Push Down Automata (PDA)

- Non-deterministic PDA
  - Languages accepted by NPDA
- Deterministic PDA
  - Languages accepted by DPDA

Push Down Automata (PDA)

- A class of automata associated with CFLs
- An FSA, being at a certain state advances to the next state based on the input it reads
- A PDA advances to the next state based on the input it reads and the top most symbol in the stack.
- Unlike FSA, PDA has a **memory** in the form of a **stack**
- Two types of PDA: NPDA & DPDA

Non-deterministic PDA (NPDA)

- <u>Definition</u>: NPDA is a 7-tuple M=(Q, Σ, V, P, q_0, Z_0, F) where
  1. Q : finite set of states
  2. Σ : input alphabet
  3. V : symbols on the stack (stack alphabet)
  4. P : is a total function called pushdown function and is given by P:QXVX(ΣU{λ}) → QXV*
  5. q_0 Є Q is the start state
  6. Z_0 : stack initializer symbol
  7. F $\underline{C}$ Q : set of final states

NPDA: cont'd

- Note that the arguments of P are
  - The current state
  - The current input symbol
  - The current symbol on top of the stack
- The result is a set of pairs (q, x) where q is the next state and x is a string which is put on top of the stack in place of the single symbol there before
- λ-transition is possible, i.e. the second argument may be empty (λ)
- No move is possible if the stack is empty.

NPDA: cont'd

NPDA operations (execution)

- Read an input
- Pop the top element from the stack
- Push element(s) to the stack
- Enter next state

- The operations can be represented as follows: (q', s, x; q'', y) where
  - q' : current state
  - s : element popped from the stack
  - x : incoming input
  - q'' : next state
  - y : symbol pushed on to the stack

NPDA: cont'd

- the operations can also be represented using a transition diagram

  (q', s, x; q'', y) is represented in such a way that the arc from state q' to state q'' is labeled with s, x; y

- Example: suppose the set of transition rules of an NPDA contain p(q1, a, b) = {(q2, cd), (q3, λ)}

  Hence, if at any time the automata is in state q1, the input symbol read is **a**, and the symbol on top of the stack is **b**, then either:

  1. the automata goes into state q2 and the string **cd** replaces **b** on top of the stack, or
  2. it goes into state q3 with the symbol **b** removed from the top of the stack

NPDA: cont'd

- Instantaneous Description (ID)

  An ID of a PDA M is (q, x, α) where $q \in Q$, $x \in \Sigma^*$ and $\alpha \in V^*$

- An initial ID is (q_0, x, Z_0), i.e. the PDA is at state q_0, the input string to be processed is x and the stack contains Z_0
- a move relation (denoted by $\vdash$) between IDs is defined as:

  $(q, a_1a_2 \ldots a_n, Z_1Z_2\ldots Z_m) \vdash (q', a_2\ldots a_n, \beta Z_2\ldots Z_m)$

  if $P(q, a_1, Z_1)$ contains (q', β)

Acceptance of strings by NPDA

- The set of strings accepted by NPDA M is denoted by L(M) and defined as follows:

1. $L(M) = \{x \mid (q0, x, Z0) \vdash^* (q, \lambda, v)$ for some q in F and v in $V^*\}$ (acceptance by Final Sate)
2. $L(M) = \{x \mid (q0, x, Z0) \vdash^* (q, \lambda, \lambda)$ for some q in $Q\}$ (acceptance by Empty Stack)

- Note that in (1), the stack content (v) is irrelevant. i.e. all strings that can put M into a final state at the end of the string are accepted.

Acceptance by NPDA: cont'd

- Example 1:

  Construct NPDA that accepts the language

  $L = \{xcx^r \mid x Є \{a, b\}\}$, x^r is the reverse of x.

  an example of a string in L can be:

  w = abbcbba

Acceptance by NPDA: cont'd

- <u>Solution</u>:

  The NPDA operates as follows:

1. as it reads symbols to the left of the symbol c, it pushes the symbol read and remains in the same state
2. when it 'sees' c, it enters a new state without doing anything on the stack
3. it compares the incoming symbol with the top element on the stack. If there is a match, it pops off the top element. Otherwise, the operation stops.

