Unit-5

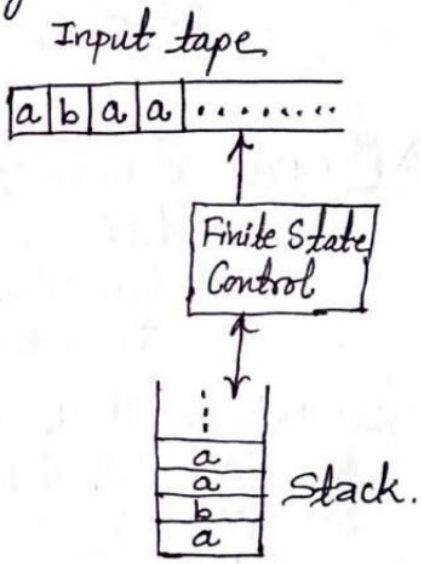
Push Down Automata (PDA)

Introduction:- The context free languages have a type of automaton that defines them. This automaton is called "pushdown automaton" which can be thought as a ε-NFA with the addition of stack. The presence of stack means that, the pushdown automata can ~~rember~~ remember infinity amount of information. However, the pushdown automaton can only access the information on its stack in a last-in-first-out way.

We can define PDA informally as shown in figure:

PDA is an abstract machine determined by following three things:

- → Input table
- → Finite state control
- → A stack.



Each moves of the machine is determined by three things;

- → The current state.
- → Next input symbol
- → Symbol on the top of stack.

The moves consist of;

- → Changing state | Staying on same state.
- → Replacing the stack top by string of zero or more symbols.

Poping the top symbol off the stack means replacing it by ε.

Pushing on the stack means replacing stack's top, say x, by yx.

Formal Definition:

A PDA is defined by seven tuples $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ (Gramma, Edel)

where,

Q = A finite set of states.

Σ = A finite set of input symbols.

Γ = A finite stack Alphabets. (Gramma of uppercase gamma)

δ = transition function

q_0 = The start state.

z_0 = The start stack Symbol

F = The set of Final/Accepting States.

Here, δ takes as argument a triple $\delta(q, a, X)$ where:

i) q is a state in Q.
ii) a is either an Input symbol in Σ or $a = \varepsilon$.
iii) X is a Stack Symbol, that is a member of Γ.

The output of δ is finite set of pairs (p, γ) where;

p is a new state.
γ is a string of stack symbols that replaces X at the top of the stack.

Example:
If $\gamma = \varepsilon$, then the stack is popped.
If $\gamma = X$ then the stack is unchanged.
If $\gamma = YZ$ then X is replaced by Z and Y is pushed onto the stack.

⊛ Representation of PDA:

PDA can be represented by using two techniques:
→ transition table
→ transition diagram.

i) Transition table:- Consider a context free language defined by the grammar $a^n b^n$. Then the transition table used to represent this language is given as;

q_0 on getting ε pops out top of stack

initial stack symbol

Move Number	State	Input	Stack Symbol	Move (δ)
1	q_0	A	Z_0	(q_0, aZ_0).
2	q_0	A	A	(q_0, aa).
3	q_0	ε	A	(q_0, a)
4	q_1	B	A	(q_1, ε)
5	q_1	ε	Z_0	(q_2, Z_0).

State q_0 on getting input A goes to top of stack Z_0

Our last input was A so, top of stack is now A. Now state q_0 on getting another A's stays on q_0 and push input symbol to stack

q_1 on getting input B to top of stack A switches the state q_1 and pop the stack

If state q_1 is reading nothing (ε) and top of stack is Z_0 then it switches to final state q_2

Here, q_0 is the starting state of PDA, q_2 is the final state of PDA and Z_0 is the initial stack symbol.

ii) Transition Diagram:

We can use transition diagram to represent a PDA, where

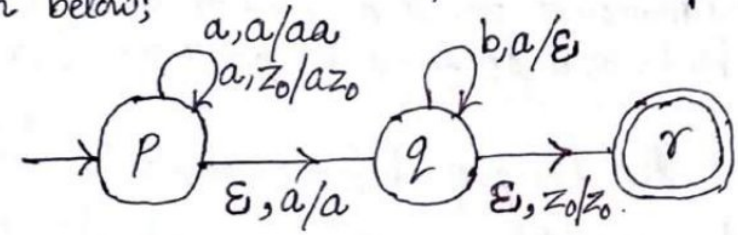
→ Any state is represented by a node in a diagram.

→ Any arrow comming from nowhere to a state indicates the start state and doubly circled states are accepting/final states.

→ The arc corresponds to transition of PDA as;

- Arc labelled as $a, X|\alpha$ means: $\delta(q, a, x) = (p, \alpha)$ for arc from state p to q.

