Introduction to the Theory of Computation

Lecture 18: PDA ↔ CFG

Tiago Cogumbreiro
Today we will learn...

- Exercises on designing a PDA
- Convert a PDA into a CFG
- Convert a CFG into a PDA

Section 2.2
Supplementary material: Professor David Chiang's lecture notes [1] [2]; Professor Siu On Chan slides
Exercise 1
Exercise 1

1. aa is a palindrome
2. aba is a palindrome
3. bbb is a palindrome
4. $\epsilon$ is a palindrome
5. a is a palindrome

Give a PDA that recognizes palindromes and show it accepts aba and rejects abb
Exercise palindrome

- Transition 1: \( q \rightarrow q_1 \) with rules: 
  - \( \epsilon, \epsilon \rightarrow \$ \)
  - \( a, \epsilon \rightarrow a \)
  - \( b, \epsilon \rightarrow b \)

- Transition 2: \( q_1 \rightarrow q_2 \) with rules: 
  - \( \epsilon, \epsilon \rightarrow \epsilon \)
  - \( a, \epsilon \rightarrow \epsilon \)
  - \( b, \epsilon \rightarrow \epsilon \)

- Transition 3: \( q_2 \rightarrow q_3 \) with rules: 
  - \( \epsilon, \$ \rightarrow \epsilon \)

- Transition 4: \( q_3 \rightarrow q_3 \) with rules: 
  - \( a, a \rightarrow \epsilon \)
  - \( b, b \rightarrow \epsilon \)
Accepts $aba$
Accepts aba
Rejects abb
Rejects $abb$

Diagram of a pushdown automaton (PDA) for the language $abb$. The diagram shows transitions for symbols $a$, $b$, and $\epsilon$.
Exercise 2
Exercise 2

$L_2 = \{a^n b^{2n} \mid n \geq 0\}$

Give a PDA that recognizes $L_2$ and show it rejects aba and accepts abb
Exercise 2 solution

\[\begin{align*}
q & \xrightarrow{\epsilon, \epsilon \rightarrow \epsilon} q_1 \xrightarrow{a, \epsilon \rightarrow a} q_2 \xrightarrow{b, a \rightarrow \epsilon} q_3 \xrightarrow{b, \epsilon \rightarrow \epsilon} q_4 \xrightarrow{\epsilon, \epsilon \rightarrow \epsilon} q_2
\end{align*}\]
$L_2$ does not contain aba
$L_2$ does not contain $aba$
\( L_2 \) contains \( abb \)
$L_2$ contains \texttt{abb}
Context Free Languages
Main result

Context free languages

**Theorem:** Language $L$ has a context free grammar if, and only if, $L$ is recognized by some pushdown automaton.

Next

1. We show that from a CFG we can build an equivalent\(^\dagger\) PDA
2. We show that from a PDA we can build an equivalent\(^\dagger\) CFG

\(^\dagger\) Equivalence with respect to recognized languages. Let $P$ be a PDA and $C$ a CFG we say that $P$ is equivalent to $C$ (and vice versa) if, and only if, $L(P) = L(C)$
Converting a CFG into a PDA
Converting a CFG into a PDA

- (0) Push the sentinel $ to the stack
- (1) Push the initial variable $S$ to the stack
- In a loop:
  - (2) Every rule $S \rightarrow w$ corresponds to popping $S$ and pushing $w$ (in reverse)
  - (3) Pop terminals from stack
  - (4) Empty stack means recognized

Example $L_3 = \{a^n b^n \mid n \geq 0\}$

$S \rightarrow aSb \mid \epsilon$

<table>
<thead>
<tr>
<th>PDA operation</th>
<th>Output</th>
<th>Accept?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0) $, \epsilon \rightarrow $</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>(1) $, \epsilon \rightarrow S$</td>
<td>$S$</td>
<td></td>
</tr>
<tr>
<td>(2) $, S \rightarrow aSb$</td>
<td>aSb$</td>
<td></td>
</tr>
<tr>
<td>(3) $, a \rightarrow \epsilon$</td>
<td>Sb$</td>
<td>a</td>
</tr>
<tr>
<td>(2) $, S \rightarrow aSb$</td>
<td>aSbb$</td>
<td>a</td>
</tr>
<tr>
<td>(3) $, a \rightarrow \epsilon$</td>
<td>Sbb$</td>
<td>aa</td>
</tr>
<tr>
<td>(2) $, S \rightarrow \epsilon$</td>
<td>bb$</td>
<td>aa</td>
</tr>
<tr>
<td>(3) $, b \rightarrow \epsilon$</td>
<td>b$</td>
<td>aab</td>
</tr>
<tr>
<td>(3) $, b \rightarrow \epsilon$</td>
<td>$</td>
<td>aabb</td>
</tr>
</tbody>
</table>
1. **Initial variable:** From the initial state $q_1$ push the initial variable onto the stack via $\epsilon$ and move to the loop state ($q_2$)

2. **Productions:** For each rule $(S \rightarrow aSb)$, perform a multi-push edge via $\epsilon$ from $q_2$ back to $q_2$, by popping the variable of the rule $S$ and performing a multi-push of the body $aSb$.

