6.5 Pushdown Automata

- deterministic pushdown automaton is very much like a deterministic finite automaton
  - but with addition of storage capability
  - storage is in the form of a stack
    - only element of storage that we can exam is the one last put on the stack (on top)
  - transition now based on:
    - current state
    - input symbol
    - symbol on top of stack
  - usually don’t represent PDA as graphs
    - but instead write out values of transition function
    - l is always initial state...
    - F is accepting state
    - empty stack represented by

- formally, a PDA is a 7-tuple 
  \((Q, \Sigma, \Gamma, q_0, T, Z, \delta)\)

  - \(Q\) is finite set of states
  - \(\Sigma\) is alphabet of input and stack symbols
  - \(\Gamma\) is stack alphabet
  - \(q_0\) is starting state (one)
  - \(T\) is subset of \(Q\) giving accepting states
  - \(Z\) is subset of \(\Gamma\) and identifies initial stack symbol
  - \(\delta\) is transition function
    - of form: \(\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*\)
    - \(\delta(s, a, b) = (s', x)\)
      - \(s = \) current state
      - \(a = \) current input symbol
      - can also be \(\lambda\) (indicating input not consumed)
      - \(b = \) is symbol at top of stack
        - always removed from stack
        - if it needs to be kept on stack, then it must be part of \(x\)
        - \(\varepsilon = \) empty stack
      - \(s' = \) new state
      - \(x = \) zero or more symbols that replace \(b\) at top of stack
    - in general, transitions take on four forms:
      - \(\delta(s, *, b) = (s', b)\) \(//\) does not change stack
      - \(\delta(s, *, b) = (s', xb)\) \(//\) adds \(x\) to the stack
      - \(\delta(s, *, b) = (s', \lambda)\) \(//\) removes \(b\) from stack
      - \(\delta(s, *, b) = (s', x)\) \(//\) replaces \(b\) with \(x\)

- Examples

  - **M1** : (accepts strings of form \(a^n b^n ; n \geq 1\))
    - \(Q = \{1, 2, 3\}\)
    - \(\Sigma = \{a, b, \lambda\}\)
    - \(\Gamma = \{\lambda, A, \varepsilon\}\)
    - \(q_0 = 1\)
    - \(T = 3\)
    - \(Z = \varepsilon\)
    - \(\delta:\)
      - \(\delta(1, a, \varepsilon) = (1, A\varepsilon)\)
      - \(\delta(1, a, A) = (1, AA)\)
      - \(\delta(1, b, A) = (2, \lambda)\)
\[ \delta(2, b, A) = (2, \lambda) \]
\[ \delta(2, \lambda, \varepsilon) = (3, \varepsilon) \]

**L2 = \{ w : \#a(w) = \#b(w) \}**
- equal number of a and b in string
- (order doesn’t matter)

- \( Q = \{ 1, 2 \} \)
- \( \Sigma = \{ a, b, \lambda \} \)
- \( \Gamma = \{ \lambda, A, B, \varepsilon \} \)
- \( q_0 = 1 \)
- \( T = 2 \)
- \( Z = \varepsilon \)

- **\( \delta \):**
  - \( \delta(1, a, \varepsilon) = (1, A\varepsilon) \)
  - \( \delta(1, b, \varepsilon) = (1, B\varepsilon) \)
  - \( \delta(1, a, A) = (1, AA) \)
  - \( \delta(1, b, B) = (1, BB) \)
  - \( \delta(1, a, B) = (1, \lambda) \)
  - \( \delta(1, b, A) = (1, \lambda) \)
  - \( \delta(1, \lambda, \varepsilon) = (2, \varepsilon) \)

  - every time we see a,
    - if A on top of stack (or stack empty), push another A
    - if B on top of stack, pop B off
  - every time we see b,
    - if B on top of stack (or stack empty), push another B
    - if A on top of stack, pop A off
  - if out of input symbols and stack empty, move to accepting state.
a, $\epsilon$ -> A
a, A -> AA
a, B -> $\lambda$
b, $\epsilon$ -> B
b, B -> BB
b, A -> $\lambda$

$\lambda, \epsilon$ -> $\epsilon$