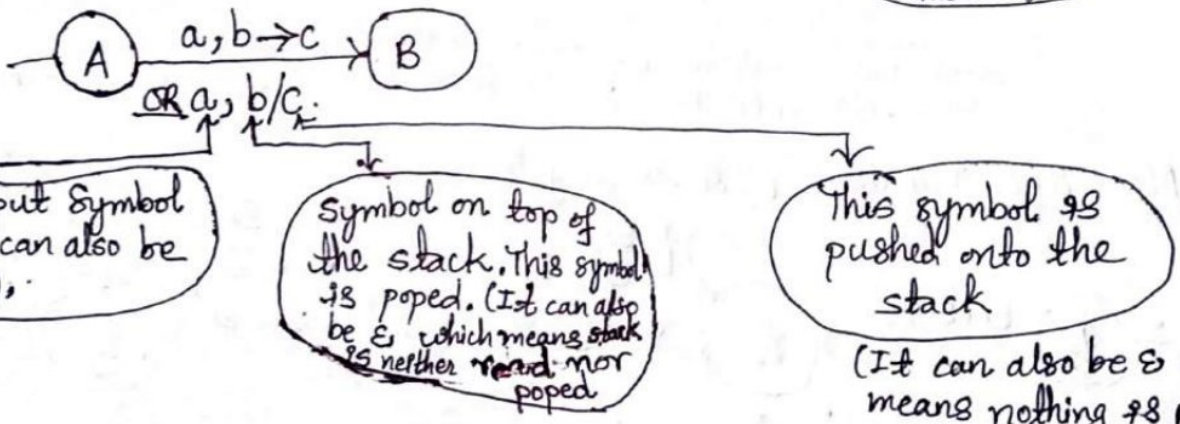
The PDA of $a^n b^n$ can be represented using transition diagram as shown below;



where: p, q, r are state names we can name any

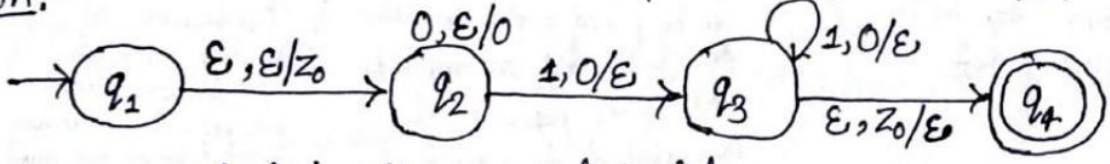
This explains Operation and move of PDA including examples

Graphical Notation of Push down automata:



Example 1: Construct a PDA that accepts $L = \{0^n 1^n | n \geq 0\}$.

Solution:



Here, we started with any initial state q_1. In q_1 we have transitions $\varepsilon, \varepsilon/Z_0$ where ε is input. ε/Z_0 denotes initially stack contains ε(empty) symbols and Z_0 is pushed symbol on stack which denotes it is the first (or bottom most) element of stack. Now we are on state q_2 where we can get input 0, or 1 according to the question. Now on getting 0 as input we do not pop anything from stack (i.e, ε), but we push the comming 0's to stack. Similarly on getting input 1 we go to next state q_3. Here $0/\varepsilon$ denotes 0 is the topmost element of stack which should be poped from the stack and ε denotes

nothing to be pushed onto stack.

Now in state q_3 on getting input 1, check if 0 is on the topmost position of stack, if yes then, pop it and nothing has to be pushed onto the stack. Now on getting input ε, we check if z_0 is on the topmost position of stack, if yes then nothing has to be pushed onto the stack and we reached to final state q_4 accepting the particular string.

Example 2: Construct a PDA that accepts Even Palindromes of the form $L=\{WW^R \mid W=(a+b)^+\}$.

Solution:

We know that palindromes is a word or sequence that reads the same backwards as forwards. e.g. NOON, 123321, abba etc.

Let us understand the language before constructing PDA.

$L=\{WW^R \mid W=(a+b)^+\}$

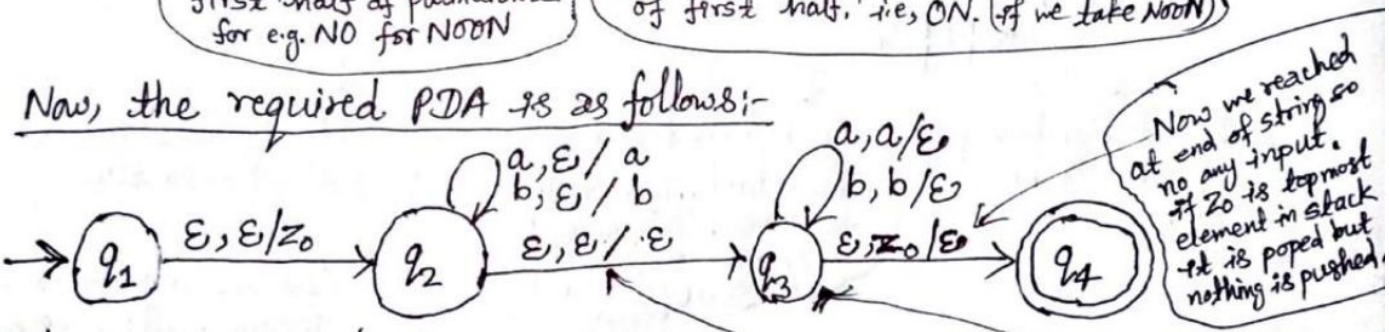
4 letters so even palindrome

this +ve closure shows that there must be atleast one symbol be there, it cannot be empty & a, b are our inputs.

