Foundations of Computing

Lab 5 – PDAs and CFGs

February 14, 2024

Outline

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape
2. Optionally, pop a value from the Stack
3. Use the input symbol and the stack symbol to choose a next state
4. Optionally, push a value onto the Stack

A PDA M accepts a string w if the NFA in the control stops in an accept state once all the input has been processed

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape
2. Optionally, pop a value from the Stack
3. Use the input symbol and the stack symbol to choose a next state
4. Optionally, push a value onto the Stack

A PDA M accepts a string w if the NFA in the control stops in an accept state once all the input has been processed

Observations:

- Since the control is an NFA, ϵ transitions are allowed
- A PDA may choose not to touch the stack in a particular step
- Unlike the case for DFA/NFA, deterministic PDA's are not equal to non-deterministic ones. We will only study non-deterministic PDAs.

Example – Exercise from class last Wednesday

Show a PDA that recognizes the language

$$L = \{w \mid w \text{ has an equal number of 0s and 1s}\}$$

1. Describe a PDA algorithm for this language
2. Write the states and transition function
3. Draw the PDA graph

Example – Exercise from class last Wednesday

Show a PDA that recognizes the language

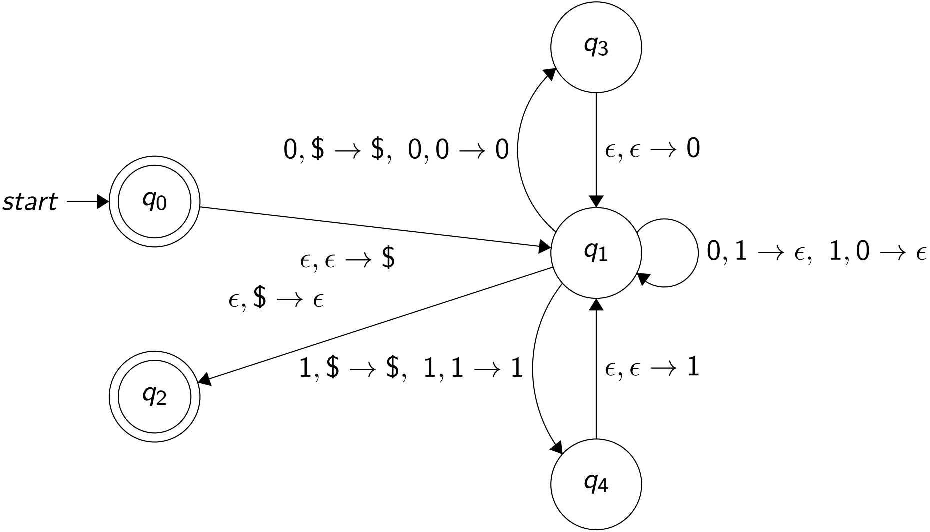
$$L = \{w \mid w \text{ has an equal number of 0s and 1s}\}$$

1. Describe a PDA algorithm for this language
2. Write the states and transition function
3. Draw the PDA graph

Algorithm:

1. Push \$ on the stack
2. If input is 0, pop value from the stack
   - If it's a 0 or \$ push it back on the stack and push another 0 on top
   - If it's a 1 pop it off the stack
3. If input is 1, pop value from the stack
   - If it's a 1 or \$ push it back and push another 1 on top
   - If it's a 0 pop it off the stack
4. When the input is done, if \$ is top of the stack, accept

Resulting PDA



Another Example – the Power of Non-determinism

Build a PDA that recognizes the language

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$$

Another Example – the Power of Non-determinism

Build a PDA that recognizes the language

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$$

Solution Idea:

- Already know how to check if number of b's matches number of a's

Another Example – the Power of Non-determinism

Build a PDA that recognizes the language

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$$

Solution Idea:

- Already know how to check if number of b's matches number of a's
- Can similarly check if number of c's matches number of a's

Another Example – the Power of Non-determinism

Build a PDA that recognizes the language

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$$

Solution Idea:

- Already know how to check if number of b's matches number of a's
- Can similarly check if number of c's matches number of a's
- But, how do we know which one to match?

Another Example – the Power of Non-determinism

Build a PDA that recognizes the language

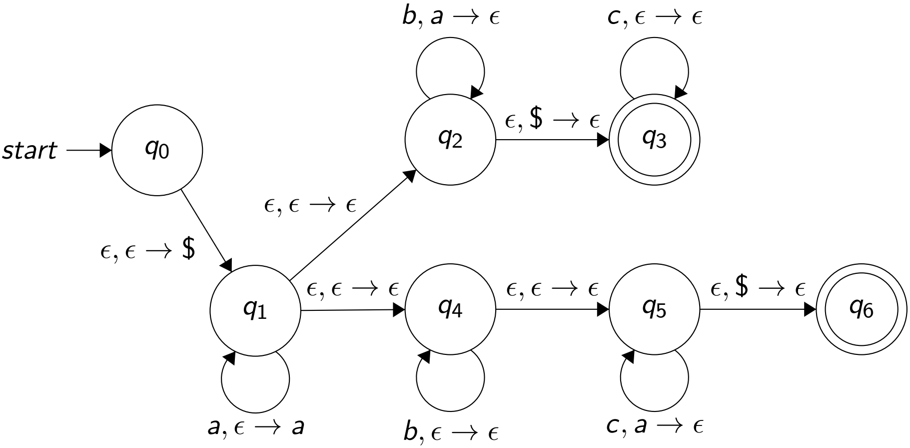
$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$$

