CSC 320 - Lecture 10b

#context-free #PDA #languages #non-context-free #pumping-lemma

---

Pushdown Automata

Think. Nondeterministic finite automaton with addition of **stack**. (We note that deterministic PDA are different). The PDA is automatically more powerful.

A stack provides additional memory.

Show. We will show that languages recognized by pushdown automata are exactly the context-free languages.

Definition

A **pushdown automaton (PDA)** is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ with...

- Q: finite set of states
- Σ: finite **input alphabet**
- Γ: finite **stack alphabet**
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ transition function
- $q_0 \in Q$: start state
- $F \subseteq Q$: set of accept states

Note. $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ and $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$

Note. You can also have a 7-tuple where $Z_0 \in \Gamma$ is the start symbol for the stack.

Computation of PDA

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA. Then M **accepts** input $w \in \Sigma^*$ if w can be written as $w = w_1 w_2 \dots w_m, |w| \leq m$, where: $w_i \in \Sigma_\varepsilon$ and there exist states $r_0, r_1, \dots, r_m \in Q$ and strings $s_0, s_1, \dots, s_m \in \Gamma^*$ such that...

- $r_0 = q_0$ and $s_0 = \varepsilon$
- for $i = 0, \dots, m-1 : (r_{i+1}, b) \in \delta(r_i, w_{i_1}, a)$ with $s_i = at, s_{i+1} = bt$, $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$
- $r_m \in F$

Note. $s_i's$: sequence of stack contents that M has on accepting branch (of computation). M starts computation with empty stack.

Note

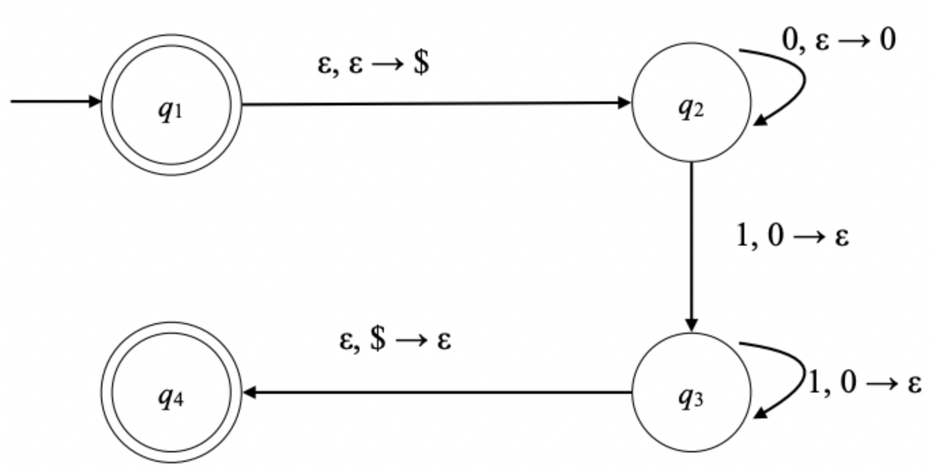
$(r_{i+1}, b) \in \delta(r_i, w_{i_1}, a)$ means: when M is in state r_i reading w_{i+1} from input and top stack symbol is a, then M can do the following: move into state r_{i+1} and replace top stack symbol by b.

If $a = \varepsilon$ then top stack symbol is ignored and symbol b is push onto stack.

If $b = \varepsilon$ then top stack symbol a is removed from stack.

Example: State Diagram Representation of PDA

$\Sigma = \{0, 1\}$, $\Gamma = \{0, \$\}$: Input $w = 0011$



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

Note. Empty string is accepted.

Note. It is nicer if you empty the stack at the end of reading the accepted string. But it is not obligatory. We only require that the stack is empty at the start.

Mini Example

Input $w = 0101$ (NOT ACCEPTED)

- Stack: $\$$ (Read 0)
- Stack: $\$0$ (Read 1)
- Stack $\$$ (Remove $\$$)

Sequence: $q_1 q_2 q_2 q_3 q_4$.

Computation

- In q_1 reading input symbol ε and ignoring stack content, move to q_2 and push symbol $\$$ onto stack.
- In q_2 reading first input symbol 0 and ignoring stack content, remain in q_2 and push symbol 0 onto stack.
- In q_2 reading second input symbol 0 and ignoring stack content, remain in q_2 and push symbol 0 onto stack.
- In q_2 reading third input symbol 1 and while 0 is top stack symbol, move to q_3 and pop symbol 0 from stack.
- In q_3 reading fourth input symbol 1 and while 0 is top stack symbol, remain in q_3 and pop symbol 0 from stack.
- In q_3 reading input symbol ε and while $\$$ is top stack symbol, move to q_4 and pop symbol $\$$ from stack.