first W represents first half of palindrome for e.g. NO for NOON

second W with R represents reverse of first half. i.e, ON. (if we take NOON)

Now, the required PDA is as follows:-



Now we reached at end of string so no any input. If z_0 is topmost element in stack it is poped but nothing is pushed

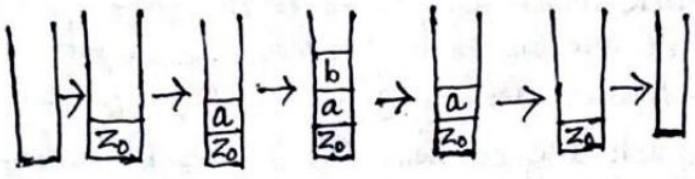
This part we discussed in example 1

q_2 on input a, nothing to be poped and a is pushed to stack. & q_2 on input b nothing is poped and b is pushed to stack

On reaching at middle part of palindrome word we do not input any symbol, we do not pop and we do not need to push anything

On q_3 getting a we check if topmost symbol is a on stack if it is then pop it & nothing to be pushed. Similarly for input b.

Now let we check if abba is accepted:



Hence accepted.

#If stack does not remain empty finally on tracing then that string is not accepted by PDA. (like string abab).

⊛ Instantaneous Description for PDA:

In short we say it as ID

Any configuration of a PDA can be described by a triplet (q, w, r) where,

q is the state
w is the remaning input.
r is the stack contents.

$\in$ → belongs to
ε → epsilon.

Such a description by triple is called an instantenous description of PDA (ID). Instantaneous description is helpful for describing changes in the state, input and stack of the PDA.

Let $P=\{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$ be a PDA. Then, we define a relation $\vdash$, "yields" as; $(q, aw, z\alpha) \vdash (p, w, \beta\alpha)$ if $\delta(q, a, z)$ contains (p, β) and may be a ε. This move reflects the idea that, by consuming "a" from the input, and replacing z on the top of stack by β, we can go from state q to state p.

Example: For the PDA described earlier accepting language ww^R, [see example 2] the accepting sequence of ID's for string 1001 can be shown as; $(q_0, 1001, \varepsilon)$

Here 1001 is input string

$\vdash (q_1, 1001, z_0)$
$\vdash (q_1, 001, 1z_0)$

Here POP if there is mismatch between Input & z_0 content i.e, 0 and 1 mismatch here

$\vdash (q_1, 01, z_0)$
$\vdash (q_1, 1, 0z_0)$
$\vdash (q_1, \varepsilon, z_0)$
$\vdash (q_2, \varepsilon, \varepsilon)$ Accepted.

Therefore $(q_0, 1001, z_0) \vdash^* (q_2, \varepsilon, \varepsilon)$.

$\vdash$ we read it as turnsten. It connects the two or more ID's. * after $\vdash$ symbol denotes moves can be more than one.

⊛ Language of a PDA:

We can define acceptance of any string by a PDA in two ways;

i) Acceptance by final state:

Given a PDA, P the language accepted by final state, L(P) is;

$\{w \mid (q, w, z_0) \vdash^* (p, \varepsilon, r)\}$ where $p \in F$ and $r \in \Gamma^*$.

i.e, finally anything can be in stack no matter

ii) Acceptance by empty stack:

Given a PDA, P the language accepted by empty stack, L(P) is

$\{w \mid (q, w, z_0) \vdash^* (p, \varepsilon, \varepsilon)\}$ where $p \in Q$.

i.e, finally stack should be empty

⊛. Deterministic Pushdown Automata (DPDA):-

A PDA is said to be deterministic if for every input alphabet 'a' (may be ε) and top of stack symbol, PDA has either unique transition (i.e, moves to only one state) or its transition is not defined.

Formally A pushdown automata $P=(Q,\Sigma,\Gamma,\delta,q_0,z_0,F)$ is deterministic pushdown automata if there is no configuration for which P has a choice of more than one moves. i.e, P is deterministic if following two conditions are satisfied;

i) For any $q\in Q, x\in\Gamma$, if $\delta(q,a,x)\neq\Phi$ for some $a\in\Sigma$ then $\delta(q,\varepsilon,x)=\Phi$
i.e, if $\delta(q,a,x)$ is non-empty for some a, then $\delta(q,\varepsilon,x)$ must be empty.

ii) For any $q\in Q$, $a\in\Sigma\cup\{\varepsilon\}$ and $x\in\Gamma$, $\delta(q,a,x)$ has at most one element.

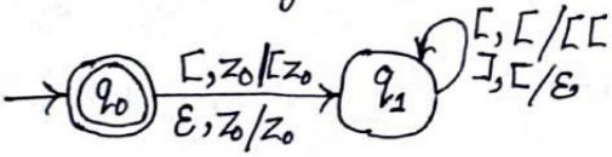
Example 1: Construct DPDA that accepts balanced parenthesis i.e, equal no. of opening and closing braces. eg: [[]], []] etc.

