Pushdown Automata (Introduction):

A pushdown Automata (PDA) is a way to implement a context free Grammar in a similar way we design finite Automata for Regular Grammar.

→ It is more powerful than FSM

→ FSM has a very limited memory but PDA has more memory.

→ PDA = finite state machine + "A Stack"

Regular Grammar:

Noam chomsky gave a Mathematical model of Grammar which is effective for writing computer languages.

The four types of Grammar according to Noam chomsky are:

Grammar Type	Grammar Accepted	Language Accepted	Automata
Type-0	Unrestricted Grammar	Recursively Enumerable lang	TM
Type-1	Context Sensitive	CSL	LBA
Type-2	Context Free	CFL	PDA
Type-3	Regular Grammar	RL	Finite State Automata.

Let us see important point related to PDA

(1) It is more powerful than FSM -

There are limitation in FSM i.e due to limited memory. There were certain Grammar or certain languages that couldn't be design FSM due to there limited Memory

So the PDA is more powerfull than FSM coz those languages couldn't be design using fs can be design using the push down Automata

(2) FSM has a veey limited memory but PDA has more memory.

(3) PDA = "finite state machine" + "A Stack"
pushdown Automata it consist of fsm and extra thing is a stack.
This stack has infinite memory and this is the higher power of PDA

- A stack helps PDA to behave like more powerfull than finite state machine..

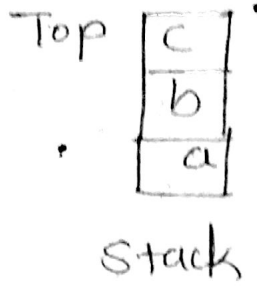
Now Let us see what is a stack? and what are the main operation that stack can perform. If you studied about Data structure you may know that what is stack but let me explain it to you what is stack over here...

A Stack: A stack is a way we arrange elements one on Top of another.
A stack has Two basic operation.

PUSH: A new element is added at the Top of the stack

Pop: The Top element of the stack is read and removed.

So Let us try to understand push and pop operation.



Stack

→ So as definition of stack a, b, c arrange one on top of another

. Top most element denoted by, . Top

Eg:
You can imagine stack of book or stack of coins

push
When you want to insert book you have to insert it on top of the book

Stack

ex: 

stack

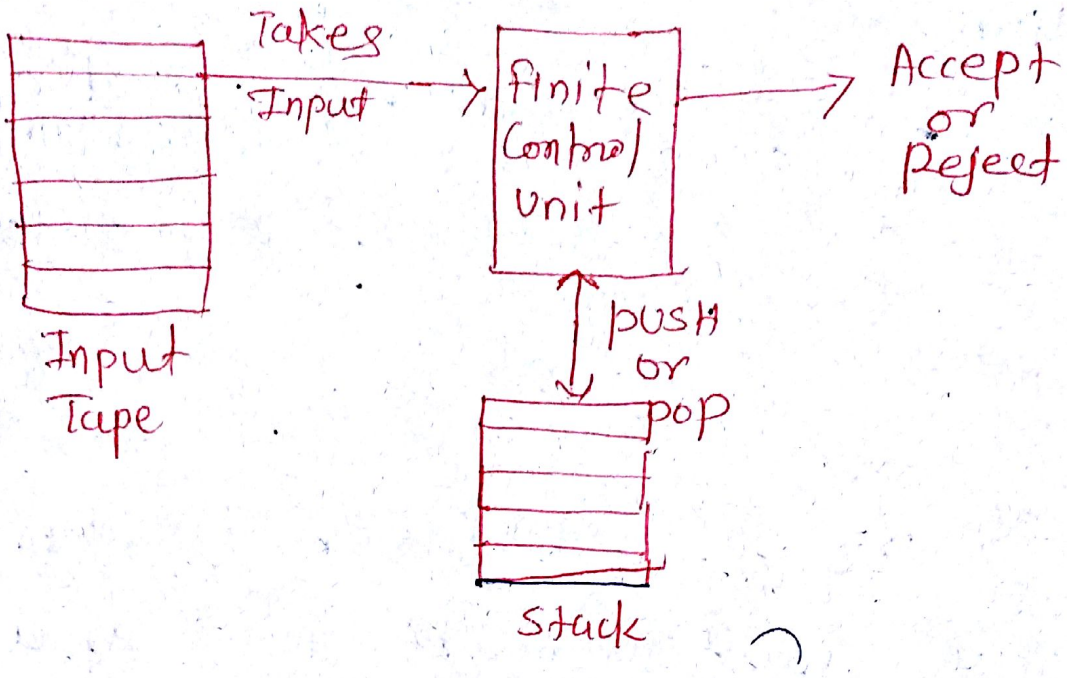
pop:
If you want to remove book you have to remove it from top most. poping means removing Top most element. when top most element 'd' is removed then top most element will be c. again if you perform pop operation top will be pointing to 'b'

Now if you understood the stack let us understand A pushdown Automata components

A pushdown Automata has 3 components

1) An input Tape
2) A finite control unit
3) A stack with infinite size

Here we have diagrams depicts the 3 Components



A PDA can be formally described as a 7-Tuple
$(Q, \Sigma, S, \delta, q_0, I, F)$ -

Q - is the finite number of states

Σ - is input alphabet

S - is stack symbol (It can be denoted by Γ

δ - is the transition function :

$$Q \times (\Sigma \cup \{\varepsilon\}) \times S \times Q \times S^*$$

q_0 - is the initial state ($q_0 \in Q$)

I - is the initial stack top symbol
($I \in S$)

F - is a set of accepting states
($F \in Q$)

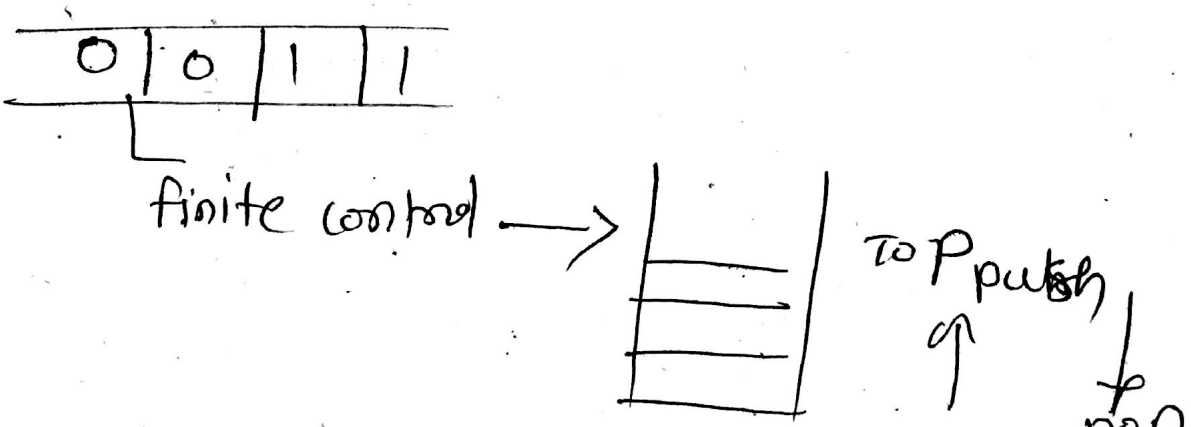
OR

$(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

FSM:

ex: of FSM $0^n 1^n$ $n>0$

Here 0^n no. of 0's Infinite it can be recognize by finite Automata but when 1^n it means in this language 0 is followed by 1 and number of 0's and is are equal. so here comparasion is there and when there is comparision fA is get failed. because it has finite memory It can not be remember like you have to count '0' then Count '1' so for this finite memory cannot work you require extra memory i.e stack which is in PDA



Here when you have 0 in i/p string push on to the top. and when you have 1 pop the the element from the stack and finally ϵ remains then you can say $0^n 1^n$. so it is more powerfull

Gamma (uppercase Γ, lowercase γ) is the third letter of the Greek alphabet in the system of Greek numerals it has a value of 3

where

Q is a finite set of states

Σ - is the input alphabet;

Γ - is the stack alphabet

δ - $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$

s.t. $\delta(q, a, X)$ is finite is the transition function;

- $q_0 \in Q$ is the start state
- $Z_0 \in \Gamma$ is the start symbol and
- $F \subseteq Q$ is the set of final states.