3. **Alphabet:** For each letter $a$ of the grammar draw a self loop to $q_2$ that reads $\epsilon$ and pops $a$ from the stack

4. **Final transition:** Once the stack is empty transition to the final state $q_3$ via $\epsilon$

$$S \rightarrow aSb$$

$$S \rightarrow \epsilon$$
aabb is in $L_3 = \{ a^n b^n \mid n \geq 0 \}$, show acceptance
aabb is in $L_3 = \{a^n b^n \mid n \geq 0\}$, show acceptance
1. The states $q_1, q_2, q_3, q_4$ are always in the converted PDA
2. States $q_1$ and $q_2$ push the sentinel and initial variable
3. The edge between $q_3$ and $q_4$ is always $\epsilon, \$ \rightarrow \epsilon$
4. There is always a self loop for each letter in the alphabet of $a, a \rightarrow \epsilon$
5. The only difficulty is *generating the substitution rules*
How to encode $S \rightarrow aSb$?

(multi push)
Encoding multi-push productions

By example $X \rightarrow aYb$

1. reverse the production, example:
   $X \rightarrow aYb$ yields $bYa$.

2. Create one state $R_i$ for each variable/terminal in the reversed string, each transition pushes a variable/terminal of the reversed string.

---

Note: In the book (and in my diagrams) I merge the first two transitions. This is equivalent to the above method; you can use either, as long as you do it correctly.
Exercise 3
Exercise 3

Convert the following grammar into a PDA

\[
A \rightarrow 0A1 \mid B \\
B \rightarrow 1B \mid \epsilon
\]
Exercise 3

Convert the following grammar into a PDA

$$A \rightarrow 0A1 \mid B$$
$$B \rightarrow 1B \mid \epsilon$$
Converting a PDA into a CFG
Converting a PDA into a CFG

1. modify the PDA into a **simplified** PDA:
   - has a single accepting state
   - empties the stack before accepting
   - every transition is in one of these forms:
     - skips popping and pushes one symbol onto the stack: $\epsilon \rightarrow c$
     - pops one symbol off the stack and skips pushing: $c \rightarrow \epsilon$
Converting a PDA into a CFG

1. modify the PDA into a simplified PDA:
   - has a single accepting state
   - empties the stack before accepting
   - every transition is in one of these forms:
     - skips popping and pushes one symbol onto the stack: $\epsilon \rightarrow c$
     - pops one symbol off the stack and skips pushing: $c \rightarrow \epsilon$

2. given a simplified PDA build a CFG
   - $A_{qq} \rightarrow \epsilon$ if $q \in Q$
   - $A_{pq} \rightarrow A_{pr} A_{rq}$ if $p, q \in Q$
   - $A_{pq} \rightarrow a A_{rs} b$ if $(r, u) \in \delta(p, a, \epsilon)$ and $(q, \epsilon) \in \delta(s, b, u)$
Simplifying a PDA
Simplifying a PDA

Transformation 1: Has a single accepting state

Transformation 2: Empties the stack before accepting

\[∀a ∈ \Sigma\] means that there will be one edge \(ε, a → ε\) per \(a ∈ \Sigma\)

† Notation \(∀a ∈ \Sigma\) means that there will be one edge \(ε, a → ε\) per \(a ∈ \Sigma\)
Simplifying a PDA

Transformation 3

Every transition is in one of these forms:

- skips popping and pushes one symbol onto the stack: $\epsilon \rightarrow c$
- pops one symbol off the stack and skips pushing: $c \rightarrow \epsilon$

Case 1

Case 2
Example 4

Simplified PDA

- single accepting state
- empties the stack before accepting
- every transition is in one of these forms:
  - \( \epsilon \rightarrow C \)
  - \( C \rightarrow \epsilon \)

Is it simplified?
Example 4

Simplified PDA

- single accepting state
- empties the stack before accepting
- every transition is in one of these forms:
  - $\epsilon \rightarrow c$
  - $c \rightarrow \epsilon$

Is it simplified?