Solution:

The transition table for the given DPDA is as follows;

Move number	state	Input	Stack Symbol	Move(s)
1.	q_0	[	z_0	$(q_1,[z_0)$
2.	q_1	[	[	$(q_1,[[)$
3.	q_1	]	[	(q_1,ε)
4.	q_1	ε	z_0	(q_0,z_0)

The transition diagram for this is as shown below;



Example 2: Construct a DPDA accepting language $L=\{wCw^R \mid w\in(0+1)^*\}$.

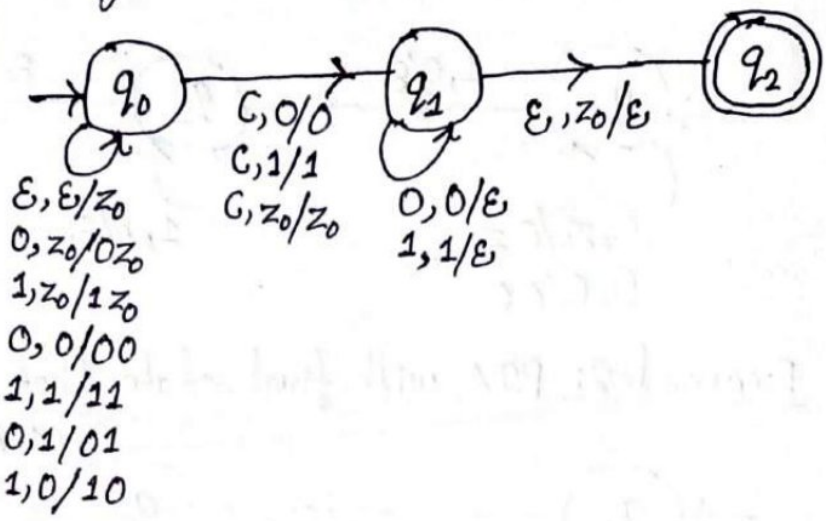
Solution:

$P=(\{q_0,q_1,q_2\},\{0,1\},\{0,1,z_0\},\delta,q_0,z_0,\{q_2\})$ where δ is defined as;

1. $\delta(q_0,\varepsilon,\varepsilon)=(q_0,z_0)$
2. $\delta(q_0,0,z_0)=(q_0,0z_0)$
3. $\delta(q_0,1,z_0)=(q_0,1z_0)$
4. $\delta(q_0,0,0)=(q_0,00)$

5. $\delta(q_0,1,1) = (q_0,11)$
6. $\delta(q_0,0,1) = (q_0,01)$
7. $\delta(q_0,1,0) = (q_0,10)$
8. $\delta(q_0,C,0) = (q_1,0)$
9. $\delta(q_0,C,1) = (q_1,1)$
10. $\delta(q_0,C,z_0) = (q_1,z_0)$
11. $\delta(q_1,0,0) = (q_1,\varepsilon)$
12. $\delta(q_1,1,1) = (q_1,\varepsilon)$
13. $\delta(q_1,\varepsilon,z_0) = (q_2,\varepsilon)$.

Now, the general notation or transition diagram is as follows:



⊛. Non Deterministic PDA (NPDA):

A nondeterministic pushdown automaton (npda) is basically an nfa with a stack added to it. A nondeterministic pushdown automaton is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$. It is an NFA which is a 5-tuple, and added two things to it:

Γ is a finite set of symbols called the stack alphabet and $z_0 \in \Gamma$ is the stack start symbol. Other 5 tuples are as usual we can describe them.

We also need to modify δ, the transition function, so that it manipulates the stack. The transition function for an npda has the form:

$$\delta: Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to \text{finite subsets of } Q \times \Gamma^*.$$

⊛ Construction of PDA by Final State:

A PDA accepts a string when after reading the entire string in the final state. From the starting state we can make moves that end up in a final state with any stack values. The stack values are irrelevent as long as we end up in a final state.

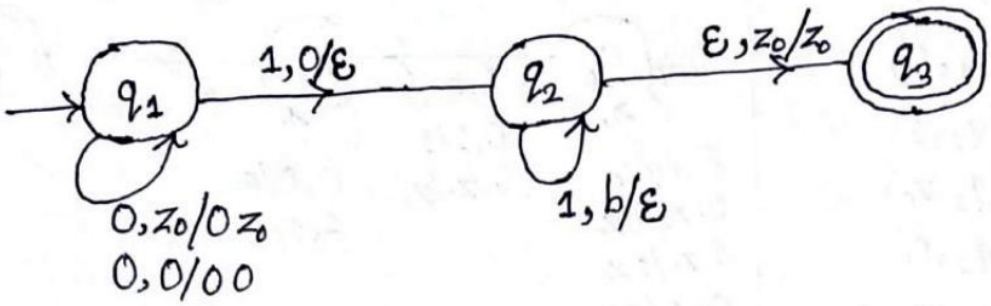
For a PDA $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ the language accepted by the set of final states F as;

$$L(PDA) = \{w \mid (q_0, w, z_0) \vdash^* (q, \varepsilon, \alpha)\} \text{ where } q \in F.$$

