Closure and Determinism
Stack Languages and Predicting Machines
Determmistic union: $L, R$ deterministic $\Rightarrow L \cup R$ deterministic?

- $L, R \in \text{REG} \Rightarrow L \cup R \in \text{REG}$

  simulate deterministic automata in parallel (automata should not block)

- $L, R \in \text{DPD}_\ell \Rightarrow L \cup R \notin \text{DPD}_\ell$

  $\{ a^n b^n \mid n \geq 1 \} \cup \{ a^n b^m c^n \mid m, n \geq 1 \}$

- $L \in \text{DPD}_\ell$, $R \in \text{REG} \Rightarrow L \cup R \in \text{DPD}_\ell$

  parallel simulation with $\lambda$-moves

- avoid infinite $\lambda$-computations: $!!$

  PDA should read all possible input
\( L \in \text{DPD}_\ell, \ R \in \text{REG} \)

\{ \ a^n b^m a^n \mid m, n \in \mathbb{N} \ \} = \\
\{ \ a^{2n} \mid n \in \mathbb{N} \ \} \cup \{ \ a^n b^m a^n \mid m, n \in \mathbb{N}, m \geq 1 \ \}

\( b; Z/Z \)

\( a; +A \)

\( b; A/A \)

\( a; A/\lambda \)

\( b; Z/Z \)

\( \lambda; Z/\lambda \)
\textbf{regular languages} \checkmark

\begin{itemize}
  \item read all input: complete and deterministic
\end{itemize}

\begin{itemize}
  \item context-free languages \xmark
\end{itemize}

\[
\{ a^n b^n c^n \mid n \in \mathbb{N} \} \notin \text{CF}
\]

\begin{itemize}
  \item complement $\{a, b, c\}^* - a^* b^* c^* \\
  \cup \{ a^i b^j c^k \mid i \neq j \} \cup \{ a^i b^j c^k \mid j \neq k \}$
\end{itemize}

\textbf{deterministic cf languages} \checkmark

\begin{itemize}
  \item read all input $\mapsto$ as complete as possible
  \item but $\mapsto$ infinite $\lambda$-computations
\end{itemize}

\textbf{predicting machines}

“can we reach a non-$\lambda$ transition with present stack?”
$L \in \text{DPD}_\ell \Rightarrow L\# \in \text{DPD}_\ell$

‘classic’ construction

\[ L = \text{fag} = \text{fxa}^2 \]

no $\lambda$-instructions leaving final state (a normal form?)

how do we achieve that?

what about single letter quotient?

\[ L/\{a\} = \{ x \mid xa \in L \} \]
We study the language of stacks during computations of a PDA. This language is regular! The proof is a simple consequence of the $[p, A, q]$-construction! Exclamation mark!
Stacks of the pushdown automaton

Stack language

\[ SN(A) = \{ \alpha \in \Gamma^* | (p_{in}, w, \alpha) \vdash^* (q, \lambda, \lambda) \} \]

for some \( w \in \Sigma^* \), some \( q \in Q \}

input \( w \) is irrelevant here

\( B_1B_2\ldots B_n \in SN(A) \)

Build automaton:

\[ (p, w, B) \vdash^* (q, \lambda, \lambda) \]

iff

\[ [p, B, q] \Rightarrow^* w \]

(for some \( w \in \Sigma^* \))

every state initial & final
$\text{SN}(\mathcal{A}) = \{ \alpha \in \Gamma^* \mid (p_{in}, w, \alpha) \vdash^* (q, \lambda, \lambda) \}
$ for some $w \in \Sigma^*$, some $q \in Q$

variant [also regular]

$\{ \ldots \mid \ldots \text{ for some } w \in R, \text{ some } q \in F \}$

$\text{SF}(\mathcal{A}) = \{ \alpha \in \Gamma^* \mid (p_{in}, w, Z_{in}) \vdash^* (q, \lambda, \alpha) \}
$ for some $w \in \Sigma^*$, some $q \in F$
\[ SN(\mathcal{A}) = \{ \alpha \in \Gamma^* \mid (p_{in}, w, \alpha) \vdash^* (q, \lambda, \lambda) \] for some \( w \in \Sigma^* \), some \( q \in F \} \]

Buchi: regular canonical systems

\begin{align*}
\text{type-0 productions } & \alpha \rightarrow \beta \\
\text{prefix rewriting} & \quad \boxed{\alpha} \Rightarrow \boxed{\beta}
\end{align*}

\[ L(\text{rcs}) = \{ w \mid w \Rightarrow^* \lambda \} \]

rcs defines regular language

simulate prefix \( \alpha \rightarrow \beta \) by PDA \([\text{use } F]\)
regular properties of the stack

stack belongs to regular language $R$?

e.g. $R = B(AA + B)^*$

deterministic automaton for reverse

update stack
$B/AB$
$\langle B, 1 \rangle/\langle A, 1 \rangle\langle B, 2 \rangle$
$\langle B, 2 \rangle/\langle A, 2 \rangle\langle B, 1 \rangle$
$\langle B, 3 \rangle/\langle A, 3 \rangle\langle B, 2 \rangle$
$\langle B, g \rangle/\langle A, g \rangle\langle B, g \rangle$

add state info to stack

success $\in R$
$\langle B, 1 \rangle$, $\langle B, 3 \rangle$ on top
- stack language $\text{SN}(A)$ is regular
- a (deterministic) PDA can keep regular info on its stack

$\Rightarrow$ a (deterministic) PDA can predict future behaviour using present stack by inspecting top

 predicting machines

“reaches $A$ the empty stack from my stack?”

<table>
<thead>
<tr>
<th></th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>$B$</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$A$</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>$A$</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
**DPD\(\ell\) closed quotient with REG**

![Quotient Automaton Diagram]

Can we accept extending with \(y \in R\)? stack \(\alpha\) satisfies:

\[(p, y, \alpha) \vdash^*_{\mathcal{A}} (q, \lambda, \beta), \quad q \in F, \ y \in R, \ \text{some} \ \beta\]

\[\text{SN}(A_{p,R}) = \{ \alpha \in \Gamma^* \mid (p, w, \alpha) \vdash^*_{A_{p,R}} (q, \lambda, \lambda) \text{ for some } w \in \Sigma^* \}\]

\(A_{p,R}\) constructed from \(A\)
- initial state \(p\)
- intersection with \(R\) (product construction)
- change to empty stack acceptance

\(Y\): \(\alpha \in \text{SN}(A_{p,R})\)
\(N\): \(\alpha \notin \text{SN}(A_{p,R})\)
as promised: deterministic cf languages ✓

read all input → as complete as possible
but → infinite λ-computations

predicting machines

“can we reach a non-λ transition with present stack?”