Pushdown automata

The e class of machines corresponding to the CFLs.
- Need unbounded memory to go beyond finite-state
- Access to memory is restricted

Input $w$ → Finite state control → Output: yes ($w \in L$) or no ($w \notin L$)

Stock is essentially an NFA

- Stock notation:
  \[
  \begin{array}{c}
  \text{top} \\
  \downarrow \\
  A \quad \text{BBAC} \\
  \uparrow \\
  \text{bottom}
  \end{array}
  \]
  written as $ABBAC$

Pop: returns top symbol (e.g., $A$)
Top symbol removed from stack (e.g., BBAC)
Push($XYZ$): new content is $X$ $Y$ $Z$ $BBAC$

Definition: A PDA is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ with:
- $Q$: states \( \{ q_1, q_2, \ldots \} \) finite
- $\Sigma$: input alphabet \( \{ a, b, c, \ldots \} \) inempty sets
- $\Gamma$: stack alphabet \( \{ A, B, C, \ldots \} \)
- $q_0$: start state \( q_0 \in Q \)
- $z_0$: start stack symbol \( z_0 \in \Gamma \)
- $F$: final states \( F \subseteq Q \)

Note: usually $\Sigma \subseteq \Gamma$, but not necessarily

Notation: we use for strings in $\Sigma^*$: $w, x, y, z, \ldots$
- $\Gamma^*$: $\alpha, \beta, \gamma, \ldots$
Transition function $\delta$:  

Transitions determined by:  
- current state  
- input (or $\varepsilon$-move)  
- top of stack  

Effect:  
- new state  
- pop, and push new string  
- advance on input  

$\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \times 2^\Gamma$  

written $\delta(q, a, X) = \{(q_1, a_1), \ldots, (q_n, a_n)\}$

A transition is executed as follows:  
1) pop stack, top to determine $X$  
2) read input to determine $a$ (unless $a = \varepsilon$)  
3) with current state $q$, select non-deterministically one of the pairs $(q_i, a_i) \in \delta(q, a, X)$  
4) change state to $q_i$  
5) advance past $a$ on input (unless $a = \varepsilon$)  
6) push $a_i$ on top of stack.

Note: initially, stack must contain $2_0$, to allow the first transition to pop the stack. ($\varepsilon$ is not allowed for the stack symbol.)

Graphical representation as transition diagram:  

If $(q_i, a_i) \in \delta(q, a, X)$  

\[
q \xrightarrow{a, X/a_i; \varepsilon} q_i
\]
Example: \( L = \{ 0^n 1^m | m \geq 1 \} \)

PDA M: 
- \( Q = \{ q_0, q_1, q_2 \} \)
- \( \Sigma = \{ 0, 1 \} \)
- \( \Gamma = \{ \lambda, Z_0 \} \)
- \( F = \{ q_2 \} \)

Transitions:

\[ \begin{array}{c}
0, Z_0 / \lambda & \xrightarrow{1, \lambda / \epsilon} & 1, X / \epsilon \\
0, X / \epsilon & \xrightarrow{\epsilon, Z_0 / \epsilon} & q_2 \\
0, X / \epsilon & \xrightarrow{\epsilon, Z_0 / \epsilon} & q_2 \\
0, \lambda / \lambda X \\
\end{array} \]

Idea:
- if input is not in \( 0^*1^* \) then transition will not be defined
- if too few 1's, will not go to final state
- if too many 1's, gets stuck and cannot advance on input

Note: diagram means
\[ \delta(q_0, 0, Z_0) = \{ (q_0, XZ_0) \} \]

Instantaneous description (ID):

\[ ID = \langle q, \alpha, \alpha \rangle \]

- state
- input
- stack

- \( q \in Q \)
- \( \alpha \in \Sigma^* \)
- \( \alpha \in \Gamma^* \)

Transitions described using IDs:

Suppose \( (q_i, \alpha_i) \in \delta(q, \alpha, \lambda) \)

We can say: \( \langle q, \alpha, \lambda \beta \rangle \rightarrow \langle q, \alpha, \alpha \beta \rangle \)

"goes to" (directly)

We write \( ID_1 \rightarrow^* ID_n \) if
\[ ID_1 \vdash ID_2 \vdash ID_3 \vdash \cdots \vdash ID_n \]

Execution trace
Acceptance:

A PDA accepts \( w \) if there is at least one execution trace that leads to a final state when input is finished.