No!

```
Example 4

Is it simplified?

No!

ε → c

Example 4

Is it simplified?

No!

ε → c
```
Example 4

Not Simplified

Simplified
Example 4

Not Simplified

Simplified
Simplified PDA to CFG
Simplified PDA to CFG

Given a simplified PDA build a CFG

1. \( A_{qq} \rightarrow \epsilon \) if \( q \in Q \)
2. \( A_{pq} \rightarrow A_{pr} A_{rq} \) if \( p, r, q \in Q \)
3. \( A_{pq} \rightarrow a A_{rs} b \) if \((r, u) \in \delta(p, a, \epsilon)\) and \((q, \epsilon) \in \delta(s, b, u)\)

\[
A \rightarrow q \\
\epsilon \in q \in Q \\
A \rightarrow p q \\
p, r, q \in Q \\
A \rightarrow p q a A \rightarrow s b (r, u) \in \delta(p, a, \epsilon) \\
p \in q \in \delta(s, b, u) \\
A \rightarrow s q b, u \rightarrow \epsilon
\]

for \( p=\{a, \epsilon \rightarrow u\} \rightarrow r \) in transitions:  # for every transition
for \( s=\{b, u \rightarrow \epsilon\} \rightarrow q \) in transitions:  # for every transition
yield \( A_{pq} \rightarrow a A_{rs} b \)
Example 5

Balanced parenthesis that are wrapped inside an outermost parenthesis.

Is this PDA simplified?
Example 5

Balanced parenthesis that are wrapped inside an outermost parenthesis.

Is this PDA simplified?

Yes!
Example 5

**Step 1:** \( A_{qq} \rightarrow \epsilon \) if \( q \in Q \)

**Step 2:** \( A_{pq} \rightarrow A_{pr} A_{rq} \) if \( p, r, q \in Q \)

\[
\begin{align*}
A_{11} & \rightarrow \epsilon \\
A_{22} & \rightarrow \epsilon \\
A_{33} & \rightarrow \epsilon \\
A_{44} & \rightarrow \epsilon \\
A_{1,2} & \rightarrow A_{1,3} A_{3,2} \\
A_{1,2} & \rightarrow A_{1,4} A_{4,2} \\
A_{1,3} & \rightarrow A_{1,2} A_{2,3} \\
A_{1,3} & \rightarrow A_{1,4} A_{4,3} \\
A_{2,1} & \rightarrow A_{2,3} A_{3,1} \\
A_{2,1} & \rightarrow A_{2,4} A_{4,1} \\
A_{2,3} & \rightarrow A_{2,1} A_{1,3} \\
A_{2,3} & \rightarrow A_{2,4} A_{4,3} \\
A_{2,4} & \rightarrow A_{2,1} A_{1,4} \\
A_{2,4} & \rightarrow A_{2,3} A_{3,4} \\
A_{3,1} & \rightarrow A_{3,2} A_{2,1} \\
A_{3,1} & \rightarrow A_{3,2} A_{2,1} \\
A_{3,2} & \rightarrow A_{3,1} A_{1,2} \\
A_{3,2} & \rightarrow A_{3,4} A_{4,2} \\
A_{3,4} & \rightarrow A_{3,1} A_{1,4} \\
A_{3,4} & \rightarrow A_{3,2} A_{2,4} \\
A_{4,1} & \rightarrow A_{4,2} A_{2,1} \\
A_{4,1} & \rightarrow A_{4,3} A_{3,1}
\end{align*}
\]
Example 5

Step 1: $A_{qq} \to \epsilon$ if $q \in Q$

Step 2: $A_{pq} \to A_{pr} A_{rq}$ if $p, r, q \in Q$

$A_{4,2} \to A_{4,1} A_{1,2}$

$A_{4,2} \to A_{4,3} A_{3,2}$

$A_{4,3} \to A_{4,1} A_{1,3}$

$A_{4,3} \to A_{4,2} A_{2,3}$
Example 5

Stack 0

<table>
<thead>
<tr>
<th>Push</th>
<th>Pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1, read o, q2</td>
<td></td>
</tr>
<tr>
<td>q2, read o, q2</td>
<td></td>
</tr>
<tr>
<td>q2, read c, q2</td>
<td></td>
</tr>
<tr>
<td>q2, read c, q3</td>
<td></td>
</tr>
</tbody>
</table>

New rules:

- $A_{1,2} \rightarrow oA_{22}c$
- $A_{1,3} \rightarrow oA_{22}c$
- $A_{2,2} \rightarrow oA_{22}c$
- $A_{2,3} \rightarrow oA_{22}c$

Intuition

- Create a table for each letter being pushed/popped.
- Pair each push with each pop.
Exercise 6

Simplify the PDA below
Exercise 6

Solution