Pushdown Automata (PDA)

The structure and the content of the lecture is based on http://www.eecs.wsu.edu/~ananth/CptS317/Lectures/index.htm
Excursion

Context-free grammar $G = (V_N, V_T, S, P)$, where:

- $V_N$: set of non-terminals
- $V_T$: set of terminals
- $P$: set of productions, each of which is of the form $V \Rightarrow \alpha_1 | \alpha_2 | \ldots$
  - Where each $\alpha_i$ is an arbitrary string of nonterminals and terminals
- $S$: starting symbol
PDA - the automata for CFLs

- What is?
  - What FA is to Reg Lang, PDA is to CFL
- PDA == $[\varepsilon\text{-NFA} + \text{“a stack”}]$
- Why a stack?

![Diagram of PDA]

- Input string
- $\varepsilon$-NFA
- Accept/reject
- A stack filled with “stack symbols”
Pushdown Automata - Definition

A PDA $P := (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$:

- $Q$: states of the $\varepsilon$-NFA
- $\Sigma$: input alphabet
- $\Gamma$: stack symbols
- $\delta$: transition function
- $q_0$: start state
- $Z_0$: Initial stack top symbol
- $F$: Final/accepting states
δ(q,a,X) = {(p,Y), …}

1. state transition from q to p
2. a is the next input symbol
3. X is the current stack top symbol
4. Y is the replacement for X; it is in \( \Gamma^* \) (a string of stack symbols)
   i. Set \( Y = \varepsilon \) if \( \text{Pop}(X) \)
   ii. If \( Y = X \) then stack top is unchanged
   iii. If \( Y = Z_1Z_2…Z_k \) then X is popped and is replaced by Y in reverse order (i.e., \( Z_1 \) will be the new stack top)
Example (palindrome)

Let $L_{wwr} = \{ww^R \mid w \text{ is in } \{0,1\}^*\}$

- CFG for $L_{wwr}$: $S \rightarrow 0S0 \mid 1S1 \mid \varepsilon$
- PDA for $L_{wwr}$:

$$P := (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

$$= (\{q_0, q_1, q_2\}, \{0,1\}, \{0,1,Z_0\}, \delta, q_0, Z_0, \{q_2\})$$

Mark the bottom of the stack
PDA for $L_{wwr}$

1. $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$
2. $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$
   \{ First symbol push on stack \}

3. $\delta(q_0, 0, 0) = \{(q_0, 00)\}$
4. $\delta(q_0, 0, 1) = \{(q_0, 01)\}$
5. $\delta(q_0, 1, 0) = \{(q_0, 10)\}$
6. $\delta(q_0, 1, 1) = \{(q_0, 11)\}$
   \{ Grow the stack by pushing new symbols on top of old \ (w-part) \}

7. $\delta(q_0, \varepsilon, 0) = \{(q_1, 0)\}$
8. $\delta(q_0, \varepsilon, 1) = \{(q_1, 1)\}$
9. $\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}$
   \{ Switch to popping mode, nondeterministically \ (boundary between $w$ and $w^R$) \}

10. $\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$
11. $\delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$
   \{ Shrink the stack by popping matching symbols \ ($w^R$-part) \}

12. $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$
   \{ Enter acceptance state \}
PDA as a state diagram

$$\delta(q_i, a, X) = \{(q_j, Y)\}$$
PDA for $L_{wwr}$: Transition Diagram

$\Sigma = \{0, 1\}$
$\Gamma = \{Z_0, 0, 1\}$
$Q = \{q_0, q_1, q_2\}$

Grow stack

Switch to popping mode

Pop stack for matching symbols

Go to acceptance

Non-deterministic PDA: 2 output transitions i.e.:
$(q_0, 0, 0) = (q_0, 0), (q_0, \varepsilon, 0) = (q_1, 0)$:

- Push input symbols onto the stack
- Non-deterministically move to a popping state (with or without consuming a single input symbol)
- If next input symbol is same as top of stack, pop
- If $Z_0$ on top of stack move to accept state
How does the PDA for $L_{wwr}$ work on input “1111”?

All moves made by the non-deterministic PDA

Instantaneous Description (ID)

$q_0, 111, 1Z_0$  
$q_0, 1111, Z_0$  
$q_0, 11, 11Z_0$  
$q_0, 111Z_0$  
$q_0, \varepsilon, 1111Z_0$  
$q_1, 1111Z_0$  
$q_1, 111Z_0$  
$q_1, 11Z_0$  
$q_1, \varepsilon, Z_0$  
$q_2, \varepsilon, Z_0$

Acceptance by final state:

= empty input AND final state
Example 2: language of balanced paranthesis

\[ \Sigma = \{ (, ) \} \]

\[ \Gamma = \{ Z_0, ( \} \]

\[ Q = \{ q_0, q_1, q_2 \} \]

On seeing a ( push it onto the stack
On seeing a ) pop if a ( is in the stack

To allow adjacent blocks of nested paranthesis
Example 2: language of balanced paranthesis (another design)

$\Sigma = \{ (, ) \}$
$\Gamma = \{ Z_0, ( ) \}$
$Q = \{ q_0, q_1 \}$

$\text{start} \quad q_0 \quad \varepsilon, Z_0 \rightarrow Z_0 \quad \varepsilon, Z_0 \rightarrow Z_0 \quad q_1$
There are two types of PDAs that one can design: those that accept by final state or by empty stack.