Defn Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. A triple $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$ is an instantaneous description of P we denote $Q \times \Sigma^* \times \Gamma^*$ by $ID(P)$

Defn - Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA, we define the binary relation $\vdash_P$ on $ID(P)$ such that

$(q, \alpha w, X\beta) \vdash_P (p, w, \alpha\beta) \iff (p, \alpha) \in \delta(q, a, X)$

for $p, q \in Q$, $a \in \Sigma \cup \{\epsilon\}$, $w \in \Sigma^*$, $X \in \Gamma$, $\alpha, \beta \in \Gamma^*$. $\vdash_P^*$ is the reflexive transitive closure of $\vdash_P$

PDA

DPDA (Deterministic)

NDPA (Nondeterministic)

Transition Symbol δ : By transition Symbol we can categories ~~whether~~ either PDA is deterministic or Nondeterministic

1) If move's on the particular single state then it is Deterministic PDA

2) If move for the particular symbol goes to more than one state then that is known as Non Deterministic PDA

(A) No move is possible if stack is empty

Ex:

$Q = \{q_0, q_1, q_f\}$, $\Sigma = \{a, b\}$,

$\Gamma = \{a, Z_0\}$, $F = \{q_f\}$

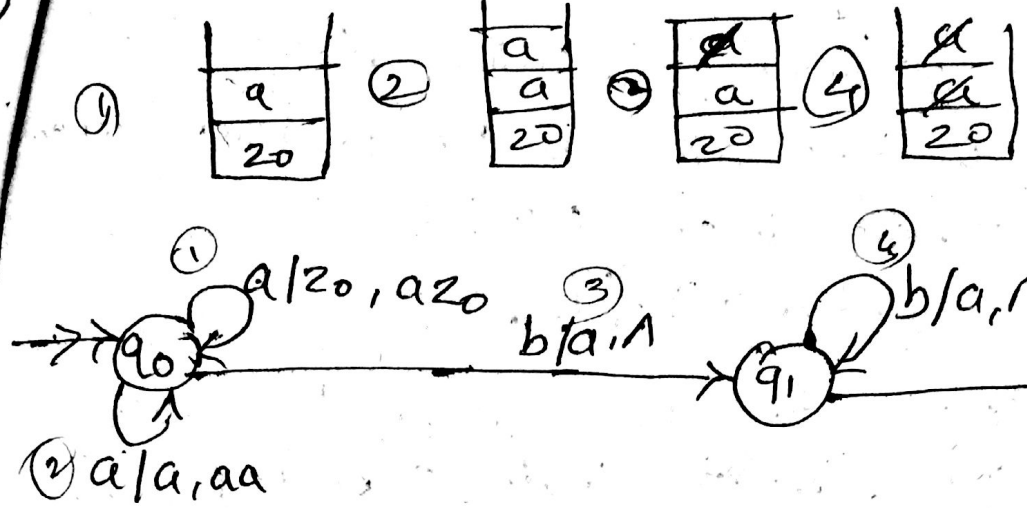
$\delta \rightarrow \delta(q_0, a, Z_0) = (q_0, aZ_0)$

~~δ~~ $\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \wedge)$

$\delta(q_1, b, a) = (q_1, \wedge)$

$\delta(q_1, \wedge, Z_0) = (q_f, \wedge)$



Now Here Λ null symbol means poping element

(*) Z_0 means final state

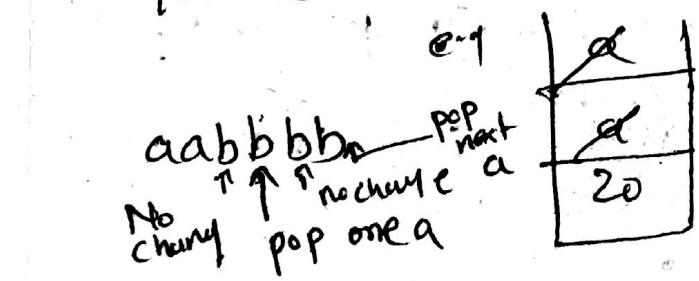
you can say that this language or grammer is for $a^n b^n$ you can check for aaabbb or any string this can be acceptable by PDA but not acceptable by FSM

Ex. Give a PDA to accept the following language $L = \{a^n b^{2n} \mid n \geq 1\}$ by final state

→ set of string accepted {abb, aabbbb, ...}

so Now to check a^n & b^{2n} no change in a when we have first b. but you should pop a when you get second b

e.g



if input string consumed

a a bbbb
↑ ↑

as we looking for aa bb it means

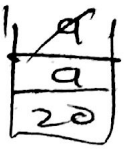
$a^n b^{2n}$ b^{2n} means two times 'n' as in a^n

and when we got first b No push operation but go to next state No pop operation

i.e $\delta(q_0, b, a) = (q_1, a)$

in next state or when we got second b then go to next state and pop operation must be performed

