Pushdown Automata (PDA)

Reading: Chapter 6
Introduction

- Pushdown automata are used to determine what can be computed by machines.
- More capable than finite-state machines but less capable than Turing machines.
- A type of automaton that uses a stack.
- A pushdown automaton (PDA) differs from a finite state machine in two ways:
  - It can use the top of the stack to decide which transition to take.
  - It can manipulate the stack as part of performing a transition.
PDA - the automata for CFLs

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

- Why a stack?

Diagram:
- Input string
- \(\varepsilon\text{-NFA}\)
- Accept/reject
- A stack filled with \text{“stack symbols”}
PDA

- PDA reads a given input string from left to right.
- In each step, it chooses a transition by indexing a table by input symbol, current state, and the symbol at the top of the stack.
- A PDA can also manipulate the stack, as part of performing a transition.
  - The manipulation can be to push a particular symbol to the top of the stack
  - Pop off the top of the stack.
  - The automaton can alternatively ignore the stack, and leave it as it is.
Given an input symbol, current state, and stack symbol, the automaton can follow a transition to another state, and optionally manipulate (push or pop) the stack.

- If at most one such transition action is possible, then the automaton is called a deterministic pushdown automaton (DPDA).
- If several actions are possible, then the automaton is called nondeterministic, (NPDA).

In case of NPDA, if one of the action leads to an accepting state after reading the complete input string, then language is accepted by the automaton.
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 Γ* (a string of stack symbols)

i. Set Y = ε for: Pop(X)
   Action: Y = ε

ii. If Y = X: stack top is unchanged
   Action: Y = X

iii. If Y = Z₁Z₂…Zₖ: X is popped and is replaced by Y in reverse order (i.e., Z₁ will be the new stack top)
   Action: Y = Z₁Z₂…Zₖ

Non-determinism
Example

Let \( L_{\text{wwr}} = \{ w w^R | w \text{ is in } (0+1)^* \} \)

- CFG for \( L_{\text{wwr}} \) : 
  \[ S \rightarrow 0S0 | 1S1 | \varepsilon \]

- PDA for \( L_{\text{wwr}} \) :

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

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

Initial state of the PDA:
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

0, $Z_0/0Z_0$
1, $Z_0/1Z_0$
0, 0/00
0, 1/01
1, 0/10
1, 1/11

Pop stack for matching symbols

0, 0/ $\epsilon$
1, 1/ $\epsilon$

Switch to popping mode

$\epsilon$, $Z_0/Z_0$
$\epsilon$, 0/0
$\epsilon$, 1/1

Go to acceptance

$\epsilon$, $Z_0/Z_0$

This would be a non-deterministic PDA
PDA’s Instantaneous Description (ID)

A PDA has a configuration at any given instance: 
\((q,w,y)\)

- \(q\) - current state
- \(w\) - remainder of the input (i.e., unconsumed part)
- \(y\) - current stack contents as a string from top to bottom of stack

If \(\delta(q,a,X) = \{(p,A)\}\) is a transition, then the following are also true:

- \((q,a,X) |\rightarrow (p,\varepsilon,A)\)
- \((q,aw,XB) |\rightarrow (p,w,AB)\)

|\rightarrow| sign is called a “turnstile notation” and represents one move

|\rightarrow*| sign represents a sequence of moves
How does the PDA for $L_{wwr}$ work on input “1111”?

All moves made by the non-deterministic PDA

Path dies…

Path dies…

Path dies…

Path dies…

Acceptance by final state:

= empty input AND final state
There are two types of PDAs that one can design: those that accept by **final state** or by **empty stack**.

### Acceptance by...

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

- **PDAs that accept by **empty stack**:**
  - For a PDA $P$, the language accepted by $P$, denoted by $N(P)$ by **empty stack**, is:
    - $\{w \mid (q_0,w,Z_0) \xrightarrow{---^*} (q,\varepsilon, \varepsilon) \}$, 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

\[ \text{P}_F: \]
\[ (,Z_0 / ( Z_0 \]
\[ (, / ( ( \]
\[ , / \varepsilon \]

PDA with transitions:
- Start in state \( q_0 \)
- Transition on \( \varepsilon, Z_0 / Z_0 \) from \( q_0 \) to \( q_1 \)
- Transition on \( \varepsilon, Z_0 / Z_0 \) from \( q_1 \) back to \( q_0 \)

An equivalent PDA that accepts by empty stack

\[ \text{P}_N: \]
\[ (,Z_0 / ( Z_0 \]
\[ (, / ( ( \]
\[ , / \varepsilon \]
\[ \varepsilon, Z_0 / \varepsilon \]

PDA with transitions:
- Start in state \( q_0 \)
- Transition on \( \varepsilon, Z_0 / Z_0 \) from \( q_0 \) to \( q_0 \)

How will these two PDAs work on the input: \( ( ( ( ) ) ( ) ) ( ) \)
PDAs accepting by final state and empty stack are equivalent

