Foundations of Computing

Lab 5 – PDAs and CFGs

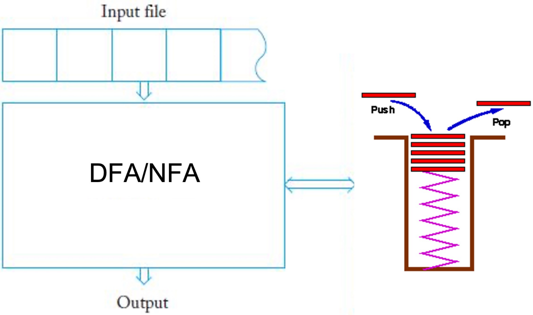
Arkady Yerukhimovich

February 15, 2023

Outline

1. Pushdown Automata (PDAs)
2. Context-Free Grammars (CFGs)
3. Solutions

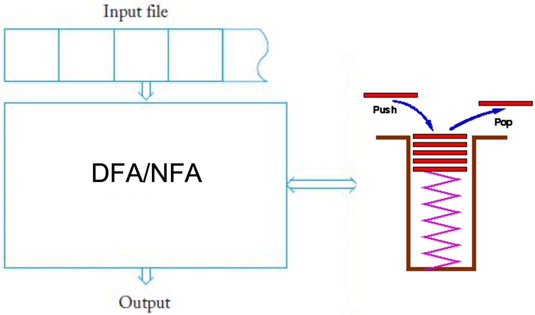
Pushdown Automata (PDA)



A PDA consists of:

- An NFA for a control unit
- A Stack for storage

Pushdown Automata (PDA)



A PDA consists of:

- An NFA for a control unit
- A Stack for storage

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.

An Example PDA

Consider the following PDA "Algorithm"

1. Read a symbol from the input
2. If it is a 0 and I have not seen any 1s, then push a 0 onto the stack
3. If it is a 1, pop a value (a 0) from the stack
4. Accept if and only if the stack becomes empty when we read the last character
5. Reject if either
   - the stack becomes empty and the input is not done or
   - there are still 0s left on the stack when the last input is read or
   - there are any 0s after the first 1

An Example PDA

Consider the following PDA "Algorithm"

1. Read a symbol from the input
2. If it is a 0 and I have not seen any 1s, then push a 0 onto the stack
3. If it is a 1, pop a value (a 0) from the stack
4. Accept if and only if the stack becomes empty when we read the last character
5. Reject if either
    - the stack becomes empty and the input is not done or
    - there are still 0s left on the stack when the last input is read or
    - there are any 0s after the first 1

This recognizes

$$L = \{0^n 1^n \mid n \geq 0\}$$

Back to Our Example

Recall the PDA we described before:

- On input 0, push a 0 on the stack
- On input 1, pop a value from the stack
- If all 0s come before all 1s and the stack is empty when run out of inputs, accept

Back to Our Example

Recall the PDA we described before:

- On input 0, push a 0 on the stack
- On input 1, pop a value from the stack
- If all 0s come before all 1s and the stack is empty when run out of inputs, accept

Let's build a PDA for this algorithm:

- $Q = \{q_0, q_1, q_2, q_3\}$
  - q_0 – start state
  - q_1 – seen only 0s
  - q_2 – seen 0s followed by 1s
  - q_3 – accept state
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, \$\}$ – \$ is a special symbol to indicate the stack is empty
- $q_0 = q_0$
- $F = \{q_3\}$

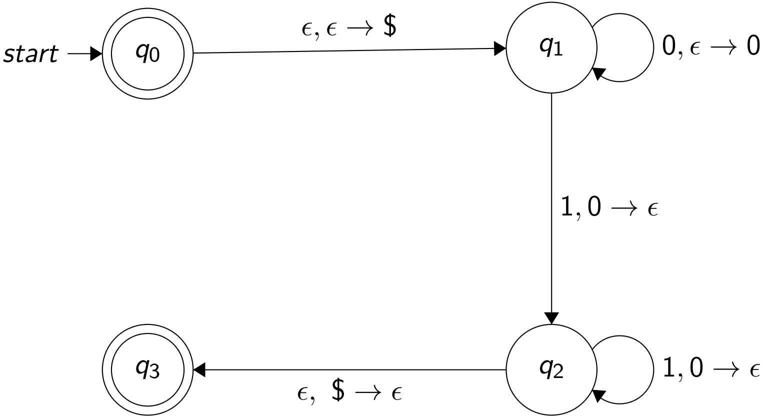
Transition Function

Input:	0			1			ϵ		
Stack:	0	\$	ϵ	0	\$	ϵ	0	\$	ϵ
q_0									$\{(q_1, \$)\}$
q_1			$\{(q_1, 0)\}$	$\{(q_2, \epsilon)\}$					
q_2				$\{(q_2, \epsilon)\}$				$\{(q_3, \epsilon)\}$	
q_3									

Table: Transition Function δ

Empty cells correspond to output of $\emptyset$

Example PDA as a Graph



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

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

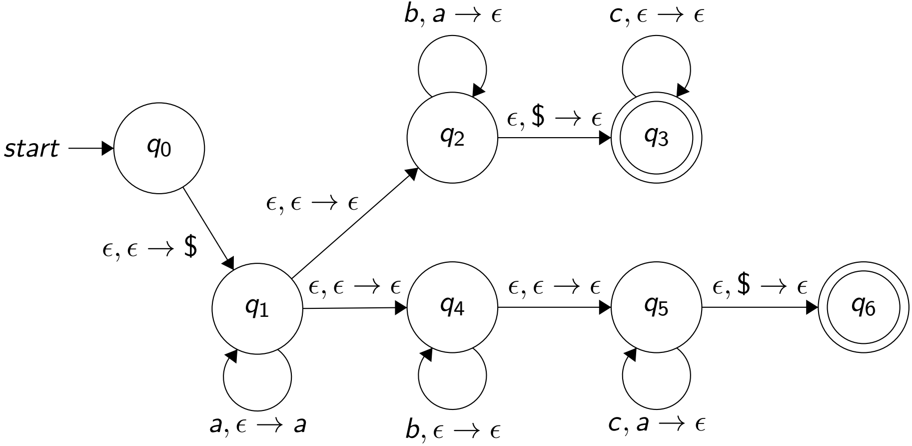
Solution Idea:

- 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

Another Example – the Power of Non-determinism

Build a PDA that recognizes the language

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



An Exercise – Work in Groups

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 $L = \{w \in \{a, b\}^* \mid n_a(w) = n_b(w)\}$

Example 1

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
- Generate outside-in and think recursively
- Each layer is either $a \ldots b$ or $b \ldots a$
- How do we generate this?

Example 1

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
- Generate outside-in and think recursively
- Each layer is either $a \dots b$ or $b \dots a$
- How do we generate this?

Solution:

$$S \to aSb \mid bSa \mid \epsilon$$

Example 2

Question

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

Example 2

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 2

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 2

Question

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

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 2

Question

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

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:

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

Exercises

Construct CFGs for the following languages:

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