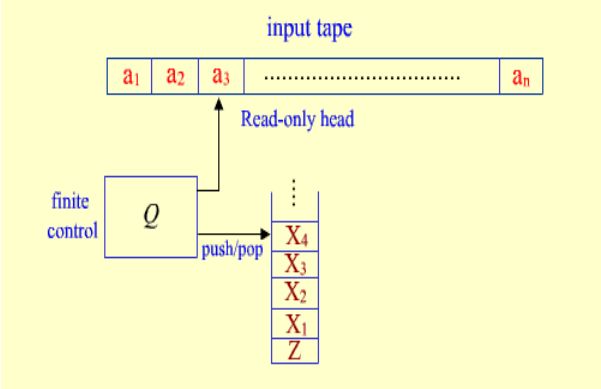
UNIT - IV

Push Down Automata (PDA)

- Regular language can be characterized as the language accepted by finite automata. Similarly, we can characterize the context-free language as the language accepted by a class of machines called "Pushdown Automata" (PDA). Pushdown automation is an extension of the NFA.
- It is observed that FA has limited capability. (In the sense that the class of languages accepted or characterized by them is small). This is due to the "finite memory" (number of states) and "no external memory" involved with them.
- A PDA is simply an NFA augmented with an "external stack memory". The addition of a stack provides the PDA with a last-in, first-out memory management capability. This "Stack" or "pushdown store" can be used to record potentially unbounded information. It is due to this memory management capability with the help of the stack that a PDA can overcome the memory limitations that prevents a FA to accept many interesting languages like $\{a^n b^n \mid n \geq 0\}$.
- Although, a PDA can store an unbounded amount of information on the stack, its access to the information on the stack is limited. It can push an element onto the top of the stack and pop off an element from the top of the stack. To read down into the stack the top elements must be popped off and are lost. Due to this limited access to the information on the stack, a PDA still has some limitations and cannot accept some other interesting languages.
- The deterministic version of PDA accepts only a subset of all CFL's where as non-deterministic version allows all CFL's. The PDA will have an input tape, a finite control, and a stack.



- The input head is read-only and may only move from left to right, one symbol (or cell) at a time. In each step, the PDA pops the top symbol off the stack; based on this symbol, the input symbol it is currently reading, and its present state, it can push a sequence of symbols onto the stack, move its read-only head one cell (or symbol) to the right, and enter a new state, as defined by the transition rules of the PDA.
- PDA are nondeterministic, by default. That is, $\in$- transitions are also allowed in which the PDA can pop and push, and change state without reading the next input symbol or moving its read-only head. Besides this, there may be multiple options for possible next moves.

Formal Definition:

- A pushdown automaton is a ε-NFA with a pushdown stack (last-in, first-out stack).
- Pushdown automata define exactly the context-free languages. There are seven components to a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$,

  Where

  1. Q is a finite set of states.
  2. Σ is a finite set of input symbols (the input alphabet).
  3. Γ is a finite set of stack symbols (the stack alphabet).
  4. δ is a transition function from $(Q \times (\Sigma \cup \{\varepsilon\}) \cup \Gamma)$ to subsets of $(Q \times \Gamma^*)$:

- Suppose δ(q, *a*, X) contains (p, γ). Then whenever P is in state q, looking at the input symbol *a* with X on top of the stack, P may go into state q, move to the next input symbol, and replace X on top of the stack by the string γ.
- The second component, *a*, may be ε in which case P makes the move without looking at the input symbol and does not move to the next input symbol.
- Note that P is nondeterministic so there may be more than one pair in δ(q, *a*, X).

5. q_0 is the start state.
6. Z_0 is the start stack symbol.
7. F is the set of final (accepting) states.

Explanation of the transition function, δ:

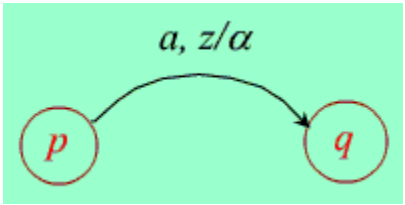
If, for any $a \in \Sigma$, $\delta(q,a,z) = \{(p_1,\beta_1),(p_2,\beta_2),\cdots,(p_k,\beta_k)\}$. This means institutively that whenever the PDA is in state q reading input symbol a and z on top of the stack, it can non deterministically for any i, $1 \le i \le k$

- go to state p_i
- pop z off the stack
- push β_i onto the stack (where $\beta_i \in \Gamma^*$) (The usual convention is that if $\beta_i = X_1X_2\cdots X_n$, then X_1 will be at the top and X_n at the bottom.)
- move read head right one cell past the current symbol a.