so $\delta(q_1, b, a) = (q_2, \epsilon)$



① for 'a's in I/p remain in same state. and a's push on to the stack

and

② first b go to state q_1 No change in top of the stack performed

③ for 2nd b we go to state q_2

④ for 3rd b we should go to q_1 so for even no. of be we can go to state q_2

⑤ for odd no. of b we should go to q_1

⑥ for even no. of b's we should go to q_2

⑦ for even b pop operation should be perfor

$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, a, a) = (q_0, aa)$

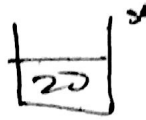
$\delta(q_0, b, a) = (q_1, a)$

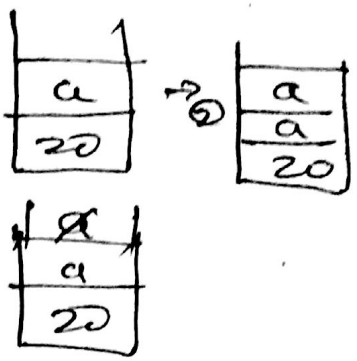
$\delta(q_1, b, a) = (q_2, \epsilon)$

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

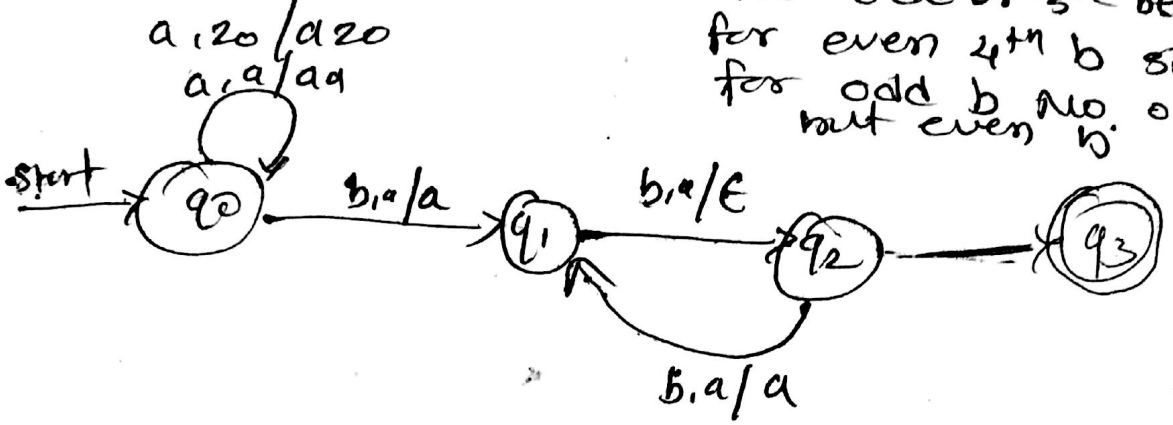
~~$\delta(q_2, b, a) = (q_1, a)$~~

$\delta(q_2, \epsilon, z_0) = (q_3, z_0)$





first b no change in stack
for odd or 3rd be q0 to q1
for even 4th b state q2
for odd b No operation
next even b pop operation



q is current state b is i/p Top is = a

$\delta(q_1, b, a) = (q_2, \epsilon)$ means pop

PDA, $P = (\{q_0, q_1, q_2, q_3\},$

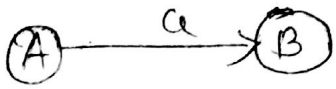
$\Sigma = \{a, b\}$

$\{a, z_0\}$

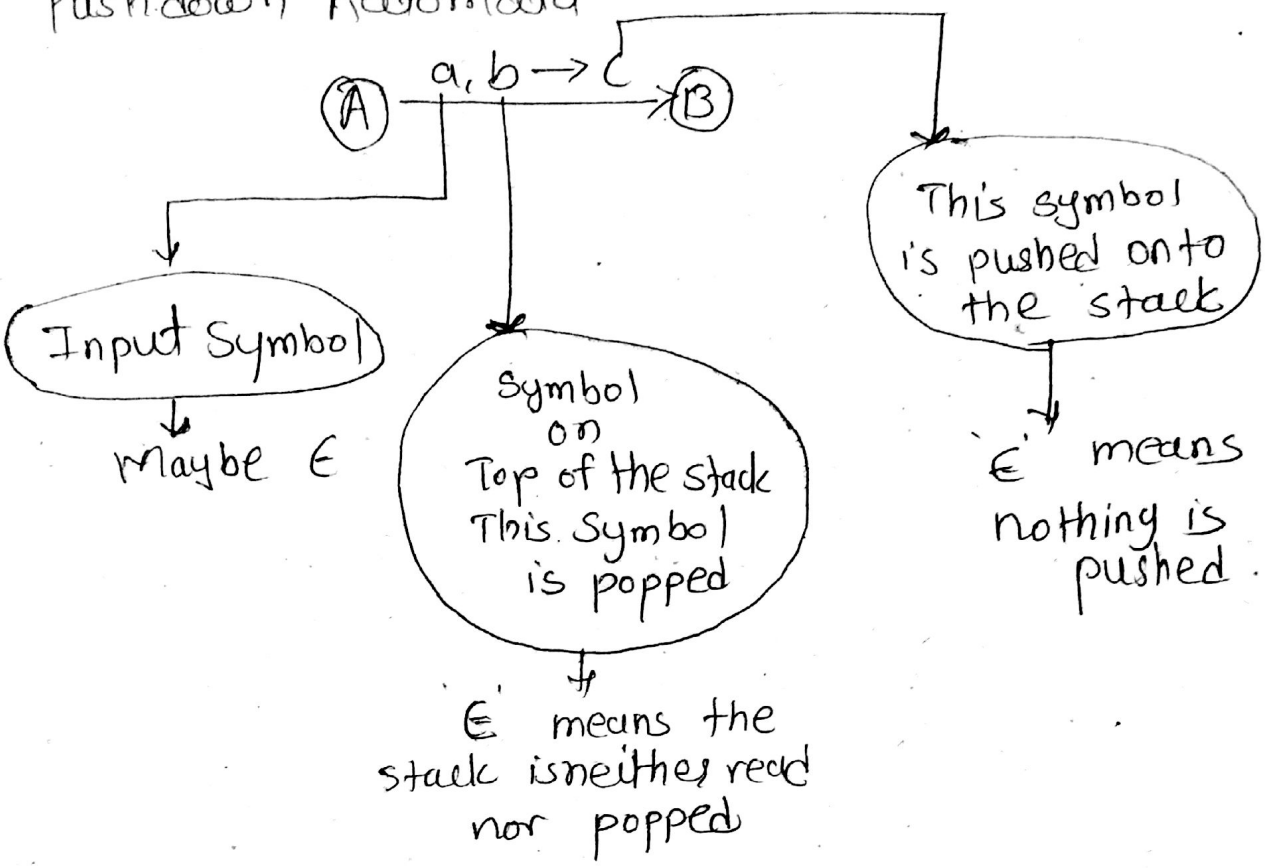
$\delta, q_0, z_0, q_3\}$

Pushdown Automata (Graphical Notation)

finite State Machine

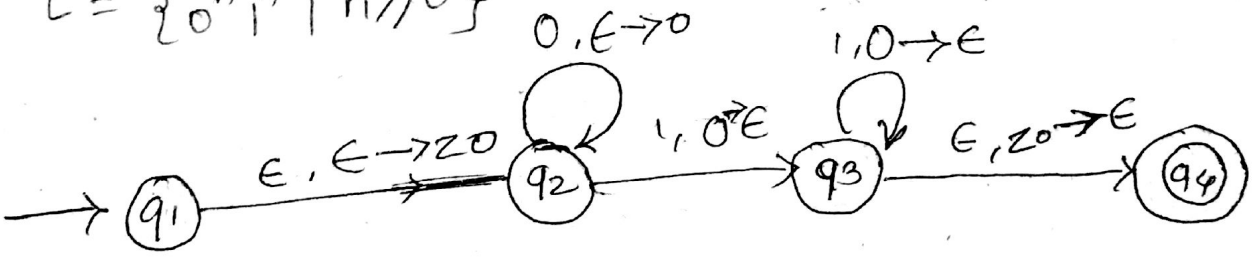


Pushdown Automata

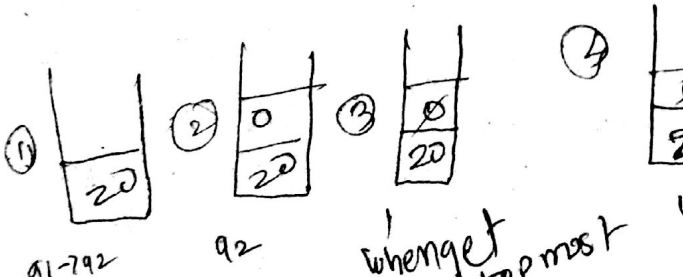


Example Construct a PDA that accepts $L = \{0^n 1^n \mid n \geq 0\}$

0011



00!!



q1→q2 q2 when get 1 pop topmost 0 from top

when get next 1 then pop topmost element

→ After that ∈ means final state and get Z0.