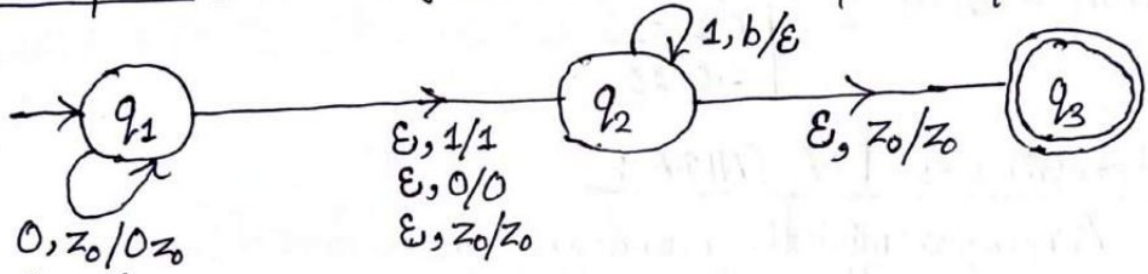
(* denotes moves may be more than one)

(such that — ID consisting of triplets — turnstile notation connecting ID's — another ID consisting triplets)

Example1:- PDA with final state with equal number of zeros (0's) followed by equal numbers ones (1's).



Example2: PDA with final state that accept the palindrome.

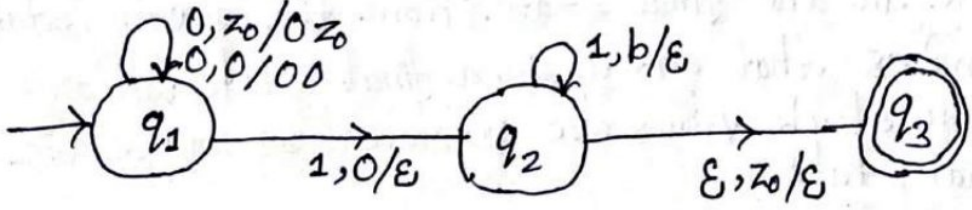


⑧. Construction of PDA by Empty Stack:-

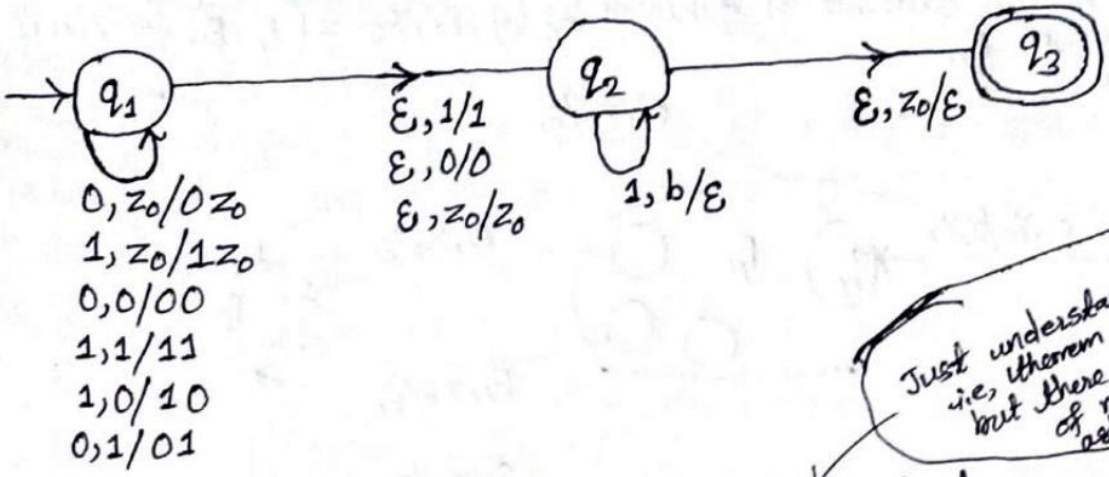
A PDA accepts a string when, after reading the entire string, when PDA has emptied its stack. For a PDA $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ the language accepted by empty stack is;

$$L(PDA) = \{w \mid (q_0, w, z_0) \vdash^* (q, \varepsilon, \varepsilon)\} \text{ where, } q \in Q.$$

Example1: PDA with empty stack with equal number of zeros (0's) followed by equal numbers ones (1's).



Example2: PDA with empty stack that accept the palindrome.



Just understand theories i.e, theorem may be asked but there is less chance of numerical being asked for this and vice-versa

⊛. Conversion of PDA accepting by Empty stack to accepting by final state:

Let us consider the PDA P_N that accepts a language L by empty stack and P_F is it's equivalent PDA that accepts L by final state.

Since no final state F written as we don't care about F during acceptance by empty stack

Theorem: If $L = L(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, z_0)$, then there is a PDA P_F such that $L = L(P_F)$.

Proof: The construction of P_F from P_N is done as follows:

1. Introduce the new symbol X_0 ($X_0 \notin \Gamma$) and place this symbol into the bottom of the stack P_F.
2. Introduce a new start state p_0, whose sole function is to push z_0, the start symbol of P_N, onto the top of stack and enter state q_0 (the start state of P_N).
3. Copy the other states and transition functions of P_N in P_F.
4. Finally add another new state p_f, which is the accepting state of P_F; this PDA transfers to state p_f whenever it discoveres that P_N would have emptied its stack.