If a = $\in$, then $\delta(q,\in,z) = \{(p_1,\beta_1),(p_2,\beta_2),\cdots(p_k,\beta_k)\}$ means institutively that whenever the PDA is in state q with z on the top of the stack regardless of the current input symbol, it can nondeterministically for any i, $1 \le i \le k$,

- go to state p_i
- pop z off the stack
- push β_i onto the stack, and
- leave its read-only head where it is

State transition diagram: A PDA can also be depicted by a state transition diagram. The labels on the arcs indicate both the input and the stack operation. The transition $\delta(p,a,z) = \{(q,\alpha)\}$ for $a \in \Sigma \cup \{\in\}$, $p,q \in Q, z \in \Gamma$ and $\alpha \in \Gamma^*$ is depicted by



Final states are indicated by double circles and the start state is indicated by an arrow to it from nowhere.

Instantaneous Descriptions:

- A configuration or an instantaneous description (ID) of PDA at any moment during its computation is an element of $Q \times \Sigma^* \times \Gamma^*$ describing the current state, the portion of the input remaining to be read (i.e. under and to the right of the read head), and the current stack contents. Only these three elements can affect the computation from that point on and, hence, are parts of the ID.
- The start or inital configuartion (or ID) on input ω is (q_0, ω, z_0). That is, the PDA always starts in its start state, q_0 with its read head pointing to the leftmost input symbol and the stack containing only the start/initial stack symbol, z_0.

The "next move relation" one figure describes how the PDA can move from one configuration to another in one step.

Formally,

$(q, a\omega, z\alpha) \vdash_M (p, \omega, \beta\alpha)$

iff $(p, \beta) \in \delta(q, a, z)$

'a' may be $\in$ or an input symbol.

Language accepted by a PDA M:

There are two alternative definiton of acceptance as given below.

1. Acceptance by final state:

Consider the PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Informally, the PDA M is said to accept its input ω by final state if it enters any final state in zero or more moves after reading its entire input, starting in the start configuration on input ω. Formally, we define L(M), the language accepted by final state to be

$\{ \omega \in \Sigma^* \mid (q_0, \omega, Z_0) \vdash_N^* (p, \in, \beta)$ for some $p \in F$ and $\beta \in \Gamma^* \}$

2. Acceptance by empty stack (or Null stack): The PDA M accepts its input w by empty stack if starting in the start configuration on input w, it ever empties the stack w/o pushing anything back on after reading the entire input. Formally, we define N(M), the language accepted by empty stack, to be

$$\{ w \in \Sigma^* \mid (q_0, w, z_0) \vdash_N^* (p, \in, \in) \text{ for some } p \in Q \}$$

Note that the set of final states, F is irrelevant in this case and we usually let the F to be the empty set i.e. F = Q

2. DETERMINISTIC PUSH DOWN AUTOMATA (DPDA)

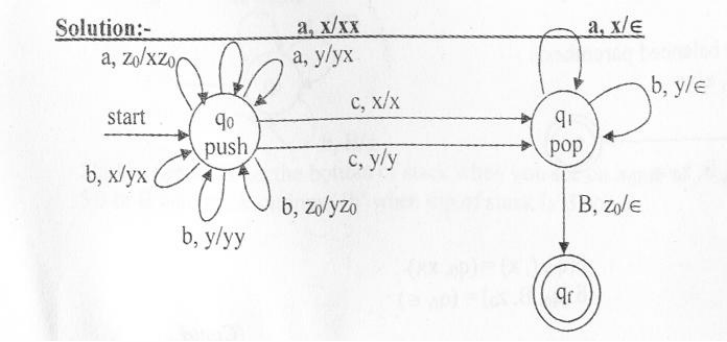
- A PDA is deterministic (DPDA) if there is never a choice for a next move in any instantaneous description.
- A PDA M=(Q, Σ, Γ, δ, q_0, Z_0, F) is deterministic if:
    1. For each q in Q and Z in Γ, whenever δ(q, ε, z) is nonempty, then δ(*q*, *a*, *z*) is empty for all a in Σ.
    2. For no q in Q, Z in Γ and a in ΣU{ε} does δ(*q*, a, z) contain more than one element.