Acceptance by...

1. **PDAs that accept by final state:**
   - For a PDA \( P \), the language accepted by \( P \), denoted by \( L(P) \) by final state, is:
     - \( \{w \mid (q_0,w,Z_0) \rhd^{*} (q,\varepsilon, A) \} \), s.t., \( q \in F \)

2. **PDAs that accept by empty stack:**
   - For a PDA \( P \), the language accepted by \( P \), denoted by \( N(P) \) by empty stack, is:
     - \( \{w \mid (q_0,w,Z_0) \rhd^{*} (q,\varepsilon, \varepsilon) \} \), for any \( q \in Q \).

Q) Does a PDA that accepts by empty stack need any final state specified in the design?
Example: L of balanced parenthesis

PDA that accepts by final state

P_F:
\[(,Z_0 / ( Z_0, ( / ( (, ( / \varepsilon
\]

start

\[\varepsilon, Z_0 / Z_0\]

q_0

[\varepsilon, Z_0 / \varepsilon

\]

q_1

An equivalent PDA that accepts by empty stack

P_N:
\[(,Z_0 / ( Z_0, ( / ( (, ( / \varepsilon, \varepsilon, Z_0 / \varepsilon
\]

start

\[\varepsilon, Z_0 / Z_0\]

q_0

\]

How will these two PDAs work on the input: ( ( ( ) ) ( ) ) ( )
Equivalence of PDAs and CFGs
CFGs == PDAs ==> CFLs
Converting CFG to PDA

**Main idea:** The PDA simulates the leftmost derivation on a given \( w \), and upon consuming it fully it either arrives at acceptance (by empty stack) or non-acceptance.

This is same as: “implementing a CFG using a PDA”
Converting a CFG into a PDA

Main idea: The PDA simulates the leftmost derivation on a given $w$, and upon consuming it fully it either arrives at acceptance (by empty stack) or non-acceptance.

Steps:

1. Push the right hand side of the production onto the stack, with leftmost symbol at the stack top
2. If stack top is the leftmost variable, then replace it by all its productions (each possible substitution will represent a distinct path taken by the non-deterministic PDA)
3. If stack top has a terminal symbol, and if it matches with the next symbol in the input string, then pop it.

This is same as: “implementing a CFG using a PDA”
**Formal construction of PDA from CFG**

- **Given:** $G = (V_N, V_T, S, P)$
- **Output:** $P_N = (\{q\}, V_T, V_N \cup V_T, \delta, q, S)$
- $\delta$:
  - For all $A \in V_N$, add the following transition(s) in the PDA:
    - $\delta(q, \varepsilon, A) = \{ (q, \alpha) | "A \rightarrow \alpha\in P\}$
  - For all $a \in V_T$, add the following transition(s) in the PDA:
    - $\delta(q, a, a) = \{ (q, \varepsilon) \}$

Note: Initial stack symbol (S) same as the start variable in the grammar.
Example: CFG to PDA

- $G = (\{S,A\}, \{0,1\}, P, S)$
- $P$:
  - $S \rightarrow AS \mid \epsilon$
  - $A \rightarrow 0A1 \mid A1 \mid 01$
- $PDA = (\{q\}, \{0,1\}, \{0,1,A,S\}, \delta, q, S)$
- $\delta$:
  - $\delta(q, \epsilon, S) = \{(q, AS), (q, \epsilon)\}$
  - $\delta(q, \epsilon, A) = \{(q,0A1), (q,A1), (q,01)\}$
  - $\delta(q, 0, 0) = \{(q, \epsilon)\}$
  - $\delta(q, 1, 1) = \{(q, \epsilon)\}$

How will this new PDA work?

Let's simulate string 0011
Simulating string 0011 on the new PDA ...

PDA (δ):

δ(q,ε,S) = { (q,AS), (q,ε) }
δ(q,ε,A) = { (q,0A1), (q,A1), (q,01) }
δ(q,0,0) = { (q,ε) }
δ(q,1,1) = { (q,ε) }

Stack moves (shows only the successful path):

S => AS => 0A1S => 0011S => 0011

Leftmost deriv.:

S => AS
=> 0A1S
=> 0011S
=> 0011

Accept by empty stack
Summary

- PDA
  - Definition
  - With acceptance – by final state
  - With acceptance – by empty stack
- PDA (by final state) = PDA (by empty stack) \(\leq\) CFG