Rejects when:
- no transition is possible ("stuck"), or
- input not over, but stack is empty, or
- input over, but not in final state

(to reject \( w \), for every execution trace, one of these must hold)

Definition: language \( L(M) \) accepted by a PDA

\[
M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)
\]

\[
L(M) = \{ w \in \Sigma^* | (q_0, w, Z_0) \xrightarrow{*} (q_f, \varepsilon, \alpha) \}
\]

with \( q_f \in F \), and \( \alpha \in \Gamma^* \)

Note: nondeterminism is dealt with as for NFA’s

\[
(q_0, w, Z_0) \xrightarrow{*} (q_f, \varepsilon, \alpha)
\]

means that \( \xrightarrow{*} \) can lead to \( (q_f, \varepsilon, \alpha) \)

(provided the right nondeterministic choices are made)

\( L(M) \) is called final state language.

Example: PDA for \( 0^*1^* \) on input 0011

Initial ID: \( ID_0 = (q_0, 0011, Z_0) \)

Execution:

\[
(q_0, 0011, Z_0) \xrightarrow{*} (q_0, 011, XZ_0) \xrightarrow{*} (q_0, 11, XZ_0)
\]

\[
\xrightarrow{*} (q_0, \varepsilon, Z_0) \xrightarrow{*} (q_f, \varepsilon, Z_0)
\]

Thus \( (q_0, 0011, Z_0) \xrightarrow{*} (q_f, \varepsilon, \varepsilon) \)
Note: At end stack was empty, but this cannot happen in-between, as PDA must pop at each transition. (This is why we initialize the stack with $Z_0$)

- We have &-moves that ignore the input, but we must use the symbol on top of the stack
- We can simulate moves that "ignore" top of stack as follows:

\[
\Gamma = \{ X_0, \ldots, X_n \}
\]

\[
\begin{align*}
\Delta : & X_0 \rightarrow X_0 \\
\Delta : & X_n \rightarrow X_n \\
\Delta : & X_i \rightarrow X_j
\end{align*}
\]

\[\text{i.e. whatever is on top of the stack, we move it and then put it back again}\]

We could also define a different notion of acceptance:

\[
N(M) = \{ w \in \Sigma^* | \langle q_0, w, Z_0 \rangle \xrightarrow{*} \langle q, \epsilon, Z \rangle \text{ for any } q \}
\]

\[\text{\(N(M)\) is called empty stack language}\]

Note: for \(N(M)\) we ignore completely final states and accept if at the end of the input word the stack is empty.

Reject when, for every execution trace, one of the following happens:
- Before \(w\) is over, PDA gets stuck
- The stack is empty
- The stack is not empty
Example: \( L_{eqel} = \{ w w^R \mid w \in \{ a, b \}^* \} \)

\[
\begin{align*}
M: & \quad Q = \{ q_0, q_1 \} \\
& \quad \Gamma = \{ A, B, Z_0 \} \\
& \quad \Sigma = \{ a, b \} \\
& \quad F = \emptyset \\
\end{align*}
\]

We want that \( N(M) = L_{eqel} \)

\[a, b / AB \]
\[a, A / AA \]
\[a, Z_0 / AZ_0 \]
\[\varepsilon, Z_0 / \varepsilon \]
\[b, Z_0 / BZ_0 \]
\[b, A / BA \]
\[b, B / BB \]

\[a, A / \varepsilon \]
\[\varepsilon, Z_0 / \varepsilon \]
\[\varepsilon, A / A \]
\[\varepsilon, B / B \]

Idea: 1) push \( w \) onto stack one by one, staying in \( q_0 \)
2) guess mid-point and move to \( q_1 \)
3) in \( q_1 \), match remaining input with stack one by one
4) at end remove \( Z_0 \) from top of stack to accept

Note: in 3, stack pops \( w \) in reverse order

**Exercise E6.1**: Construct execution trace that shows acceptance of \( abbbba \)

**Exercise E6.2**: Construct \( M' \) s.t. \( L(M') = L_{eqel} \)

**Exercise E6.3**: Let \( L_{pol} = \{ w \in \{ a, b \}^* \mid w \text{ is a palindrome} \} \)

Construct \( M_{pol}^{\delta} \) s.t. \( L(M_{pol}^{\delta}) = L_{pol} \)

Construct \( M_{pol}^{\delta} \) s.t. \( N(M_{pol}^{\delta}) = L_{pol} \)
The two acceptance conditions give rise to automata with the same expressive power.