(1) In push down Automata when we reach to final state string get accepted

(2) when we find stack is empty, we see that z0 is final element then input string get accepted.

above given is Non-deterministic push Down Automata.

Ex pushdown Automata Example (Even palindrome)

construct a PDA that accepts Even palindromes of the form $L = \{ ww^R \mid w = (a+b)^+ \}$

example of palindrom String - NooN
NO LEMON NO MELON
123321
abba

odd palindrom → RACECAR
MOM
BOB
Madam

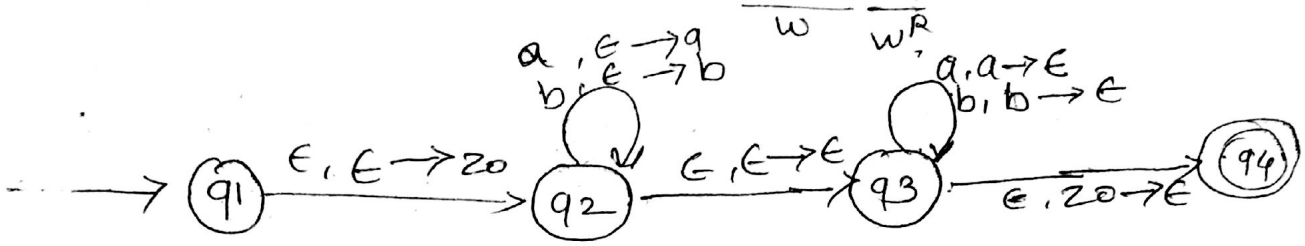
So in above given example we want to desing even palindrom; but we are toying to design palindrome with symbol a and b only

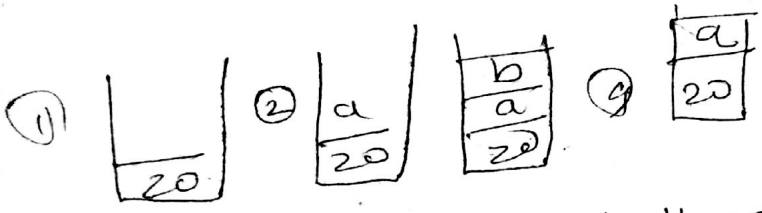
+ positive clousre - If you see positive clousre it means that it should contain's at least one symbol & it can not be empty and ∈ not allowed

* If you see * (star) symbol it means you can have empty symbol or ∈ and

Ex: $L = \{w\, w^R \mid w = (a+b)^+\}$

w - means word means first half of string
w^R - w superscript R means reverse of string like NOON





① $q_1 \rightarrow q_2$ ∈ we don't pop anything but we push Z0 to denote first element of the stack.

② q_2 selfloop - we can either get i/p a or b for i/p a we don't pop but we push a on top of the stack

③ i/p b - push b to stack

Then afther q1, & q2 we assume we reach to midpoint if we consider NOON we reach to midlepoint and we don't push or pop any symbol but directly go to next state q3

④ and in q3 we get either a or b if you get i/p a pop a from stack & don't push anything also. If you get i/p b pop it from stack

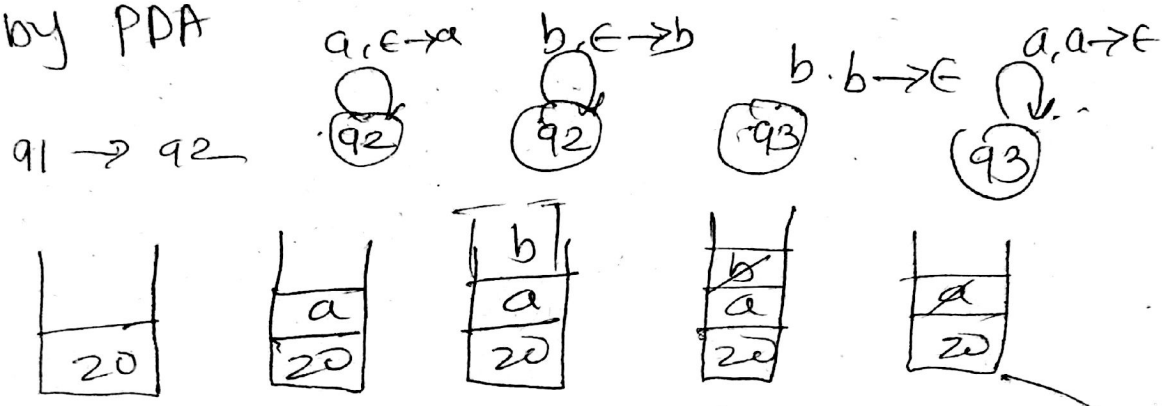
This is repeated still our string is consume or string is finished when we reach to end of the string ~~i.e~~ that is we don't

$\epsilon, z_0 \rightarrow \epsilon$

read any string but we get z_0 top most element of stack. then we don't push any thing and we reach to final state

ex a b b a string should get accepted

by PDA

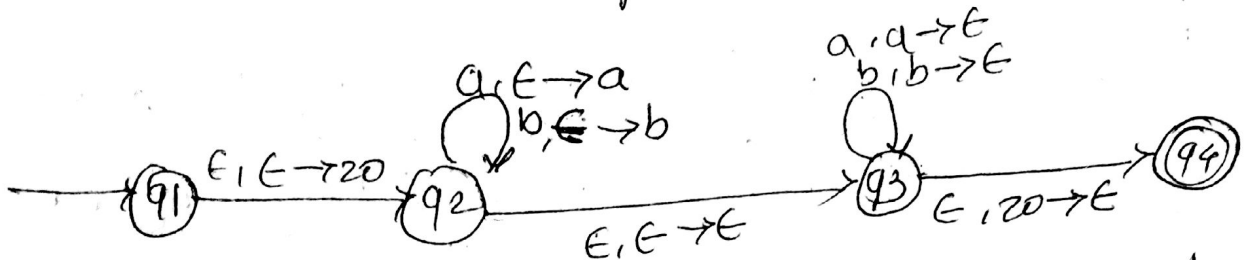


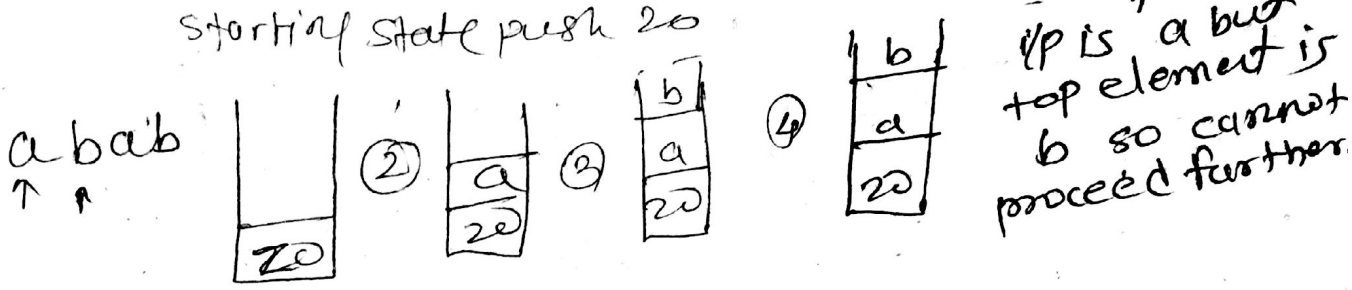
Here if you see b on to top of the stack pop it or delete it

pop a from stack then we got z_0 i.e end of the string

since, we reach the final state and since our stack is empty then palindrom string is accepted