- $P_F \leq PDA$ accepting by final state
  - $P_F = (Q_F, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$

- $P_N \leq PDA$ accepting by empty stack
  - $P_N = (Q_N, \Sigma, \Gamma, \delta_N, q_0, Z_0)$

**Theorem:**

- $(P_N \Rightarrow P_F)$ For every $P_N$, there exists a $P_F$ s.t. $L(P_F) = L(P_N)$

- $(P_F \Rightarrow P_N)$ For every $P_F$, there exists a $P_N$ s.t. $L(P_F) = L(P_N)$
**How to convert an empty stack PDA into a final state PDA?**

**$P_N \Rightarrow P_F$ construction**

- Whenever $P_N$’s stack becomes empty, make $P_F$ go to a final state without consuming any addition symbol.
- **To detect empty stack in $P_N$:** $P_F$ pushes a new stack symbol $X_0$ (not in $\Gamma$ of $P_N$) initially before simulating $P_N$.

**Diagram:**

$P_F = (Q_F \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$
Example: Matching parenthesis “(” “)"

**P_N:** 
$( \{q_0\}, \{(\),\}\{Z_0,Z_1\}, \delta_N, q_0, Z_0)$

**δ_N:**
- $\delta_N(q_0,(Z_0) = \{ (q_0,Z_1Z_0) \}$
- $\delta_N(q_0,(Z_1) = \{ (q_0, Z_1Z_1) \}$
- $\delta_N(q_0),Z_1) = \{ (q_0, \epsilon) \}$
- $\delta_N(q_0, \epsilon,Z_0) = \{ (q_0, \epsilon) \}$

**P_f:** 
$( \{p_0,q_0,p_f\}, \{(\),\}\{X_0,Z_0,Z_1\}, \delta_f, p_0, X_0, p_f)$

**δ_f:**
- $\delta_f(p_0, \epsilon,X_0) = \{ (q_0,Z_0) \}$
- $\delta_f(q_0,(Z_0) = \{ (q_0,Z_1 Z_0) \}$
- $\delta_f(q_0,(Z_1) = \{ (q_0, Z_1Z_1) \}$
- $\delta_f(q_0),Z_1) = \{ (q_0, \epsilon) \}$
- $\delta_f(q_0, \epsilon,Z_0) = \{ (q_0, \epsilon) \}$
- $\delta_f(p_0, \epsilon,X_0) = \{ (p_f, X_0) \}$

Accept by empty stack

Accept by final state
How to convert an final state PDA into an empty stack PDA?

**P_F===> P_N construction**

- **Main idea:**
  - Whenever P_F reaches a final state, just make an ε-transition into a new end state, clear out the stack and accept.
  - **Danger:** What if P_F design is such that it clears the stack midway without entering a final state?
    - to address this, add a new start symbol X₀ (not in Γ of P_F)

\[
P_N = (Q \cup \{p_0,p_e\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)
\]

**P_N:**

- New start
- q₀: ε, X₀/Z₀X₀
- P_F: ε, any/ε
- p_e: ε, any/ε
Equivalence of PDAs and CFGs
CFGs == PDAs ==> CFLs

CFG

? 

PDA by final state

≡

PDA by empty stack
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

State is inconsequential (only one state is needed)
Formal construction of PDA from CFG

- **Given:** $G = (V, T, P, S)$
- **Output:** $P_N = (\{q\}, T, V \cup T, \delta, q, S)$
- $\delta$:
  - For all $A \in V$, add the following transition(s) in the PDA:
    - $\delta(q, \varepsilon, A) = \{(q, \alpha) \mid "A \Rightarrow \alpha" \in P\}$
  - For all $a \in T$, 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 ==> AS | \(\varepsilon\)
  - A ==> 0A1 | A1 | 01
- **PDA = (\{q\}, \{0,1\}, \{0,1,A,S\}, \delta, q, S\)**
- **\(\delta\):**
  - \(\delta(q, \varepsilon, S) = \{(q, AS), (q, \varepsilon)\}\)
  - \(\delta(q, \varepsilon, A) = \{(q,0A1), (q,A1), (q,01)\}\)
  - \(\delta(q, 0, 0) = \{(q, \varepsilon)\}\)
  - \(\delta(q, 1, 1) = \{(q, \varepsilon)\}\)

How will this new PDA work?