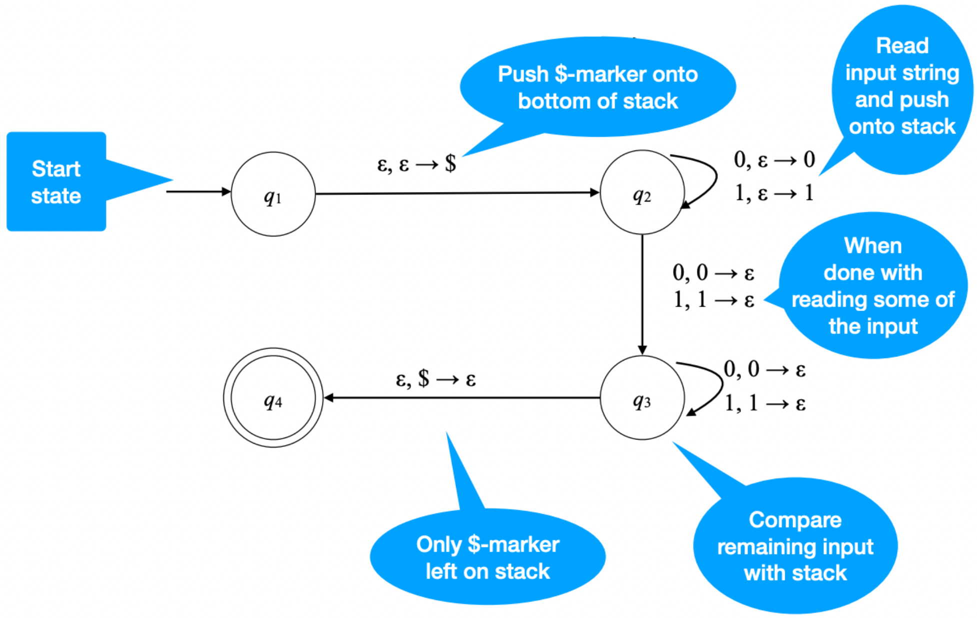
Stack
0
0
$\$$

Note. When $s = 00011$ is $s \in L(M)$? NO!

What is $L(M)$? $L = \{0^n 1^n | n \geq 0\}$

Example: Designing DFA

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



Note. Only strings accepted by the machine are of form ww^R. However, not every possible computation branch will yield acceptance, and every string of form ww^R has accepting branch in computation tree.

What is the accepting state sequence of computation for input $w = 10100101$?

- $q_1q_2q_2q_3q_4$
- $q_1q_2q_2q_2q_3q_3q_3q_3q_4$
- $q_1q_2q_2q_2q_2q_2q_3q_3q_3q_3q_3q_4$
- None of the above.

Sequence: $q_1q_2q_2q_2q_2q_2q_2\ldots$

Context-Free Languages

Theorem. A language is context free if and only if some PDA recognizes it.

Proof Idea.

- **If**. Since every context free language L can be produced by context free grammar $G, L = L(G)$, convert G into PDA M with $L(M) = L(G) = L$.
- **Only If**. Given pushdown automaton M, create context free grammar G with $L(G) = L(M)$.

If language is context free then some PDA recognizes it.

Given $G = (V, \Sigma, R, S)$ context-free, $L = L(G)$, design PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$.

- Places marker symbol ($) and start variable S onto (empty) stack.
- For each top stack symbol...
  - If variable, say A, then choose from G some rule $A \longrightarrow u, u = \alpha_1\alpha_2 \ldots \alpha_k$ and substitute A with $\alpha_1\alpha_2 \ldots \alpha_k$ (with α_1 new top symbol).
  - If terminal, say a, read next input symbol w_i reject if $w_i \neq a$, pop if $w_i = 1$.
  - If $ go to accept state.

Detailed Description of M

For $G = (V, \Sigma, R, S)$ context-free design $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ with...