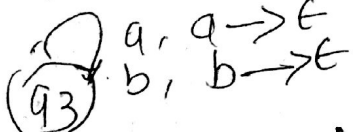
Consider abcab is not palindrome string
Let's start with digram (same diagram)





first step - push Z0 ont to stack
then
see 2nd step q2 a, ∈ → a push a
b, ∈ → b push b to the stack

Now we reach to ~~need~~ midpoint then here we don't push element, read element go to next state q3



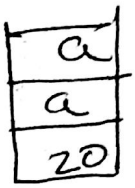
Now in cab|ab we reach to cab;ab Here we got a then we have to check top most element is a but top most element is b so this state cannot proceed so this state can not reach final state string will not get accepted

✗ Construct a DPDA for $L = \{wcw^R \mid w \text{ is in } (0+1)^+\}$

Construct a PDA for $L = \{a^n b^m c^n \mid n \geq 1, m \geq 1\}$

a^n & c^n must be equal b can be any num. of

like aabbbcc



$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

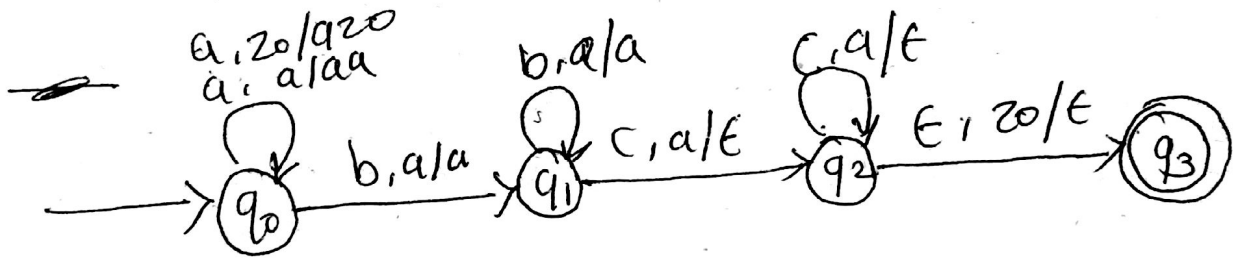
$\delta(q_0, b, a) = (q_1, a)$ No change operation

$\delta(q_1, b, a) = (q_1, a)$

$\delta(q_1, c, a) = (q_2, \epsilon)$ repeat pop

$\delta(q_2, c, a) = (q_2, \epsilon)$

$\delta(q_2, \epsilon, Z_0) = (q_3, \epsilon)$



PDA $P = \{\{q_0, q_1, q_2, q_3\}, \{a, b, c\}; \{a, Z_0\}, \delta, q_0, Z_0, q_f\}$

① String has 2 a's in the begining and 2 c's at the end
② Insert all a on to the stack
③ for b's dont do any operation, because there is no need to match b with any symbol
④ and for each c do pop operation
⑤ Initial Z_0 pushed on stack

aabbcc

$\delta(q_0, a, z_0) = (q_0, az_0)$ – push a onto stack z_0 is already on top

$\delta(q_0, a, a) = (q_0, aa)$ – push a onto stack a is already on stack remain in same state.

$\delta(q_0, b, a) = (q_1, a)$ Here b is next symbol so go to next state and no push or pop operation

$\delta(q_1, b, a) = (q_1, a)$ Here also no change operation repeat state for b

perform repet for 3rd b

$\delta(q_1, c, a) = (q_2, \epsilon)$ → Now first c symbol is there go to next state and pop one a from stack

$\delta(q_2, c, a) = (q_2, \epsilon)$ – for second c pop next a and remain in same state and

$\delta(q_2, \epsilon, z_0) = (q_3, \epsilon)$ Here Now No element then empty stack and reach to final state

Equivalence of PDA's and CFG's

Page:
Date:

Pushdown automata's are used to accept the languages generated by the (CFG) Context free Grammar.

Theorem: A language is Context free iff some Pushdown Automata recogni-zes it.

So Here we have to prove that language generated by CFG and language accepted by PDA are of the same class of languages

Proof:

Part 1: Given a CFG, show how to construct a PDA that recognizes it.

Part 2: Given a PDA, show how to construct a CFG that recognizes the same language.

part 2 is just reverse of part 1. In part 1 CFG is given and in part 2 PDA is given.

Part 1 Given a Grammar;

$S \rightarrow BS \mid A$

$A \rightarrow 0A \mid \epsilon$

$B \rightarrow BB \mid 1 \mid 2$

find or build a PDA

So Here by construction we are going to prove that CFG = PDA

If we are able to construct a PDA for given Grammar then we can prove part hence we are able to prove the Theorem — A language is context free iff some Push down Automata recognizes it.

Take Left most Derivation as we studied in RG

$S \rightarrow BS$
$\rightarrow BB1S$
$\rightarrow 2B1S$
$\rightarrow 221S$
$\rightarrow 221A$
$\rightarrow 221\epsilon$
$\rightarrow 221$

Non

We expanding the Left most Terminal until we reach the required string.

- During this LD any of the form's you get is known as Left sentential form.

Now Let us check How can we construct PDA for production that we have

General form:

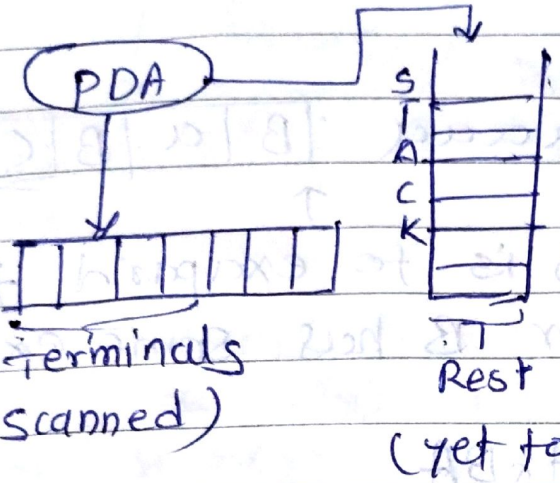
eg

$aaaa \; BaBC$

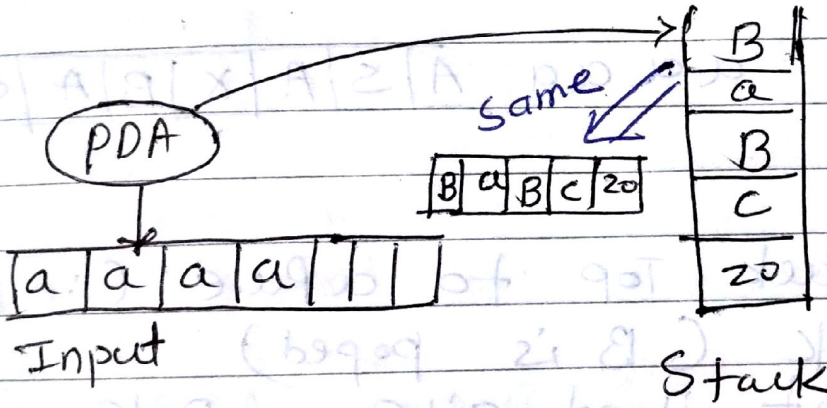
Terminals | Rest

In Terminals — All Terminal Symbol)
Rest — Non Terminal & Terminal or where we can expand.