Solution Idea:

- Already know how to check if number of b's matches number of a's
- Can similarly check if number of c's matches number of a's
- But, how do we know which one to match?
- Answer: Just guess which one to match non-deterministically

Build a PDA that recognizes the language

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$$



An Exercise – Work in Groups

1. Give a PDA M recognizing

$$L = \{ww^R \mid w \in \{0,1\}^*\}$$

Outline

Grammar

A grammar G consists of:

- V – finite set of variables (usually Capital Letters)
- Σ – a finite set of symbols called the terminals (usually lower case letters)
- R – finite set of rules how strings in L can be produced
- $S \in V$ – start variable

If no S is specified, can assume it is the variable in the first rule.

Definition

For a grammar G, the language L_G generated by G is the set of all terminal strings that can be produced by G starting with the start symbol by using a sequence of the production rules.

Strings Produced by a Grammar

For a grammar G generating language L, can generate each string $w \in L$ as follows:

1. Write down the start variable
2. Find a written-down variable and a rule starting with that variable. Replace the written variable with the right side of that rule
3. Repeat Step 2 until no variables remain

Definition

A grammar G is context-free if for all of its rules, the right side consists of exactly one variable and no terminals.

How to Design CFGs for L

Designing CFGs

- CFGs are inherently recursive (e.g., $A \to 0A1$) – need to think what happens when we recurse
- Build a string from outside in
- Build from both ends at the same time (due to recursion)

This is Tricky

Designing CFGs is not natural, takes lots of practice

Example 1

Question

Design a CFG for the language $L = \{a^m b^n c^k \mid m = n + k,\ m, n, k \geq 0\}$

Example 1

Question

Design a CFG for the language $L = \{a^m b^n c^k \mid m = n + k,\ m, n, k \geq 0\}$

Intuition:

- Each generated a is matched with either one b or one c
- Design a grammar for $a^i b^i$
- Design a grammar for $a^j c^j$

Example 1

Question

Design a CFG for the language $L = \{a^m b^n c^k \mid m = n + k,\ m, n, k \geq 0\}$

Intuition:

- Each generated a is matched with either one b or one c
- Design a grammar for $a^i b^i$
- Design a grammar for $a^j c^j$
- Consider the string aaaaabbccc
    - Red part on the inside
    - Blue part on the outside
- Generate outside part first, and then inside part

Example 1

Question

Design a CFG for the language $L = \{a^m b^n c^k \mid m = n + k,\ m, n, k \geq 0\}$

Intuition:

- Each generated a is matched with either one b or one c
- Design a grammar for $a^i b^i$
- Design a grammar for $a^j c^j$
- Consider the string aaaaabbccc
  - Red part on the inside
  - Blue part on the outside
- Generate outside part first, and then inside part
  1. S derives $a^j c^j$ and either terminate, or recurse and generate B
  2. B derives $a^i b^i$

Example 1

Question

Design a CFG for the language $L = \{a^m b^n c^k \mid m = n + k,\ m, n, k \geq 0\}$

Intuition:

- Each generated a is matched with either one b or one c
- Design a grammar for $a^i b^i$
- Design a grammar for $a^j c^j$
- Consider the string aaaaabbccc
  - Red part on the inside
  - Blue part on the outside
- Generate outside part first, and then inside part
  1. S derives $a^j c^j$ and either terminate, or recurse and generate B
  2. B derives $a^i b^i$

  Solution:

$$
\begin{aligned}
S &\rightarrow aSc \mid B \mid \epsilon \\
B &\rightarrow aBb \mid \epsilon
\end{aligned}
$$

Example 2

Question

Design a CFG for the language $L = \{w \in \{a, b\}^* \mid n_a(w) = n_b(w)\}$

Example 2

Question

Design a CFG for the language $L = \{w \in \{a, b\}^* \mid n_a(w) = n_b(w)\}$

Intuition:

- We want an equal number of a's and b's
- Every time we add an a, should also add a b
- Either a or b can be first
- Arbitrary strings with equal number of a's and b's everywhere else

Solution:

$$S \to SaSbS \mid SbSaS \mid \epsilon$$

Exercises

Construct CFGs for the following languages:

2. $\{a^n b^m \mid 2n \leq m \leq 3n\}$
3. $\{w \mid w \in \{a, b\}^* \text{ and } n_a(w) \neq n_b(w)\}$