**Theorem:** \( L = \mathcal{L}(M_f) \) for some PDA \( M_f \) \iff \( L = \mathcal{N}(M_{es}) \) for some PDA \( M_{es} \).

**Proof:**

\[ M_f = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \]

\[ M_{es} = (Q \cup \{q_{e1}, q_{e3}\}, \Sigma, \Gamma \cup \{X_0\}, \delta_{es}, q_{e1}, X_0, \emptyset) \]

\[ M_{en} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset) \]

\[ M_e = (Q \cup \{q_{e1}, q_{e3}\}, \Sigma, \Gamma \cup \{X_0\}, \delta_e, q_{e1}, X_0, \{q_{e3}\}) \]

**Question:** Why did we introduce two acceptance conditions?

\( \mathcal{L}(M) \) ... resembles acceptance condition of FSA\( \alpha \)

\( \mathcal{N}(M) \) ... useful to show equivalence between PDAs and CF\( \alpha \)s.
Relationship between PDAs and CFGs:

Let \( L(CFG) \) be the class of languages defined by CFGs.
Let \( L(PDA) \) be the class of final-state languages of PDAs.

\[ L(PDA) = \mathcal{N}(PDA) \quad \text{empty stack} \]

By the previous theorem we know that

\[ L(PDA) = \mathcal{N}(PDA) \]

**Theorem:** \( L(CFG) = \mathcal{N}(PDA) \)

**Hence:** \( \text{CFL} = L(CFG) = \mathcal{N}(PDA) = L(PDA) \)

We only sketch the proof of: \( L(CFG) \subseteq \mathcal{N}(PDA) \)
(for the details and the other direction, see textbook)

Let \( G \) be a CFG. We want to construct a PDA \( M \)
such that \( L(CFG) = \mathcal{N}(PDA) \)

**Idea:** \( M \) simulates \( L\text{-cn} \) derivations of \( G \) for input \( w \)
such that at any derivation step the sentential form
is represented by:

- sequence of symbols of \( w \) already consumed by \( M \)
- followed by contents of \( M \)'s stack

\[ S \overset{*}{\xrightarrow{L\text{-cn}}} abXYCZ \overset{*}{\xrightarrow{L\text{-cn}}} abXYCZ \]

In \( M \):
\[ <q_0, \text{ebXYCZ}, Z_0> \]

consumes prefix \( eb \)
\[ \implies <q_0, XYCZ, \lambda> \]
\[ \implies <q_0, \lambda, \lambda> \]
Construction of $M$:

Let $G = (V_N, V_T, P, S)$

We define $M_0 = \left( Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset \right)$

with $Q = \{ q_0 \}$, i.e. PDA has a single state

$\Sigma = V_T$

$\Gamma = V_T \cup V_N$

$Z_0 = S$

In $\delta$ we have two types of transitions:

1) When $e \in V_T$ is on top of stack, we expect to see $e$ in input and consume both

(sentential form of grammar derivation is not changed)

$\Rightarrow \forall e \in V_T : \delta(q_0, e, e) = \{(q_0, \varepsilon)\}$

2) When $A \in V_N$ is on top of stack, then replace it with RHS of some production for $A$ in $P$

(no input is consumed)

$\Rightarrow \forall A \in V_N : \text{ if } A \rightarrow \alpha_1 | \cdots | \alpha_n \text{ in } P$

then $\delta(q_0, \varepsilon, A) = \{(q_0, \alpha_1), \ldots, (q_0, \alpha_n)\}$

We can show: $\forall y \in V^*_T \quad \forall \beta \in V^*$

$S \rightarrow^* y \beta \text{ in } G \text{ iff }$

$\langle q_0, y \varepsilon, S \rangle \rightarrow^* \langle q_0, \varepsilon, \beta \rangle$ for all $\varepsilon \in V_T^*$

(by induction on derivation length)

For $\beta = \varepsilon$ and $\varepsilon = \varepsilon$ we get the claim. Q.E.D.