Acceptance by NPDA: cont'd

- The pushdown function (P) will have:

1. (q0, a, v) = (q0, av), a Є{a, b}, v ЄV*
2. (q0, c, v) = (q1, v)
3. (q1, b, bv) = (q1, v), b Є{a, b}, v ЄV*
4. (q1, λ, Z0) = (q2, λ)

Thus, M=(Q, Σ, V, P, q0, Z0, F) where
Q = {q0, q1, q2}
Z0 = #
Σ = {a, b, c}
V = {a, b, Z0}
F = {q2} and P is given above.

Acceptance by NPDA: cont'd

- Example:

  Let w = abbcbba. Trace manually to check whether w is accepted by M or not.

Acceptance by NPDA: cont'd

- <u>Solution</u>

State	Input	Stack
q0	abbcbba	#
q0	bbcbba	a#
q0	bcbba	ba#
q0	cbba	bba#
q1	bba	bba#
q1	ba	ba#
q1	a	a#
q1	λ	#
q2	λ	λ (input is accepted)

Acceptance by NPDA: cont'd

- Exercise 1:

  Construct NPDA for the following language.

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

- Exercise 2:

  Construct an NPDA for accepting the language:

  $L = \{ww^r : w \in \{a, b\}^+ \}$

NPDA – CFG equivalence

- Theorem: If L is a CFL, then there exists NPDA M such that L = L(M).

  proof (by construction)

  Let CFG G = (N, T, P, S)

  Define M = (Q, Σ, V, P', q0, #, {q2}) where:

  Q = {q0, q1, q2}, P' is given by:

  $P'(q0, \lambda, \#) = (q1, S\#)$

  $P'(q1, \lambda, A) = \{(q1, \beta) \mid A \rightarrow \beta \in P \text{ and } A \in N\}$

  $P'(q1, a, a) = (q1, \lambda)$ for all $a \in \Sigma$

  $P'(q1, \lambda, \#) = (q2, \lambda)$

  such that $x \in L(G)$ iff $x \in L(M)$

NPDA – CFG equivalence: cont'd

- Example: Given G = (N, T, P, S) with S = E, T = {a, b, c, +, -, *, /, (,)}, N = {E, F, T} and P:

  $E \rightarrow T \mid E + T \mid E - T$

  $T \rightarrow F \mid T * F \mid T / F$

  $F \rightarrow a \mid b \mid c \mid (E)$

  Construct NPDA M that simulates left most derivation of the grammar (to accept a+(b*c)).

NPDA – CFG equivalence: cont'd

- Theorem: If L = L(M) for some NPDA M, then L is a CFL.

Deterministic PDA (DPDA)

- A DPDA is a 7-tuple machine with the following properties:

a) P(q, a, A) contains at most one element, where q Є Q, a Є Σ U {λ}, A Є V

   (i.e. for any given input symbol and any stack top, at most one move can be made)

a) if P(q, λ, A) ≠ Ø then P(q, a, A) = Ø for all a Є Σ

   (i.e. when a λ-move is possible for some configuration, no input-consuming alternative is available)

- A language accepted by DPDA is called Deterministic CFL or simply deterministic language.

DPDA: cont'd

- Example: The language $L = \{a^n b^n : n >= 0\}$ is a deterministic CFL.

  DPDA $M = (\{q0, q1, q2\}, \{a, b\}, \{0, 1\}, P, 0, \{q0\})$ with

  $P(q0, a, 0) = (q1, 10)$

  $P(q1, a, 1) = (q1, 11)$

  $P(q1, b, 1) = (q2, \lambda)$

  $P(q2, \lambda, 0) = (q0, \lambda)$

  $P(q2, b, 1) = (q2, \lambda)$

NPDA – DPDA equivalence

- In contrast to FSA, DPDA and NPDA are not equivalent. There are CFLs that are non-deterministic.