So in diagram PDA has two part's Terminals that has Terminal Symbol and stack has Rest symbol



Let put General form in stack.



steack is filled only both ≡ ⊟ are same sequential & continuous

we put Z0 at bottom of the stack to know the end of the stack.

Left Most Desivation

S → ---a a a a B a B C

This one of the staye while Left Most Desivation

E.g Rule

$B \rightarrow ASA \times BA$ ...→

Let see this one of the Rule

so Here

$S \rightarrow \ldots aaaa \boxed{B|a|B|C} \rightarrow$

↑

Now Here B is to exapand then see that whether B has some expansions or not.

so $B \rightarrow ASA \times BA$

expand B to right hand side

i.e

$\rightarrow \ldots aaaa \boxed{A|S|A|X|B|A|a|B|C}$

as it is

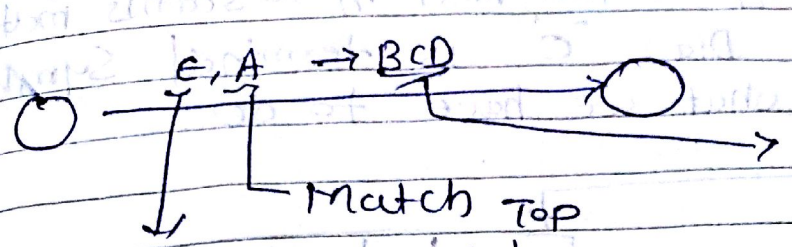
→ Match stack Top to a Rule (i.e B is Top)

→ Pop stack (B is poped)

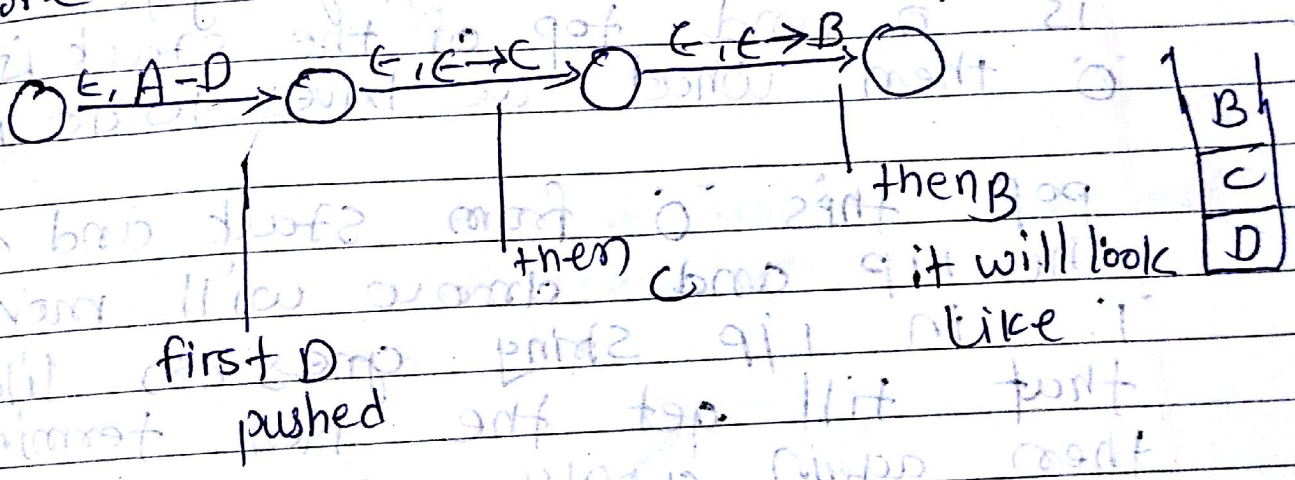
→ Push Right Hand side of Rule onto stack. (i.e in place of $B \rightarrow A|S|A|\times|B|A$ is pushed.

All right Now draw the transition Diagram.

Rule A → BCD Add this to the PDA

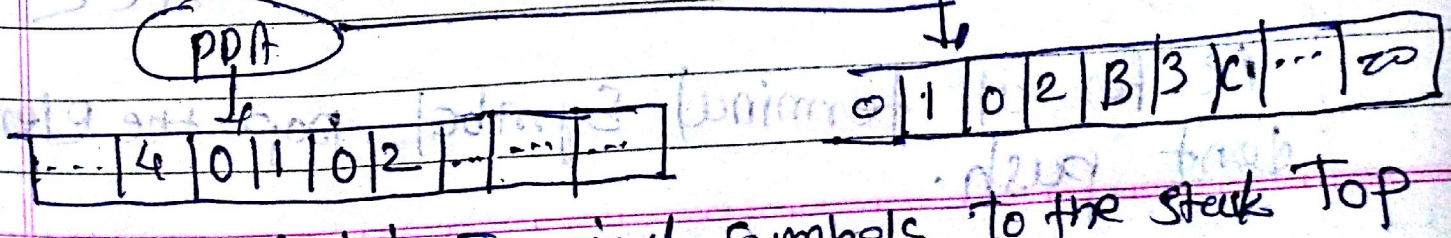


we cannot pused more than one element one time so pushed one by one using more states.



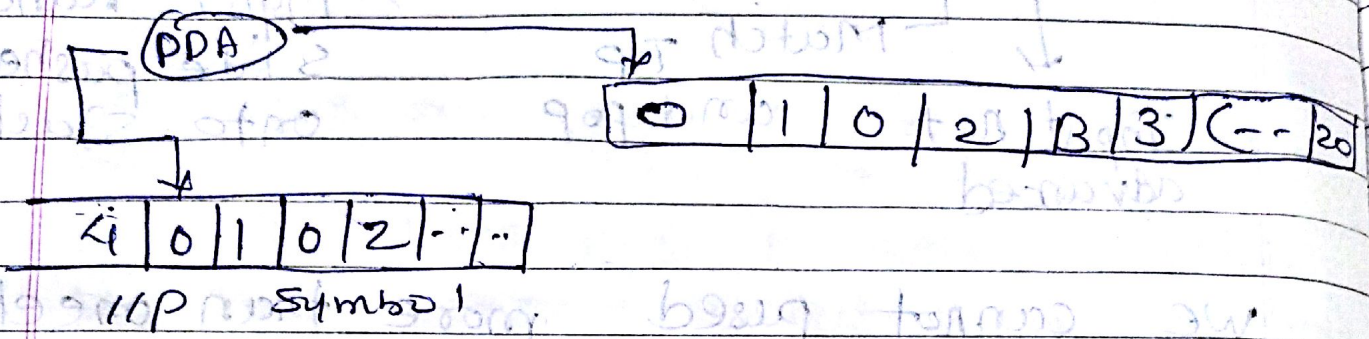
1- Rule A → 0|02B3C Non terms

So we know Rule if we are getting A on to the top of the stack we have to push right hand side