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

PDA (\(\delta\)):
\[
\begin{align*}
\delta(q, \varepsilon, S) &= \{ (q, AS), (q, \varepsilon) \} \\
\delta(q, \varepsilon, A) &= \{ (q,0A1), (q,A1), (q,01) \} \\
\delta(q, 0, 0) &= \{ (q, \varepsilon) \} \\
\delta(q, 1, 1) &= \{ (q, \varepsilon) \}
\end{align*}
\]

Stack moves (shows only the successful path):

\[
\begin{align*}
S &\Rightarrow AS \\
&\Rightarrow 0A1S \\
&\Rightarrow 0011S \\
&\Rightarrow 0011
\end{align*}
\]

Accept by empty stack
Converting a PDA into a CFG

**Main idea:** Reverse engineer the productions from transitions

If $\delta(q,a,Z) \Rightarrow (p, Y_1 Y_2 Y_3 \ldots Y_k)$:

1. State is changed from $q$ to $p$;
2. Terminal $a$ is consumed;
3. Stack top symbol $Z$ is popped and replaced with a sequence of $k$ variables.

**Action:** Create a grammar variable called $\llbracket qZp \rrbracket$ which includes the following production:

$$ \llbracket qZp \rrbracket \Rightarrow a[p Y_1 q_1] [q_1 Y_2 q_2] [q_2 Y_3 q_3] \ldots [q_{k-1} Y_k q_k] $$

**Proof discussion (in the book)**
Example: Bracket matching

To avoid confusion, we will use $b=\text{"("}$ and $e=\text{")"}$

$P_N: \ ( \{q_0\}, \{b,e\}, \{Z_0,Z_1\}, \delta, \ q_0, \ Z_0 \ )$

1. $\delta(q_0,b,Z_0) = \{ (q_0,Z_1Z_0) \}$
2. $\delta(q_0,b,Z_1) = \{ (q_0,Z_1Z_1) \}$
3. $\delta(q_0,e,Z_1) = \{ (q_0, \varepsilon) \}$
4. $\delta(q_0, \varepsilon ,Z_0) = \{ (q_0, \varepsilon) \}$

Let $A=[q_0Z_0q_0]$
Let $B=[q_0Z_1q_0]$

0. $S \Rightarrow [q_0Z_0q_0]$
1. $[q_0Z_0q_0] \Rightarrow b \ [q_0Z_1q_0] \ [q_0Z_0q_0]$
2. $[q_0Z_1q_0] \Rightarrow b \ [q_0Z_1q_0] \ [q_0Z_1q_0]$
3. $[q_0Z_1q_0] \Rightarrow \varepsilon$

Simplifying,

0. $S \Rightarrow b \ B \ S \ | \ \varepsilon$
1. $B \Rightarrow b \ B \ B \ | \ \varepsilon$

If you were to directly write a CFG:

$S \Rightarrow b \ S \ e \ S \ | \ \varepsilon$

$A \Rightarrow \varepsilon$
Two ways to build a CFG

Build a PDA → Construct CFG from PDA (indirect)

Derive CFG directly (direct)

Similarly…

Two ways to build a PDA

Derive a CFG → Construct PDA from CFG (indirect)

Design a PDA directly (direct)
Deterministic PDAs
This PDA for $L_{wwr}$ is non-deterministic

Grow stack

- $0, Z_0/0Z_0$
- $1, Z_0/1Z_0$
- $0, 0/00$
- $0, 1/01$
- $1, 0/10$
- $1, 1/11$

Switch to popping mode

- $\epsilon, Z_0/Z_0$
- $\epsilon, 0/0$
- $\epsilon, 1/1$

Pop stack for matching symbols

- $0, 0/\epsilon$
- $1, 1/\epsilon$

Accepts by final state

- $\epsilon, Z_0/Z_0$

Why does it have to be non-deterministic?

To remove guessing, impose the user to insert c in the middle
Example shows that: Nondeterministic PDAs ≠ D-PDAs

D-PDA for $L_{wcwr} = \{wcw^R | c \text{ is some special symbol not in } w\}$

- **Grow stack**
  - 0, $Z_0/0Z_0$
  - 1, $Z_0/1Z_0$
  - 0, 0/00
  - 0, 1/01
  - 1, 0/10
  - 1, 1/11

- **Pop stack for matching symbols**
  - 0, 0/ε
  - 1, 1/ε

- **Switch to popping mode**
  - c, $Z_0/Z_0$
  - c, 0/0
  - c, 1/1

- **Accepts by final state**
  - ε, $Z_0/Z_0$

Note:
- All transitions have become deterministic.
A PDA is \textit{deterministic} if and only if:

1. $\delta(q, a, X)$ has \textit{at most one} member for any $a \in \Sigma \cup \{\varepsilon\}$

$\Rightarrow$ If $\delta(q, a, X)$ is non-empty for some $a \in \Sigma$, then $\delta(q, \varepsilon, X)$ must be empty.
PDA vs DPDA vs Regular languages

- Regular languages
- D-PDA
- non-deterministic PDA

$L_{wewr}$

$L_{wwr}$
Summary

- PDAs for CFLs and CFGs
  - Non-deterministic
  - Deterministic
- PDA acceptance types
  1. By final state
  2. By empty stack
- PDA
  - IDs, Transition diagram
- Equivalence of CFG and PDA
  - CFG => PDA construction
  - PDA => CFG construction