Ling 5801: Lecture Notes 8
From CFGs to Pushdown Automata

1. PDAs are FSAs extended with an infinite pushdown store at each possible state

A Pushdown Automaton (PDA) is a tuple \( \langle Q, X, S, F, M \rangle \), where:

- \( Q \) is a finite set of states
- \( X \) is a finite set of observation symbols
- \( S \subseteq Q \) is a set of start states
- \( F \subseteq Q \) is a set of final states
- \( M \subseteq Q \times (Q \cup \{ \epsilon \}) \times (X \cup \{ \epsilon \}) \times (Q \cup \{ \epsilon \}) \) is a set of store,state transitions, of form:
  - \( \langle q, \epsilon, \epsilon, q', q'' \rangle \) — called an ‘expansion’ (or ‘stack push’):
    transition from state \( q \) to \( q' \), replacing empty string \( \epsilon \) at front of store with \( q \)
  - \( \langle q, q'', x, q', q''' \rangle \) — called a ‘state transition’:
    transition from state \( q \) to \( q' \) on observation \( x \), replacing \( q'' \) at front of store with \( q''' \)
  - \( \langle q, q'', \epsilon, q', \epsilon \rangle \) — called a ‘reduction’ (or ‘stack pop/pull’):
    transition from state \( q \in F \) to \( q' \), replacing \( q'' \) at front of store with empty string \( \epsilon \)

From ‘Candyland’ to ‘Trivial Pursuit’: pieces carry stacked-on sub-pieces corresp. to spaces

2. Language accepted by PDA \( A = \langle Q_A, X_A, S_A, F_A, M_A \rangle \)

During recognition, the automaton uses the above store,state transitions to nondeterministically explore all possible states combined with all possible infinite stores.

- The automaton starts with an empty store.
- The automaton pushes and pulls only to/from the front of the store.
- The automaton accepts the input on an empty store at any final state.

\[ L(A) = \{ x_{1..T} \mid q \in S_A, \langle q, \epsilon \rangle \in V_A(x_{1..T}) \} \]

where:

\[ V_A(x_{1..T}) = \{ \langle q, \epsilon \rangle \mid q \in F_A, \ x_{1..T} = \epsilon \} \]

\[ \cup \{ \langle q, \alpha \rangle \mid \langle q, \epsilon, x_i, q', \epsilon \rangle \in M_A, \ \langle q', \alpha \rangle \in V_A(x_{i+1..T}) \} \]

\[ \cup \{ \langle q, \alpha \rangle \mid \langle q, \epsilon, \epsilon, q', q \rangle \in M_A, \ \langle q', q \alpha \rangle \in V_A(x_{1..T}) \} \]

\[ \cup \{ \langle q, q'' \alpha \rangle \mid \langle q, q'', \epsilon, q', \epsilon \rangle \in M_A, \ \langle q', \alpha \rangle \in V_A(x_{1..T}) \} \]

3. Graphical representation of PDAs

PDAs can be represented graphically:

- start states: \( q \in S \)
• final states: \( q \in F \)

\[
\begin{array}{c}
\circlearrowright \ q \\
\end{array}
\]

• state transitions: \( \langle q, \epsilon, x, q', \epsilon \rangle \in M \)

\[
\begin{array}{c}
q \xrightarrow{x} q' \\
\end{array}
\]

• expansions: \( \langle q, \epsilon, \epsilon, q', q \rangle \in M \) (like transitions, but push previous state onto store)

\[
\begin{array}{c}
q \\
\downarrow \\
q' \rightarrow \\
\end{array}
\]

• reductions: \( \langle q, q'', \epsilon, q', \epsilon \rangle \in M, \ q_2 \in F \) (like trans, but remove last state from store)

\[
\begin{array}{c}
q'' \xrightarrow{\epsilon} q' \xrightarrow{\epsilon} q \\
\end{array}
\]

• conditional transitions: \( \langle q, q'', \epsilon, q', q'' \rangle \in M \)

\[
\begin{array}{c}
q'' \xrightarrow{\epsilon} q' \xrightarrow{\epsilon} q \\
\end{array}
\]