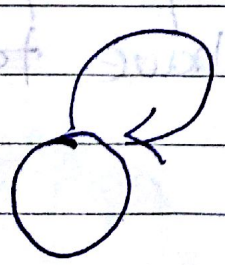
Match Terminal Symbols to the stack Top

If we encounter Terminal Symbol then look at the i/p if the next i/p to scan is matched to toppmost. In above Dig '0' is terminal Symbol on top what we have to do



So Here 0 is matched at both the places i.e next i/p going to scan is '0' and top of the stack is also '0' then what we have to do is

pop this '0' from stack and advance the i/p and arrow will move to '1' in i/p string goes on like that till get the Non terminal then again apply expansion rule of Non terminal go on doing like this till the end of the stack

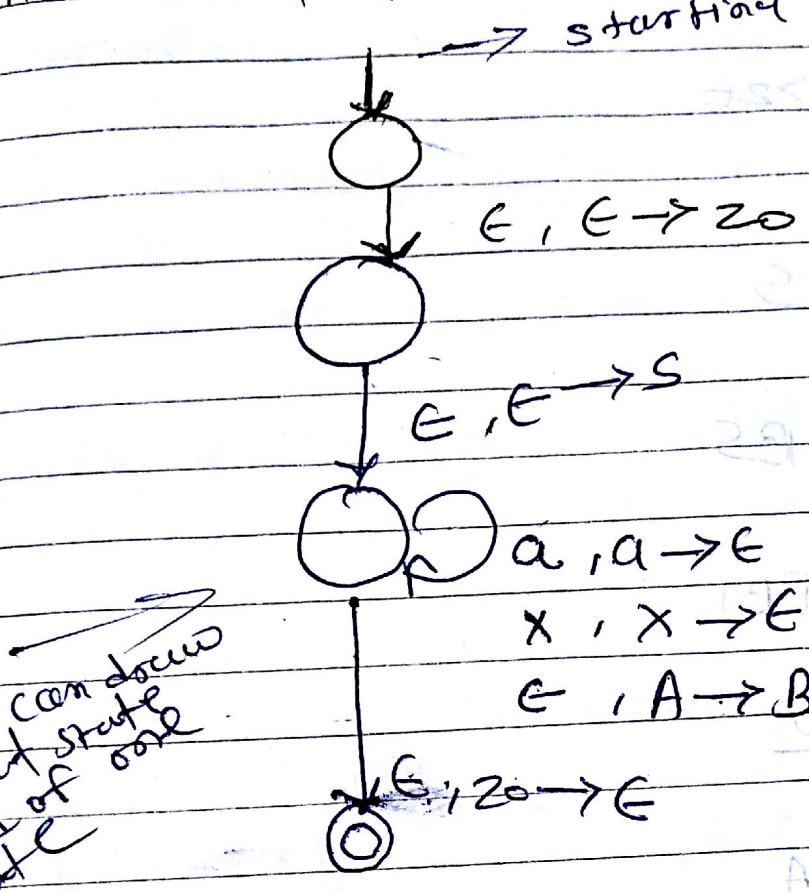


$0,0 \rightarrow \epsilon$
$1,1 \rightarrow \epsilon$
$2,2 \rightarrow \epsilon$
$X,X \rightarrow \epsilon$ for all $X \in \Sigma$

ie for all terminal Symbol pop the element don't push.

Now Let's Design final PDA for CFG

The final PDA



At the end if we reach to end of Stack & we come to final state then we can say

PDA = CFG.

Hence we can prove part 1 Theorem

Part I. Convert CFG to PDA

CFG contains variables & Terminals

Rule 1 - for each variable A

PDA for m transition → $\delta(q, \epsilon, A) = (q, \beta)$ where $A \to \beta$ is production of Grammar

Rule 2

for each Terminal 'a'

$$\delta(q, a, a) = (q, \epsilon)$$

- we know that PDA has transition & CFG has productions
- so CFG to PDA means production converted to PDA

Now CFG has variable & Terminal

Example

$S \to 0BB$

$B \to 0S \mid 1S \mid 0$

Test $w = 010^4 = w = 010000$

Check above string is desived by PDA or not.?

The equivalent PDA for the given grammar

→ Here, S & B are variable

$\delta(q, \epsilon, S) = (q, 0BB)$ ---- (1)

$\delta(q, \epsilon, B) = (q, 0S), (q, 1S), (q, 0)$ ---- (2)

$\delta(q, 0, 0) = (q, \epsilon)$ ---- (3)

$\delta(q, 1, 1) = (q, \epsilon)$ ---- (4)

Number the transitions

Start with start symbol

$\delta(q, 010000, S)$ using Transition (1) where we get

$\delta(q, 010000, 0BB)$

i/p ↑ top ↑

using (3) ~~operation~~ transition

-Now pop operation

pop '0' from i/p & top

$\delta(q, 10000, BB)$

Now i/p is 1 & B can substitute $B \to 1S$ so using (2) transition

$\delta(q, 10000, 1SB)$ ---- Now use (2) trans

1 from i/p & top will poped

$\delta(q, 0000, SB)$ --- Now using (1) transn

$\delta(q, 0000, 0BB.B)$ i/p '0' top '0' pop operation (3) transn.

$\delta(q, 000, BBB)$ ---- Now using (2) transn substitute (q, 0) value

(q, 0, 0)
state ↑ i/p ↳ stack top

δ(q, 000, 0BB) ---- (3) transf
i/p '0' top = '0' pop it

δ(q, 00, BB) ---- (2) Transf put (q, 0)

δ(q, 00, 0B) ---- i/p=0, Top='0' so pop using (3) Transf

δ(q, 0, B) ---- (2) Transf B = (q, 0)

δ(q, 0, 0) ---- (3) Transf pop opn

(q, ∈, ∈) ---- String Accept

Example 2
Grammar
S → 0S1 | 00 | 11

The equivalent PDA for the given grammar

Variable Rule 1
δ(q, ε, S) = (q, 0S1), (q, 00), (q, 11) --- (1)

Terminal Rule 2
δ(q, 0, 0) = (q, ∈) pop --- (2)
δ(q, 1, 1) = (q, ∈) pop --- (3)

String to be tested whether PDA accept it or not w = 0111

...leftmost derivation of 1001.

$\delta(q, 0111, S)$ ----- (1)
$\delta(q, 0111, 0S1)$ --- (2)
$\delta(q, 111, S1)$ ---- (1)
$\delta(q, 111, 111)$ ... (3)
$\delta(q, 11, 11)$ -- (3)
$\delta(q, 1, 1)$ -- (3)
$\delta(q, \epsilon, \epsilon)$ ---- Accepted

Example 3:

$S \rightarrow AB$
$A \rightarrow 0A \mid \epsilon$
$B \rightarrow 0B \mid 1B \mid \epsilon$

for the string $w = 0010$ check if it is accepted by PDA or not

Teeminologies Related to PDA

Instantaneous Desciption

The instantaneous description (ID) of a PDA is represented by a triplet (q, w, s) where

q – is the state
w – is unconsumed input
s – is the stack contents

Turnstile Notation:

The "turnstile" notation is used for connecting pairs of ID's that represent one or many moves of a PDA. The process of transition is denoted by the turnstile symbol "⊢".