Note: For finite automata, the deterministic and non-deterministic models were equivalent with respect to the languages accepted. The same is not true for PDAs. DPDAs accept only a subset of languages accepted NPDAs. That is NPDA is more powerful than DPDA.

- If L is a CFL, then there exists a PDA, M that accepts L.

Examples:

1. Give a PDA for the language L={wcw^R | w∈$(a+b)^+$}



The transitions are $\delta(q_0, a, z_0)=(q_0,xz_0)$

i.e In state q_0 if input symbol is 'a' and symbol on top of stack is z_0 then remain in state q_0 and push 'x' on to the stack.

$\delta(q_0, a, x)=(q_0,xx)$

$\delta(q_0, a, y)=(q_0,xy)$

$\delta(q_0, b, z_0)=(q_0,yz_0)$

$\delta(q_0, b, x)=(q_0,yx)$

$\delta(q_0, b, y)=(q_0,yy)$

$\delta(q_0, c, x)=(q_1,x)$

$\delta(q_0, c, y)=(q_1,y)$

$\delta(q_1, a, x)=(q_1, \varepsilon)$

$\delta(q_1, b, y)=(q_1, \varepsilon)$

$\delta(q_1, B, z_0)=(q_f, \varepsilon)$ (B – blank space).

This is Deterministic Push down Automata.

2. Design PDA for the following language $L=\{0^n1^{2n} \mid n\geq 1\}$

Solution: PDA $P=(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

PDA $P=(\{q_0,q_1,q_2\},\{a,b\},\{a,z_0\}, \delta,q_0, Z_0, \emptyset)$

The transitions are

$\delta(q_0, a, z_0)=\{(q_1,az_0)\}$

$\delta(q_1, a, a)=\{(q_1,aa)\}$

$\delta(q_1, b, a)=\{(q_2,a)\}$

$\delta(q_2, b, a)=\{(q_1, \varepsilon)\}$

$\delta(q_1, \varepsilon, z_0)=\{(q_1, \varepsilon)\}$

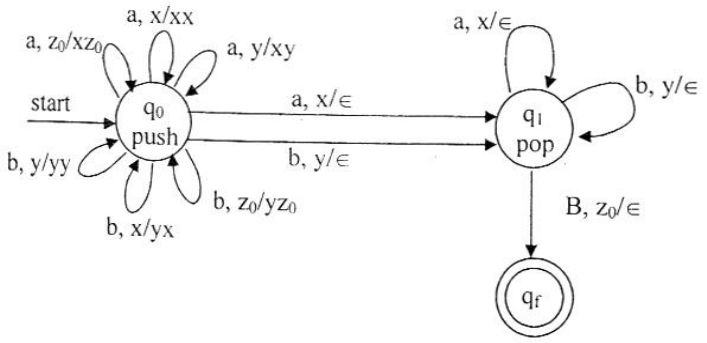
3. NON-DETERMINISTIC PUSH DOWN AUTOMATA (NDPDA)

A PDA is called as non-deterministic, if derivation generates more than one move in the designing of particular task.

Example:

1. Give PDA for $L=\{ww^R \mid w\in(a+b)^+\}$

Solution:



The transitions are

$\delta(q_0, a, z_0)=\{(q_0,xz_0)\}$

$\delta(q_0, a, x)=\{(q_0,xx),(q_1, \varepsilon)\}$

i.e In state q_0 on input symbol 'a' if top of stack is 'x' try two possibilities,

1. Push 'x' on to the top on the assumption that still we have to reach the middle of the string 2. Pop 'x' on the assumption that we are reading the first symbol of the second half of the input string and go to the state q_1.

$\delta(q_0, a, y)=\{(q_0,xy)\}$

$\delta(q_0, b, z_0)=\{(q_0,yz_0)\}$

$\delta(q_0, b, x)=\{(q_0,yx)\}$

$\delta(q_0, b, y)=\{(q_0,yy), (q_1, \varepsilon)\}$

Explanation for this transition also is same as above explanation.

$\delta(q_1, a, x)=\{(q_1, \varepsilon)\}$

$\delta(q_1, b, y)=\{(q_1, \varepsilon)\}$

$\delta(q_1, B, z_0)=\{(q_f, \varepsilon)\}$ (B – blank space).