- $Q = \{q_{start}, q_{loop}, q_{accept}\} \cup$ set of auxiliary states to push right hands of rules in R onto stack.
- $\Gamma = V \cup \Sigma \cup \{\$\}$
- $q_0 = q_{start}$
- $F = \{q_{accept}\}$

M's Transitions

- In q_{start} when reading ε and top symbol ε: push first $ and then S onto stack and move into q_{loop}.
- For each rule $A \longrightarrow \alpha_1\alpha_2 \ldots \alpha_k$, in R: in q_{loop} for top stack symbol A: replace A by $\alpha_1\alpha_2 \ldots \alpha_k$ and remain in q_{loop}.
- For each terminal $a \in \Sigma$: if a is top stack symbol then pop a and remain in q_{loop}.
- If $ is top stack symbol then pop $ and move into q_{accept}

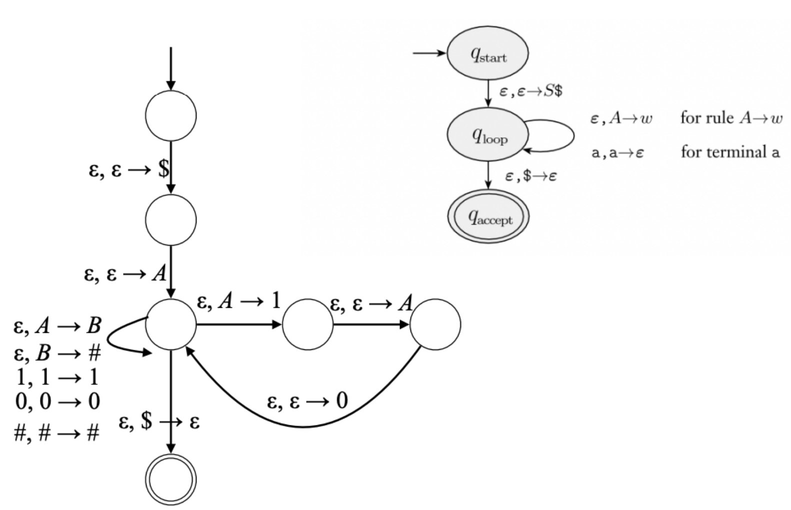
Example

Note. Midterm Practice: Test grammar and input on strings 011001 and $000\#111$.

$A \rightarrow 0A1$

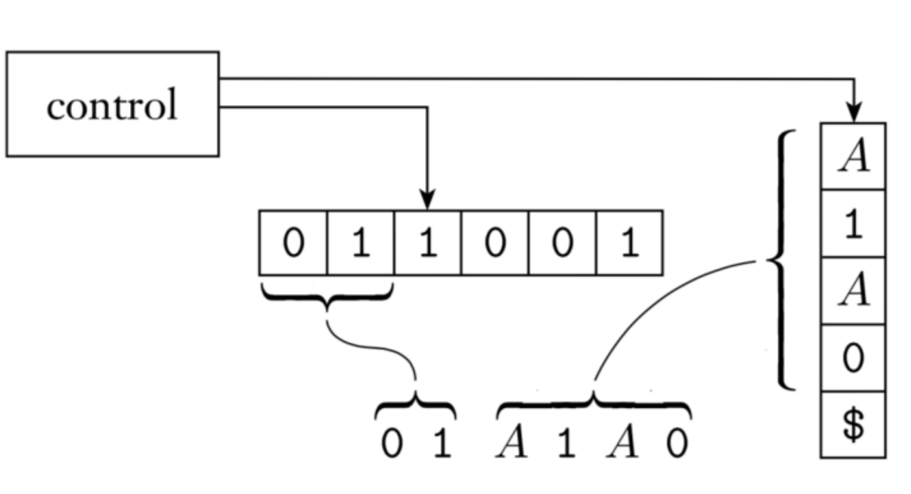
$A \rightarrow B$

$B \rightarrow \#$



PDA representing derived intermediate string $01A1A0$.

Transition $a, a \longrightarrow \varepsilon$ are used to cleanup the stack.



Note. PDA simulates G's leftmost derivation.

If a PDA recognizes some language then it is context free

- Step 1: Simplify PDA
    - Single accept state (Add new state, ε-transition (don't read, pop $\$$) from each (original) accept state to new state. Make new state only accept state).
    - $\$$ always popped exactly before moving into accept state. (Only transition from start state: push $\$$ onto empty stack).
    - Transition either push or pop, not both at the same time. (Every transition that replaces top stack symbol replace by two transitions: first one pop's the symbol and a second following directly after pushing the (original) replacement into stack).
- Step 2: Design Grammar

Previous Lecture

Lecture10a

Next Lecture

Lecture11