Pushdown Automata

Rationale

- CFG are specification mechanisms, i.e., a CFG generates a context-free language.
- Pushdown-automata are recognizing mechanisms, i.e., a PDA recognizes a context-free languiage.
- CFG are like regular expressions and PDA are like FA.

A new type of computation model

- Pushdown automata, PDA, are a new type of computation model
- PDAs are like NFAs but have an extra component called a *stack*
- The stack provides additional memory beyond the finite amount available in the control
- The stack allows PDA to recognize some nonregular languages

PDA and CFG

- PDA are equivalent in specification power with CFG
- This is useful because it gives us two options for proving that a language is context-free:
  1. construct a CFG that generates the language or
  2. construct a PDA that recognizes the language

Note

Some CFL are more easily described in terms of their generators, whereas others are more easily described in terms of their recognizers

Schematic representation of a PDA

Figure 1 contrasts the schematic representations of PDA and NFA :

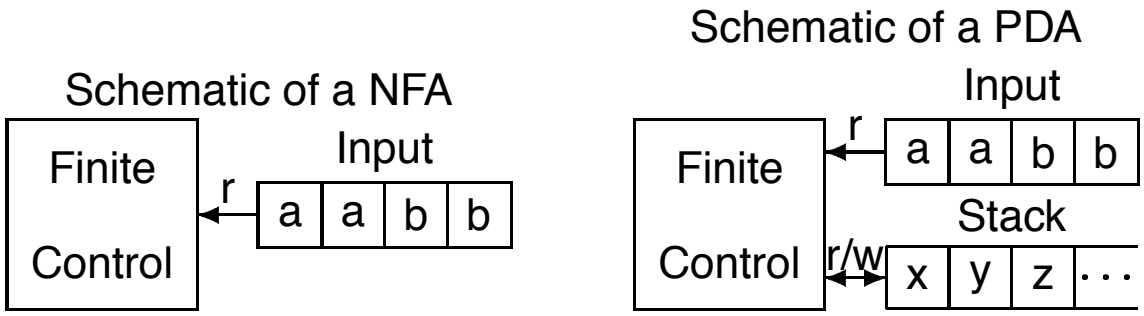

Figure 1: Schematic representations

Note: in Figure 1 `r` stands for read and `w` stands for write.

More on PDA informal

- A PDA can write symbols on the stack and read them back later
- Writing a symbol “pushes down" all the other symbols on the stack
- The symbol on the top of the stack can be read and removed.
- When the top symbol is removed the remaining symbols move up

Terminology

- Writing a symbol on the stack is referred to as *pushing* the symbol
- Removing a symbol from the stack is referred to as *popping* the symbol
- All access to the stack may be done only at the top

In other words: a stack is a “last-in fi rst-out" storage device

Benefits of the stack

- A stack can hold an unlimited amount of information
- A PDA can recognize a language like $\{0^n 1^n | n \geq 0\}$ because it can use the stack to remember the number of 0s it has seen

PDA recognizing $\{0^n 1^n | n \geq 0\}$

Informal algorithm:

1. Read symbols from the input. As each 0 is read push it onto the stack
2. As soon as a 1 is read, pop a 0 off the stack for each 1 read
3. If input fi nishes when stack becomes empty accept; if stack becomes empty while there is still input or input fi nishes while stack is not empty reject

Nature of PDA

- PDA may be nondeterministic and this is a crucial feature because nondeterminism adds power
- Some languages, such as $\{0^n 1^n | n \geq 0\}$ do not require nondeterminism, but other do. Such a language is $\{ww^{\mathcal{R}} | w \in \{0,1\}^*\}$

Toward PDA formalization

- Formal defi nition of a PDA is similar to a NFA, except the stack
- PDA my use different alphabets for input and stack, we will denote them by Σ and Γ
- Nondeterminism allows PDA to make transitions on empty input. Hence we will use $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$
- Domain of the PDA transition function is $Q \times \Sigma_\epsilon \times \Gamma_\epsilon$ where Q is the set of states
- Since a PDA can write on the stack while performing nondeterministic transitions the range of the PDA transition function is $\mathcal{P}(Q \times \Gamma_\epsilon)$

In conclusion: $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$

Definition 2.8

A *pushdown automaton* is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, Γ, and F are finite sets, and:

1. Q is a set of states
2. Σ is the input alphabet
3. Γ is the stack alphabet
4. $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ is the transition function
5. $q_0 \in Q$ is the start state
6. $F \subseteq Q$ is the set of accept states

PDA computation

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows:

- M inputs $w = w_1 w_2 \dots w_m$, where each $w_i \in \Sigma_\epsilon$
- There are a sequence of states $r_0, r_1, \dots, r_m \in Q$ and a sequence of strings $s_0, s_1, \dots, s_n \in \Gamma^*$ that satisfy the conditions:
  1. $r_0 = q_0$,$s_0 = \epsilon$, i.e., M starts in the start state with empty stack
  2. For $i = 0, 1, \dots, m-1$ we have $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\epsilon$ and $t \in \Gamma^*$, i.e., M moves properly according to the state, stack, and input symbol
  3. $r_m \in F$, i.e., an accept state occurs at the input end

Example PDA

The PDA that recognizes the language $\{0^n 1^n | n \geq 0\}$ is $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, F)$ where:

$Q = \{q_1, q_2, q_3, q_4\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, \$\}$, $F = \{q_1, q_4\}$

δ is defi ned by the table:

Σ_ϵ	0			1			ϵ		
Q/Γ_ϵ	0	\$	ϵ	0	\$	ϵ	0	\$	ϵ
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \epsilon)\}$					
q_3				$\{(q_3, \epsilon)\}$				$\{(q_4, \epsilon)\}$	
q_4									

Notation: $(q \in Q, w \in \Sigma_\epsilon, a \in \Gamma_\epsilon) = \{(r \in Q, b \in \Gamma_\epsilon)\}$ means $q \xrightarrow{w, a \to b} (r, b)$.

Transition diagrams of PDA

- We can use state transition diagrams to describe PDA
- Such diagrams are similar to state transition diagrams used to describe finite automata
- To show how PDA uses the stack we write "$a, b \to c$" to signify that the machine:
  1. is reading a from input
  2. may replace the symbol b on top of the stack
  3. with the symbol c

  where any of a, b, c may be ϵ

Interpretation

- If a is ϵ, the machine makes this transaction without reading any symbol from the input
- If b is ϵ, the machine makes this transaction without popping any symbol from the stack
- If c is ϵ, the machine makes this transaction without pushing any symbol on the stack

Transition diagram of PDA M_1

Figure 2 show the state transition diagram of M_1 recognizing the language $\{0^n1^n|n \geq 0\}$

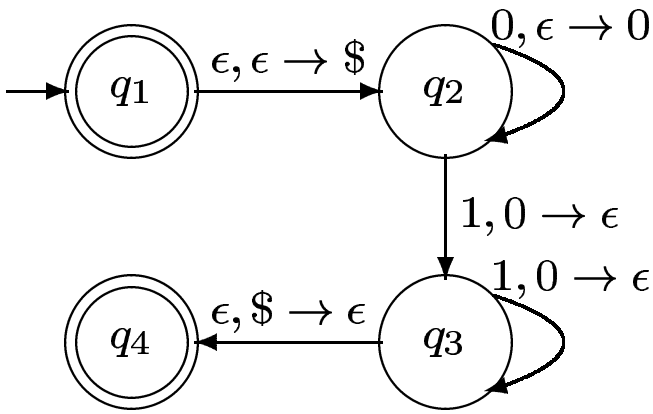

Figure 2: Transition diagram for PDA M_1

Note 1

- The formal definition of a PDA contains no explicit mechanism for testing the empty stack
- PDA in Figure 2 test empty stack by initially placing a special symbol, $, on the stack
- If ever it sees $ again on the stack, it knows that the stack is effectively empty

This is the procedure we use to test empty stack

Note 2

- The PDA has no mechanism to test explicitly the reaching of end of the input
- The PDA in Figure 2 is able to achieve this effect because the accept state takes effect only when the machine is at the end of the output

Thus, from now one we use the same mechanism to test for the end of the input

Example 2.10

This example provides the PDA M_2 that recognizes the language $\{a^i b^j c^k | i, j, k \geq 0 \wedge i = j \vee i = k\}$

Informal description:

- The PDA fi rst reads and push a's on the stack
- When this is done, it can match as with bs or cs
- Since machine does not know in advance whether as are matched with bs or cs, nondeterminism helps
- Using nondeterminism the machine can guess: the machine has two branches, one for each possible guess
- If either of these branches match, that branch accepts and the entire machine accepts

M_2 recognizing $\{a^i b^j c^k | i, j, k \geq 0 \wedge i = j \vee i = k\}$

The transition diagram is in Figure 3

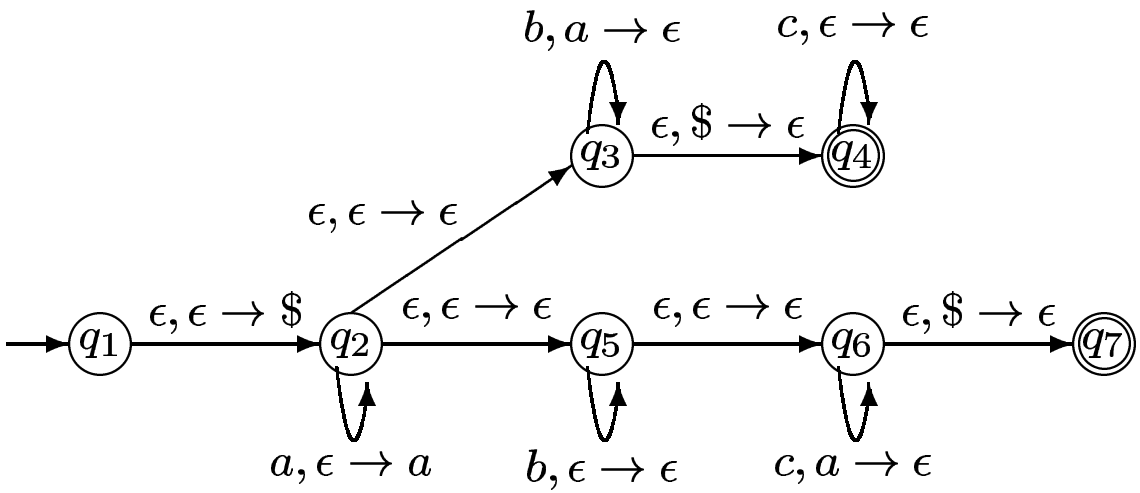

Figure 3: Transition diagram of PDA M_2

Example 2.11

This example provides the PDA M_3 that recognizes the language $\{ww^{\mathcal{R}} | w \in \{0,1\}^*\}$.

Recall: $w^{\mathcal{R}}$ means w written backwards.

Informal description

1. M_3 begins by pushing the symbols that are read onto the stack
2. At each point M_3 nondeterministically guesses that the middle of the input has been reached
3. When middle of the word has reached the machine starts popping off the stack for each symbol read, checking to see that what is read and what is popped off is the same symbol
4. If the symbol read from the input and popped out from the stack is the same and the stack empties as the same time as input is fi nished, accept; otherwise reject

The transition diagram of M_3 is in Figure 4

Transition diagram of PDA M_3

Figure 4 shows the state transition diagram of M_4 recognizing the language $\{ww^{\mathcal{R}}|w \in \{0,1\}\}$

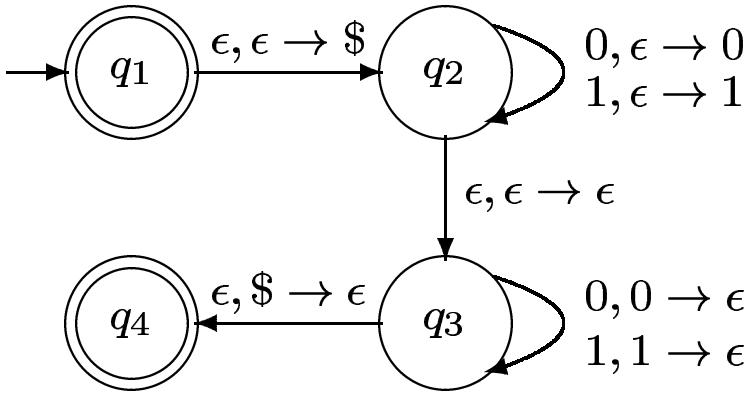

Figure 4: Transition diagram for PDA M_3