4. Example PDA for: \( a^n b^n ; n > 0 \) ( \( S \rightarrow a S b, S \rightarrow a b \) )

\[
\begin{align*}
Q & \{ q_0, q_1, q_2, q_3, q_4 \}, \ X & \{ a, b \}, \ S & \{ q_0 \}, \ F & \{ q_3 \}, \\
M & \{ \langle q_0, \epsilon, a, q_1, \epsilon \rangle, \langle q_2, \epsilon, b, q_3, \epsilon \rangle, \langle q_0, \epsilon, a, q_1, \epsilon \rangle, \langle q_4, \epsilon, b, q_3, \epsilon \rangle, \langle q_1, \epsilon, \epsilon, q_0, q_1 \rangle, \langle q_3, q_1, \epsilon, q_2, \epsilon \rangle \}
\end{align*}
\]
5. **PDAs can recognize CFGs: \( L(CFG) \subseteq L(PDA) \)**

Given any CFG \( \langle C_G, X_G, S_G, R_G \rangle \), we can define a PDA \( \langle Q, X, S, F, M \rangle \) (assume binary-branching CFG, since all CFGs can be translated into one):

\[
Q = \{ q_{c/e}, q_{c/e}, q_c | c \rightarrow d e \in R_G \} \cup \{ q_{x/x}, q_x | x \in X_G \}
\]

\[
X = X_G
\]

\[
S = \{ q_{c/e} | c \in S_G \}
\]

\[
F = \{ q_c | c \rightarrow d e \in R_G \text{ or } c \in X_G \}
\]

\[
M = \{ \langle q_{c/e}, \epsilon, \epsilon, q_{d/d}, q_{c/e} \rangle, \langle q_d, q_{c/e}, \epsilon, q_{c/e}, \epsilon \rangle, \langle q_{c/e}, \epsilon, \epsilon, q_{c/e}, \epsilon \rangle, \langle q_e, q_{c/e}, \epsilon, q_e, \epsilon \rangle | c \rightarrow d e \in R_G \} \cup \{ \langle q_{x/x}, \epsilon, x, q_x, \epsilon \rangle | x \in X_G \}
\]

For example, a grammar with the following rules:

- \( V \rightarrow N \ V - a N \)
- \( V - a N \rightarrow \text{hit } N \)
- \( N \rightarrow \text{the toy} \)

would yield the following PDA:
6. Practice

How would the store look after each expansion, reduction, or transition in the above PDA?

\[ q_{V/V} \] (start state)
\[ q_{NN} \] \( q_{V/N} \) (expand, pushing \( q_{V/V} \) onto store)
\[ \ldots \]?

7. Right-expansion elimination and left-expansion elimination

Easy to implement recognizer if only one expansion, reduction per transition

(a) Right-expansion elimination (‘awaited transition’) – right child ‘c’ disappears:

\[
M^{(0)} = M \\
M^{(k)} = M^{(k-1)} \cup \{ \langle q_{a/c}, \epsilon, \epsilon, q_{d/d}, q_{a/c} \rangle, \langle q_{d}, q_{a/e}, \epsilon, q_{a/e}, \epsilon \rangle, \\
\langle q_{a/e}, \epsilon, \epsilon, q_{e/e}, q_{a/e} \rangle, \langle q_{e}, q_{a/e}, \epsilon, q_{a/e} \rangle, \\
\langle q_{d}, q_{c/e}, \epsilon, q_{a/e}, \epsilon \rangle \in M^{(k-1)}, \langle q_{c/e}, \epsilon, \epsilon, q_{c/d}, q_{a/c} \rangle \in M^{(k-1)}, \\
\langle q_{d}, q_{c/e}, \epsilon, q_{a/e}, \epsilon \rangle \in M^{(k-1)}, \langle q_{c/e}, \epsilon, \epsilon, q_{c/d}, q_{e/e} \rangle \in M^{(k-1)}, \\
\langle q_{e}, q_{c/e}, \epsilon, q_{c/e}, \epsilon \rangle \in M^{(k-1)}, \langle q_{c}, q_{a/c}, \epsilon, q_{a/e}, \epsilon \rangle \in M^{(k-1)} \}
\]

\[ M' = M^{1R} \]

Recognition is equivalent because child sub-models are preserved in order:
for example:

Repeat by mapping $q_{a/e}$, $q_a$ in result to $q_{a/e}$, $q_a$ in subsequent iteration.

