EECS 2001N : Introduction to the Theory of Computation

Suprakash Datta
Office: LAS 3043

Course page: `http://www.eecs.yorku.ca/course/2001N`
Also on Moodle

Pushdown automata (PDA)

Add a stack to a Finite Automaton

- Can serve as type of memory or counter
- More powerful than Finite Automata
- Accepts Context-Free Languages (CFLs)
- Unlike FAs, nondeterminism makes a difference for PDAs. We will only study non-deterministic PDAs and omit DPDAs.
- Pushdown automata are for context-free languages what finite automata are for regular languages.

Pushdown automata - 2

- PDAs are recognizing automata that have a single stack (=memory): Last-In-First-Out pushing and popping
- Non-deterministic PDA's can make non-deterministic choices (like NFAs) to find accepting paths of computation
- Informally: The PDA M reads w and stack element. Depending on: input $w_i \in \Sigma_\epsilon$, stack element $s_j \in \Gamma_\epsilon$ and - state $q_k \in Q$, the PDA M - jumps to a new state and pushes an element from Γ_ϵ into the stack (nondeterministically)
  If possible to end in an accepting state $q \in F \subseteq Q$, then M accepts w

Pushdown automata - Formal Description

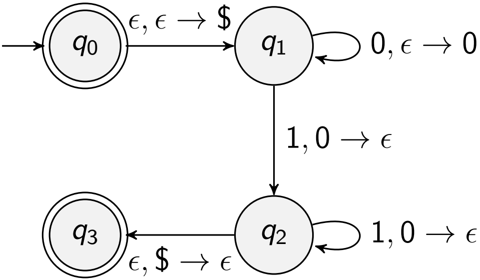
A Pushdown Automata M is defined by a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$:

- finite set of states Q
- finite input alphabet Σ
  [The book uses a tape for input, so calls Σ the **tape** alphabet]
- finite stack alphabet Γ
- start state $q_0 \in Q$
- set of accepting states $F \subseteq Q$
- transition function $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$

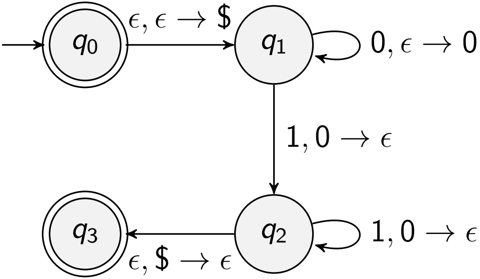
Pushdown automata - Example 1

PDA for language $L = \{0^n1^n | n \geq 0\}$

- The PDA first pushes "$\$0^n$" on stack
- Then, while reading the 1^n string, the zeros are popped
- If, in the end, \$ is left on stack, then "accept"



Pushdown automata - Tracing



On $w = 000111$ (state; stack) evolution:

- $(q_0; \epsilon) \to (q_1; \$) \to (q_1; 0\$) \to (q_1; 00\$) \to (q_1; 000\$) \to (q_2; 00\$) \to (q_2; 0\$) \to (q_2; \$) \to (q_2; \epsilon)$. q_3: accept state

On $w = 0101$:

- $(q_0; \epsilon) \to (q_1; \$) \to (q_1; 0\$) \to (q_1; \$) \to (q_2; \epsilon) \to (q_3; \epsilon)$
- But we still have part of input "01". There is no accepting path

Pushdown automata - Example 2 (Sec 3.6.3)

Suppose $\Sigma = \{a, b\}$. Design a PDA for $L = \{vbw \mid |v| = |w|\}$

- 2 states q_0 (start state) and q_1
- state q_0: automaton has not reached the middle symbol b
- state q_1: automaton has read the middle symbol b
- in state q_0 it either
  - pushes one symbol onto the stack and stays in state q_0, or
  - if the current input symbol is b, it nondeterministically "guesses" that it has reached the middle of the input string, and switches to state q_1
- in state q_1, it pops the top symbol from the stack and stays in state q_1
- The input string is accepted if and only if, at the end of input, the automaton is in state q_1 and the top symbol on the stack is \$

Pushdown automata - Exercises

- $L = \{ww^R | w \text{ is any binary string }\}$

- $L = \{a^i b^j a^k | i = j \text{ or } i = k\}$

Equivalence of PDA, CFL

- Theorem 3.7.1: A language L is context-free if and only if there is a pushdown automata M that recognizes L.
- Two step proof:
  1) Given a CFG G, construct a PDA M_G 2) Given a PDA M, make a CFG G_M
- Our book only does the first part
  - The PDA should simulate the derivation of a word in the CFG and accept if there is a derivation.
  - Need to store intermediate strings of terminals and variables. How?