This is Non-Deterministic Push Down Automata.

2. Design NDPDA for the following $L=\{a^n b^n \mid n\geq 0\}$

Solution: PDA $P=(\{q_0,q_1,q_2,q_3\},\{a,b\},\{R,B,G\}, \delta,q_0,R, \emptyset)$

The transitions are

$\delta(q_0, a, R)=\{(q_1,BR) ,(q_3, \varepsilon)\}$

$\delta(q_0, \varepsilon, R)=\{(q_3, \varepsilon)\}$

$\delta(q_1, a, B)=\{(q_1,BB)\}$

$\delta(q_1, b, B)=\{(q_2, \varepsilon)\}$

$\delta(q_2, b, B)=\{(q_2, \varepsilon)\}$

$\delta(q_2, \varepsilon, R)=\{(q_3, \varepsilon)\}$

4. From a CFG to an equivalent PDA

- Given a CFG *G*, we can construct a PDA *P* such that N(*P*) = L(*G*).
- The PDA will simulate leftmost derivations of G.
- Algorithm to construct a PDA for a CFG
    - Input: a CFG *G* = (V, T, P, S).
    - Output: a PDA *P* such that N(*P*) = L(*G*).
    - Method: Let *P* = ({q}, T, V ∪ T, δ, q, S) where
        1. $\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \rightarrow \beta \text{ is in Q }\}$ for each non-terminal *A* in V.
        2. $\delta(q, a, a) = \{(q, \varepsilon)\}$ for each terminal *a* in *T*.

- For a given input string w, the PDA simulates a leftmost derivation for w in G.
- We can prove that N(P) = L(G) by showing that w is in N(P) iff w is in L(G):
    - If part: If w is in L(G), then there is a leftmost derivation
    - $S = \gamma_1 \Rightarrow \gamma_2 \Rightarrow \ldots \Rightarrow \gamma_n = w$

      We show by induction on i that P simulates this leftmost derivation by the sequence of moves

      $(q, w, S) \vdash^* (q, y_i, \alpha_i)$

      such that if $\gamma_i = x_i\alpha_i$, then $x_iy_i = w$.
    - Only-if part: If $(q, x, A) \vdash^* (q, \varepsilon, \varepsilon)$, then $A \Rightarrow^* x$.

      We can prove this statement by induction on the number of moves made by P.

Example:

1. Construct PDA that accepts the language generated by grammar

S→aSbb|a

Solution: We first transform the grammar into GNF, changing the productions to

S→aSA|a

A→bB

B→b

The corresponding automata will have three states $\{q_0,q_1,q_2\}$, with initial state q_0 and final state q_2. First, the start symbol S is put on the stack by

$\delta(q_0, \varepsilon, z_0)=\{(q_1, sz_0)\}$

The production S→aSA will be simulated in the PDA by removing S from the stack and replacing it with SA, while reading 'a' from the input. Similarly, the rule S→a, should cause the PDA to read an 'a' while simply removing S. Thus, the two productions are represented in the PDA by

$\delta(q_1, a, S)=\{(q_1,SA),(q_1, \varepsilon)\}$

In an analogous manner, the other productions give

$\delta(q_1, b, A)=\{(q_1,B)\}$

$\delta(q_1, b, B)=\{(q_1, \varepsilon)\}$

$\delta(q_1, \varepsilon, z)=\{(q_2, \varepsilon)\}$

5. From a PDA to an equivalent CFG

- Given a PDA *P*, we can construct a CFG *G* such that L(*G*) = N(*P*).
- The basic idea of the proof is to generate the strings that cause *P* to go from state *q* to state *p*, popping a symbol X off the stack, by a non-terminal of the form [*q*X*p*].
- Algorithm to construct a CFG for a PDA
  - Input: a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$.
  - Output: a CFG $G = (V, \Sigma, R, S)$ such that L(*G*) = N(*P*).
  - Method:
    1. Let the non-terminal S be the start symbol of *G*. The other non-terminals in V will be symbols of the form [*p*X*q*] where *p* and *q* are states in Q, and X is a stack symbol in Γ.
    2. The set of productions R is constructed as follows:
       - For all states *p*, R has the production $S \rightarrow [q_0 Z_0 p]$.
       - If $\delta(q, a, X)$ contains $(r, Y_1Y_2 \ldots Y_k)$, then R has the productions
         $[qXr_k] \rightarrow a[rY_1r_1]\,[r_1Y_2r_2] \ldots [r_{k-1}Y_kr_k]$
         for all lists of states $r_1, r_2, \ldots, r_k$.
  - We can prove that $[qXp] \Rightarrow^* w$ iff $(q, w, X) \vdash^* (p, \varepsilon, \varepsilon)$.
  - From this, we have $[q_0Z_0p] \Rightarrow^* w$ iff $(q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon)$, so we can conclude L(*G*) = N(*P*).