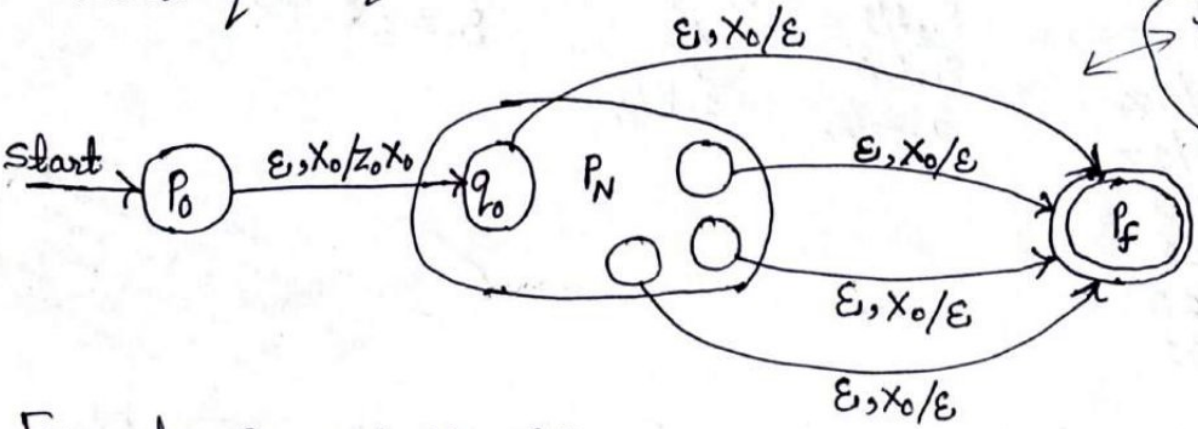
Thus the new constructed P_F has the components:

$$P_N = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\}).$$

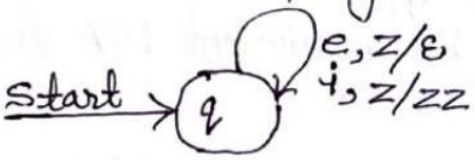
where δ_F is defined by:

1. $\delta_F(p_0, \varepsilon, X_0) = (q_0, z_0 X_0)$
2. For all states q in Q, inputs a in Σ or $a = \varepsilon$, and stack symbols Y in Γ, $\delta_F(q, a, Y)$ contains all the pairs in $\delta_N(q, a, Y)$. i.e, Copy all the transition functions of P_N.

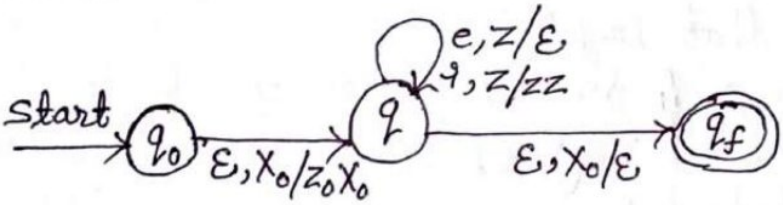
3. Add another transition function $\delta_f(q, \varepsilon, X_0) = (p_f, \varepsilon)$ for every state q in Q.



Example: Convert the following PDA that accepts the string using empty stack into its equivalent PDA that accepts the string by entering into the accepting state.



Solution:



⊛. Conversion of PDA accepting by final state to accepting by empty stack:

Theorem: Let L be $L(P_F)$ for some PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$. Then there is a PDA P_N such that $L = N(P_N)$.

Proof: The construction of P_N from P_F is done as follows:

1. Introduce a new symbol X_0 ($X_0 \notin \Gamma$) and place this symbol into the bottom of the stack of P_N.
2. Introduce a new start state p_0, whose sole function is to push Z_0 (the start symbol of P_F), onto the top of the stack and enter state q_0 (the start state of P_F).
3. For each accepting state of P_F, add a transition on ε to a new state p.

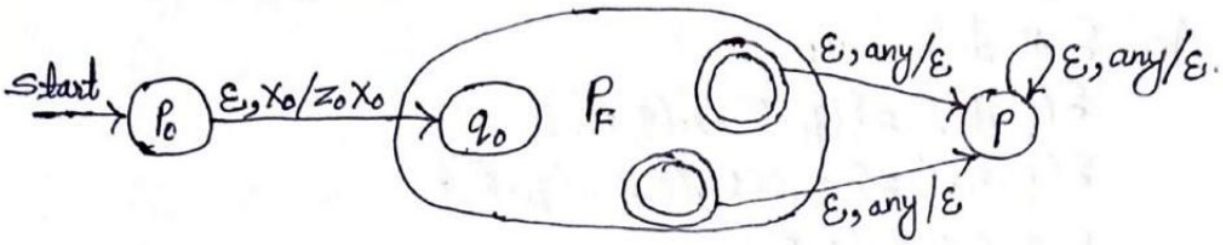
Steps 1 & 2 same as before only P_N and P_F names are interchanged

Thus the new constructed P_N has the components:

$$P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$$

where, δ_N is defined by; (1 & 2 are same as before)

1. $\delta_N(p_0, \varepsilon, X_0) = (q_0, z_0X_0)$.
2. For all states q in Q, inputs a in Σ or $a=\varepsilon$, and stack symbols Y in Γ, $\delta_N(q,a,Y)$ contains all the pairs in $\delta_F(q,a,Y)$. i.e, Copy all the transition functions of P_N.
3. For all accepting states q in F and stack symbols Y in Γ or $Y=X_0$, $\delta_N(q,\varepsilon,Y) = (p,\varepsilon)$. By this rule, whenever P_F accepts, P_N can start emptying its stack without consuming any more input.
4. For all stack symbols Y in Γ or $Y=X_0$, $\delta_N(q,\varepsilon,Y)=(p,\varepsilon)$.