Now all sequences of reductions are replaced with a single reduction!

Result (hiding unnecessary structure):

(b) Left-expansion elimination (‘active transition’) – left child ‘c’ disappears:

$$M^{(0)} = M$$
$$M^{(k)} = M^{(k-1)} \cup \{ \langle q_{d/d}, \epsilon, \epsilon, q_{c/c}, q_{a/a} \rangle, \langle q_{d/d}, q_{a/a}, \epsilon, q_{c/e}, q_{a/a} \rangle \mid \langle q_{a/a}, \epsilon, \epsilon, q_{c/c}, q_{a/a} \rangle \in M^{(k-1)}, \langle q_{c/e}, \epsilon, \epsilon, q_{d/d}, q_{c/c} \rangle \in M^{(k-1)},$$
$$\langle q_{d/d}, q_{c/e}, \epsilon, q_{c/e}, \epsilon \rangle \in M^{(k-1)} \}$$

$$M' = M^{[R]}$$

Add ‘conditional’ $\epsilon$-transition: $\langle q_{d/d}, q_{a/a}, \epsilon, q_{c/e}, q_{a/a} \rangle$ (drawn with dotted line)

Recognition is equivalent because child sub-models preserved in order:
for example:

\[
\begin{align*}
V/V &\Rightarrow V/V - aN \\
N/N &\Rightarrow N/toy \\
the/the &\Rightarrow the/toy
\end{align*}
\]

Repeat by mapping \( q_d, q_{a/a'} \) in result to \( q_c, q_{a/a'} \) in subsequent iteration.

Now all sequences of expansions are replaced with a single expansion!

Result (hiding unnecessary structure):

\[
M' = M \cup \{ (q, \epsilon, x, q', \epsilon) \mid (q, \epsilon, \epsilon, q_{x/x}, q), (q_{x/x}, \epsilon, x, q, \epsilon), (q, q, q', \epsilon, \epsilon) \in M, q' \in F \}
\]
8. Implementation of PDA

Since only one expansion/reduction between transitions, we only need four combinations:

a) \( q_{t-1} \rightarrow q_t \) (no expand + trans + no reduce)

\[
\begin{array}{ccc}
\vdots & \vdots & \vdots \\
q_{t-1}^d & f_{t}^d & q_{t}^d \\
\text{x}_t & \text{} & \text{} \\
\end{array}
\]

b) \( q_{t-1}^d \rightarrow q_{t}^{d+1} \) (expand + trans + no reduce)

\[
\begin{array}{ccc}
\vdots & \vdots & \vdots \\
q_{t-1}^{d+1} & f_{t}^{d+1} & q_{t}^{d+1} \\
\text{x}_t & \text{} & \text{} \\
\end{array}
\]
Example:

Transforming the following CFG:

Correctness can be shown from definition of accepted languages.

But, iterating over all possible infinite stores $\sigma$ results in exponential complexity.
\[
T \rightarrow V T \quad \text{(a top-level discourse type)}
\]
\[
V \rightarrow N V-aN
\]
\[
V-aN \rightarrow V-aN \ R-aN
\]
\[
V-aN \rightarrow \text{hit } N
\]
\[
R-aN \rightarrow \text{off } N
\]
\[
A-aN \rightarrow \text{off } N
\]
\[
N \rightarrow N \ A-aN
\]
\[
N \rightarrow \text{the cat}
\]
\[
N \rightarrow \text{the toy}
\]
\[
N \rightarrow \text{the mat}
\]

then recognizing 'the cat hit the toy off the mat' results in the following sequence:

\[
\begin{array}{cccccccccc}
t=0 & t=1 & t=2 & t=3 & t=4 & t=5 & t=6 & t=7 \\
\text{the} & \text{the} & \text{cat} & \text{hit} & \text{hit} & \text{the} & \text{the} \\
\text{hit} & \text{the} & \text{cat} & \text{toy} & \text{off} & \text{the} & \text{the} \\
\text{the} & \text{the} & \text{the} & \text{toy} & \text{the} & \text{the} & \text{the} \\
\end{array}
\]