Example: 1. Let $M = (\{q_0,q_1\},\{0,1\},\{X,z_0\}, \delta, q_0, z_0, \phi)$ Where δ is given by

$\delta(q_0,0,z_0)=\{(q_0,Xz_0)$

$\delta(q_1,1,X)=\{(q_1,\in)$

$\delta(q_0,0,X)=\{(q_0,XX_0)\}$, $\delta(q_1,\in,X)=\{(q_0,\in)\}$

$\delta(q_0,1,X)=\{(q_1,\in)\}$, $\delta(q_1,\in,Z_0)=\{(q_0,\in)\}$

To construct a CFG G= (V,T, P, S) generating N (M)

Let V={S,$[q_0,X,q_0]$,$[q_0,X,q_1]$,$[q_1,X,q_0]$,$[q_1,X,q_1]$,$[q_0,z_0,q_0]$,$[q_0,z_0,q_1]$,$[q_1,z_0,q_0]$, $[q_1,z_0,q_1]$} and T={0,1}.

To construct the set of productions easily, we must realize that some variable may not appear in any derivation starting from the symbol S. thus we can save some effort if we start with the productions for S, and then add productions only for those variable that appear or the right of some production already in the set . The productions for S are

$S\rightarrow[q_0,z_0,q_0]/\ [q_0,z_0,q_1]$

Next we add production for the variables $[q_0,z_0,q_0]$

These are $[q_0,z_0,q_0]\rightarrow 0[q_0,X,q_0][q_0,z_0,q_0]$

$[q_0,z_0,q_0]\rightarrow 0[q_0,X,q_1][q_1,z_0,q_0]$

There productions are required by

$\delta(q_0,0,z_0)=\{(q_0,Xz_0)\}$

Next the productions for $[q_0,z_0,q_1]$ are

$[q_0,z_0,q_1]\rightarrow 0[q_0,X,q_0][q_0,z_0,q_1]$

$[q_0,z_0,q_1]\rightarrow 0[q_0,X,q_1][q_1,z_0,q_1]$

There are also required by $\delta(q_0,0,z_0)=\{(q_0,Xz_0)\}$

1) $[q_0,X,q_0]\rightarrow 0[q_0,X,q_0]\ [q_0,X,q_0]$

$[q_0,X,q_0]\rightarrow 0[q_0,X,q_1]\ [q_1,X,q_0]$

$[q_0,X,q_1]\rightarrow 0[q_0,X,q_0]\ [q_0,X,q_1]$

$[q_0,X,q_1]\rightarrow 0[q_0,X,q_1]\ [q_1,X,q_1]$ since $\delta(q_0,0,X)=\{(q_0,XX)\}$

2) $[q_0,X,q_1] \rightarrow 1$ since $\delta(q_0,1,X)=\{(q_1,\in)\}$

3) $[q_1,z_0,q_1] \rightarrow \in$ since $\delta(q_1,\in,z_0)=\{(q_1,\in)\}$

4) $[q_1,X,q_1] \rightarrow \in$ since $\delta(q_1,\in,X)=\{(q_1,\in)\}$

5) $[q_1,X,q_1] \rightarrow 1$ since $\delta(q_1,1,X)=\{(q_1,\in)\}$

Resulting productions are

$S \rightarrow [q_0,z_0,q_1],\ [q_1,z_0,q_1] \rightarrow \in,\ [q_1,X,q_1] \rightarrow \in$

$[q_0,z_0,q_1] \rightarrow 0[q_0,X,q_1]\ [q_1,z_0,q_1],\ [q_1,X,q_1] \rightarrow 1$

$[q_1,X,q_1] \rightarrow 0[q_0,X,q_1]\ [q_1,X,q_1],\ [q_0,X,q_1] \rightarrow 1$