⊛ Conversion of CFG to PDA:

Given a CFG, $G_1 = (V, T, P \text{ and } S)$, we can construct a push down automata, M which accepts the language generated by the grammar G_1. i.e, $L(M) = L(G_1)$.

The machine can be defined as;

$$M = (\{q\}, T, V \cup T, \delta, q, S, \Phi)$$

where,

$Q = \{q\}$ is only the state in the PDA.

$\Sigma = T$

$\Gamma = V \cup T$ (i.e, PDA uses terminals and variables of G_1 as stack symbols).

$z_0 = S$ (i.e, initial stack symbol is start symbol in grammar).

$F = \Phi$

And δ is defined as;

i) $\delta(q, \varepsilon, A) = \{(q, \alpha) / A \to \alpha$ is a production P of $G_1\}$.

ii) $\delta(q, a, a) = \{(q, \varepsilon)$, for all $a \in T\}$.

Example: Convert the grammar defined by following production into PDA;

$S \to 0S1 | A$

$A \to 1S0 | S | \varepsilon$.

Solution: Let $G=(V,T,P \text{ and } S)$ defined by following productions;

$S \to 0S1 | A$

$A \to 1S0 | S | \varepsilon$

PDA equivalent to this grammar is as;

$$M=(\{q_0\},\{0,1\},\{0,1,S,A\},\delta,q_0,S,\phi)$$

where,

$Q=\{q_0\}$

$\Sigma=\{0,1\}$

$\Gamma=\{0,1,S,A\}$

$z_0=S$

$F=\phi$

And δ is defined as;

$\delta(q_0,\varepsilon,S)=\{(q_0,0S1),(q_0,A)\}$

$\delta(q_0,\varepsilon,A)=\{(q_0,0S1),(q_0,S),(q_0,\varepsilon)\}$

$\delta(q_0,0,0)=\{(q_0,\varepsilon)\}$

$\delta(q_0,1,1)=\{(q_0,\varepsilon)\}$

Example 2: Construct a PDA equivalent to following grammar defined by;

$S \to aAA$

$A \to aS | bS | a$. Also trace acceptance of aaabaaaaa.

Solution:

Let $G=(V,T,P \text{ and } S)$ be the grammar defined by the production;

$S \to aAA$

$A \to aS | bS | a$

Now, PDA equivalent to this grammar is as;

$$M=(\{q_0\},\{a,b\},\{a,b,S,A\},\delta,q_0,S,\phi)$$

where δ is defined as;

$\delta(q_0,\varepsilon,S)=\{q_0,aAA\}$

$\delta(q_0,\varepsilon,A)=\{(q_0,aS),(q_0,bS),(q_0,a)\}$

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

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

Now, we trace the acceptance of aaabaaaaa.

$(q_0, aaabaaaaa, S)$

$\vdash (q_0, aabaaaaa, aA)$

$\vdash (q_0, abaaaaa, AA)$

$\vdash(q_0, aabaaaaa, aSA)$
$\vdash(q_0, abaaaaa, SA)$
$\vdash(q_0, abaaaaa, aAAA)$
$\vdash(q_0, baaaaa, AAA)$
$\vdash(q_0, baaaaa, bSAA)$
$\vdash(q_0, aaaaa, SAA)$
$\vdash(q_0, aaaaa, aAAAA)$
$\vdash(q_0, aaaa, AAAA)$
$\vdash(q_0, aaaa, aAAA)$
$\vdash(q_0, aaa, AAA)$
$\vdash(q_0, aaa, aAA)$
$\vdash(q_0, aa, AA)$
$\vdash(q_0, aa, aA)$
$\vdash(q_0, a, A)$
$\vdash(q_0, a)$

In CFG1

$S \to aAA$
$\to aaSA$
$\to aaaAAA$
$\to aaabSAA$
$\to aaababAAAA$
$\to aaababaAAA$
$\to aaababaaAA$
$\to aaababaaaA$
$\to aaababaaaaa$

Left most derivation that we read in regular expression chapter. In this we expand left most non-terminal in our production and we keep expanding until we reach required string.

for eg. A is replaced by aS in 2nd line.

⊛. Conversion of PDA into its equivalent CFG1:

Given a PDA $M=(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$; $F=\phi$, we can obtain equivalent CFG1, $G_1=(V, T, P \text{ and } S)$ which generates the same language as accepted by the PDA M.

