$. s.t. $((X, \gamma), a, (Y, \$), (\$, \gamma)) \in \delta'$,
- if $X \rightarrow aY$ s.t. $a \in \Sigma_r$, then $\forall \gamma$. s.t. $((X, \gamma), a, \bot, (Y, \bot)) \in \delta$ and $\forall \gamma$. s.t. $((X, \$, a, (\$, \gamma), (Y, \gamma)) \in \delta'$,
- if $X \rightarrow aYbZ$, then $\forall \gamma$. s.t. $((X, \gamma), a, (Y, (b, Z)), ((b, Z), \gamma)) \in \delta'$,
- if $X \rightarrow \epsilon$ and $X \in N_0$, then $\forall \gamma$. s.t. $((X, (b, Z)), b, ((b, Z), \gamma), (Z, \gamma)) \in \delta'$,
The monadic second order logic MSOμ over \( \tilde{\Sigma} \) is defined as:

\[
\phi := Q_a(x) | x \in X | x \leq y | \mu(x, y) | \phi | \phi \lor \phi | \exists x.\phi | \exists X.\phi
\]

where
- \( a \in \Sigma \)
- \( x \) is a first order variable
- \( X \) is a set variable
- \( Q_a(i) \) is true iff \( w[i] = a \)
- \( \mu(i, j) \) is true if \( w[i] \) is a call and \( w[j] \) is its matching return.

**Theorem** A language \( L \) over \( \tilde{\Sigma} \) is a VPL iff there is an MSOμ sentence \( \phi \) over \( \tilde{\Sigma} \) that defines \( L \)
## Decision Problems

<table>
<thead>
<tr>
<th></th>
<th>Emptynesss</th>
<th>Univ./Equiv.</th>
<th>Inclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td>NLOG</td>
<td>PSPACE</td>
<td>PSPACE</td>
</tr>
<tr>
<td>CFL</td>
<td>PTIME</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td>DCFL</td>
<td>PTIME</td>
<td>Decidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td>VPL</td>
<td>PTIME</td>
<td>EXPTIME</td>
<td>EXPTIME</td>
</tr>
</tbody>
</table>

- **Visibly pushdown automata (VPA)**
- **Closure properties**
- **Visibly pushdown grammar (VPG)**
- **Logical Characterisation**
- **Decision Problems**
- **Relation to Regular Tree Languages**
Outline

1. Visibly pushdown automata (VPA)
2. Closure properties
3. Visibly pushdown grammar (VPG)
4. Logical Characterisation
5. Decision Problems
6. Relation to Regular Tree Languages
7. Visibly pushdown \( \omega \)-languages
Relation to Regular Tree Languages

–NOT COMPLETE–
Outline

1. Visibly pushdown automata (VPA)
2. Closure properties
3. Visibly pushdown grammar (VPG)
4. Logical Characterisation
5. Decision Problems
6. Relation to Regular Tree Languages
7. Visibly pushdown $\omega$-languages
Visibly pushdown \( \omega \)-languages

–NOT COMPLETE–
Queries?
Thanks!

Thanks!!!!