The set of variables in the grammar consist of;
→ The special symbol S, which is start symbol.
→ All the symbols of the form $[pXq]$; $p, q \in Q$ and $X \in \Gamma$.
ie, $V=\{S\} \cup \{[pXq]\}$

The terminal in the grammar $T=\Sigma$.

The production of G_1 is as follows;
→ For all states $q \in Q$, $S \to [q_0, z_0, q]$ is a production of G_1.
→ For any states $q, r \in Q$, $X \in \Gamma$ and $a \in \Sigma \cup \{\varepsilon\}$,
If $\delta(q, a, X)=(p, \varepsilon)$ then $[pXq] \to a$
→ For any states $q, r \in Q$, $X \in \Gamma$ and $a \in \Sigma \cup \{\varepsilon\}$,
If $\delta(q, a, X)=(r, Y_1 Y_2 \ldots Y_k)$; where $Y_1, Y_2, \ldots Y_k \in \Gamma$ and $k \geq 0$.
Then for all lists of states $r_1, r_2, \ldots r_k$, G_1 has the production
$[pXq] \to a[rY_1r_1][r_1Y_2r_2] \ldots [r_{k-1}Y_kr_k]$.

This production says that one way to pop X and go from stack q to state r_k is to read "a" (which may be ε), then use some input to pop Y_1 off the stack while going from state r to r_1, then read some more input that pops Y_2 off the stack and goes from r_1 to r_2 and so on.....

Example1: Convert the PDA given below that recognizes a language $L = \{a^n b^n \mid n > 0\}$ defined as;

1. $\delta(q_0, a, z_0) = (q_1, az_0)$
2. $\delta(q_1, a, a) = (q_1, aa)$
3. $\delta(q_1, b, a) = (q_1, \varepsilon)$
4. $\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$. Also show the acceptance of aaabbb.

Solution: Let $G_1 = (V, T, P \text{ and } S)$ be the equivalent CFG1 for the given PDA where,

$V = \{S\} \cup \{[pXq] / p, q \in Q, X \in \Gamma\}$
S = is the start state
$T = \Sigma$

And P is defined by following production;

1. $S \rightarrow [q_0 z_0 q_0] \mid [q_0 z_0 q_1]$
   i.e, $S \rightarrow [q_0 z_0 r_2]$; for r_2 in $\{q_0, q_1\}$.
2. From the fact that $\delta(q_0, a, z_0)$ contains (q_1, az_0), we get production $[q_0 z_0 r_2] \rightarrow a [q_1 a r_1] [r_1 z_0 r_2]$; for r_1 in $\{q_0, q_1\}$.
3. From the fact that $\delta(q_1, a, a)$ contains $(q_1 aa)$, we get production $[q_1 a r_2] \rightarrow a [q_1 a r_1] [r_1 a r_2]$ for r_1 in $\{q_0, q_1\}$.
4. From the fact that $\delta(q_1, b, a)$ contains (q_1, ε), we get $[q_1 a q_1] \rightarrow b$
5. From the fact that $\delta(q_1, \varepsilon, z_0)$ contains (q_1, ε), we get $[q_1 z_0 q_1] \rightarrow \varepsilon$

Now the acceptance of aaabbb can be shown as;

$S \rightarrow [q_0 z_0 r_2]$
$\rightarrow a [q_1 a r_1] [r_1 z_0 r_2]$
$\rightarrow aa [q_1 a r_1] [r_1 a r_2] [r_1 z_0 r_2]$

$\rightarrow aaa[q_1 a r_1][r_1 a r_2][r_1 a r_2][r_1 z_0 r_2]$

$\rightarrow aaab[r_1 a r_2][r_1 a r_2][r_1 z_0 r_2]$

$\rightarrow aaabb[r_1 a r_2][r_1 z_0 r_2]$

$\rightarrow aaabbb[r_1 z_0 r_2]$

$\rightarrow aaabbb\ \varepsilon = aaabbb.$

Example 2: Convert the PDA $P=(\{p,q\},[0,1],[X,Z_0],\delta,q,Z_0)$ to a CFG if δ is given by;

$\delta(q,1,Z_0)=(q,XZ_0)$

$\delta(q,1,X)=(q,XX)$

$\delta(q,0,X)=(q,X)$

$\delta(q,\varepsilon,Z_0)=(q,\varepsilon)$

$\delta(q,0,Z_0)=(q,Z_0).$

Solution:

The equivalent grammar can be written as;

$G_1=(V,\Sigma,P,S)$ where P consists of productions as;

$S\rightarrow[qZ_1r_2]$; r_2 in $\{p,q\}$.

$[qZ_0r_2]\rightarrow 1[qXr_1][r_1Z_0r_2]$; for r_1 in $\{p,q\}$.

$[qXr_2]\rightarrow 1[qXr_1][r_1Xr_2]$; for r_1 in $\{p,q\}$.

$[qXr_2]\rightarrow 0[pXr_2]$; for r_2 in $\{p,q\}$.

$[qXq]\rightarrow\varepsilon$

$[pXp]\rightarrow 1$

$[pZ_0r_2]\rightarrow 0[qZ_0r_2]$; for r_2 in $\{p,q\}$.