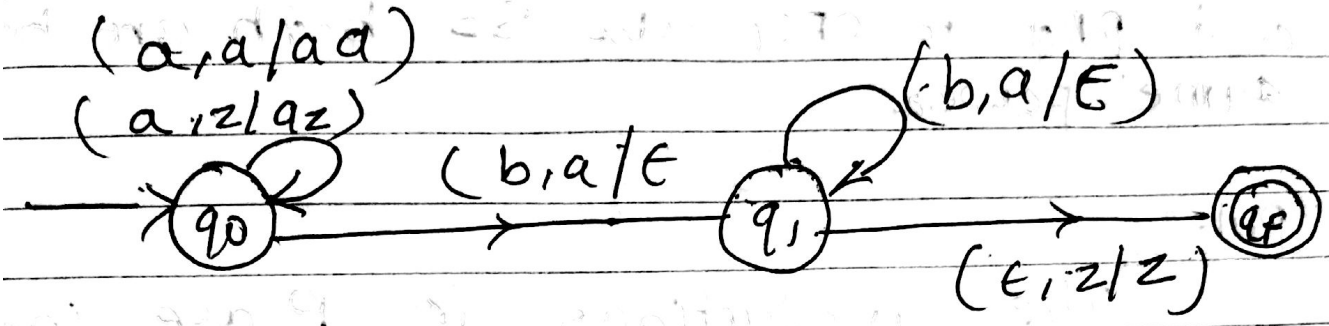
Consider PDA $(Q, \Sigma, S, \delta, q_0, I, F)$
A transition can be mathematically represented by the following turnstile notation –

$$(p, aw, T\beta) \vdash (q, w, ab)$$

This implies that while taking a transition from state p to state q, the input symbol 'a' is consumed, and the top of the stack 'T' is replaced by a new string 'a'

Define pushdown Automata for language $(a^n b^n \mid n>0)$

aabb or aaabbb



Transition function

$\delta(q_0, a, z) = (q_0, az)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z) = (q_f, z)$

when we write $\delta(q_1, \epsilon, z) = \delta(q_f, z)$ means string is accepted by final state

and when we write $\delta(q_1, \epsilon, z) = \delta(q_1, \epsilon)$ means string is accepted by empty stack.

Both are correct.

Conversion of PDA to CFG

CFG $\cong$ PDA

CFG and PDA are equivalent to each other because we can convert CFG to PDA and PDA to CFG also. So both are having same power.

Rules

The productions in P are induced by move of PDA as follows-

Rule(1) S productions are given by

$S \rightarrow [q_0 \; z_0 \; q]$ for every $q \in Q$

means every state $q \in Q$

Rule(2) for every popping Move

$\delta(q, a, z) = (q', \epsilon)$ induces production

$[q \; z \; q'] \rightarrow a$

Here a get poped

So it is written like $[q \; z \; q'] \rightarrow a$

Rule(3) for each push Move

$\delta(q, a, z) = (q_1, z_1, z_2 \cdots z_n)$ induces many production

$[q \; z \; q] \rightarrow a [q_1 \; z_1 \; q_2] [q_2 \; z_2 \; q_3] \cdots [q_m \; z_m \; q]$

where each state $q, q_1, q_2 \cdots q_m$ can be any state in

Ex: Generate CFG for given PDA M is defined as (set of state, i/p symbol, stack value)

$$M = \{\{q_0, q_1\}, \{0,1\}, \{x, z_0\}, \delta, q_0, z_0, q_1\}$$

where δ given as follows (transition)

1) $\delta(q_0, 1, z_0) = (q_0, xz_0)$
2) $\delta(q_0, 1, x) = (q_0, xx)$
3) $\delta(q_0, 0, x) = (q_0, x)$
4) $\delta(q_0, \epsilon, x) = (q_1, \epsilon)$
5) $\delta(q_1, \epsilon, x) = (q_1, \epsilon)$
6) $\delta(q_1, 0, x) = (q_1, xx)$
7) $\delta(q_1, 0, z_0) = (q_1, \epsilon)$

for each transition function we have to follow the rules or steps to see the transition is poping move or pushing move accordingly follow the steps

First is S production

Rule 1: we have to produce S production

$S \rightarrow [q_0\ z_0\ q_0]$

$/\ [q_0\ z_0\ q_1]$

Here q i.e set of states are $\{q_0, q_1\}$ so two productions written for each state

$A_2 = [q_0\ z_0\ q_0]$ is single state

$B_2 = [q_0\ z_0\ q_1]$ --- so $S \rightarrow A_2 | B_2$

1) for 1st Transition

$$\delta(q_0, 1, z_0) = (q_0, xz_0)$$

This is a push move so use Rule 3

Rules $= \delta(q, a, z) = (q_1, z_1, z_2 \ldots z_n)$

Rule 3

$[q\ z\ q'] \rightarrow a[q_1\ z_1\ q_2][q_2\ z_2\ q_3]$

q' is the any state in Q

$\delta(q_0, 1, z_0) = (q_0, xz_0)$

2 values in stack & 2 states so we have $2 \times 2 = 4$ production

$[q_0\ z_0\ q_0] \rightarrow 1[q_0\ x\ q_0][q_0\ z_0\ q_0]$

$[q_0\ z_0\ q_0] \rightarrow 1[q_0\ x\ q_1][q_1\ z_0\ q_0]$

$[q_0\ z_0\ q_1] \rightarrow 1[q_0\ x\ q_0][q_0\ z_0\ q_1]$

$[q_0\ z_0\ q_1] \rightarrow 1[q_0\ x\ q_1][q_1\ z_0\ q_1]$

2) for the 2nd Transition

$\delta(q_0, 1, x) = (q_0, xx)$ (Apply Rule 3 coz push move is)

$[q_0\ x\ q_0] \rightarrow 1[q_0\ x\ q_0][q_0\ x\ q_0]$

$[q_0\ x\ q_0] \rightarrow 1[q_0\ x\ q_1][q_1\ x\ q_0]$

$[q_0\ x\ q_1] \rightarrow 1[q_0\ x\ q_0][q_0\ x\ q_1]$

$[q_0\ x\ q_1] \rightarrow 1[q_0\ x\ q_1][q_1\ x\ q_1]$

for 3rd Transition – push move

$\delta(q_0, 0, x) = (q_0, x)$

$[q_0\ x\ q_0] \to 0[q_0\ x\ q_0]$

$=$

$[q_0\ x\ q_1] \to 0[q_0\ x\ q_1]$

4) for 4th Transition

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

$=$ $[q_0\ x\ q_1] \to \epsilon$

5] for the 5th Transition Apply Rule 2

$\delta(q_1, \epsilon, x) = (q_1, \epsilon)$

$=$ $[q_1\ x\ q_1] \to \epsilon$

6) for 6th Transition

$\delta(q_1, 0, x) = (q_1\ xx)$

$[q_1\ x\ q_0] \to 0[q_1\ x\ q_0][q_0\ x\ q_0]$
$[q_1\ x\ q_0] \to 0[q_1\ x\ q_1][q_1\ x\ q_0]$
$=$ $[q_1\ x\ q_1] \to 0[q_1\ x\ q_0][q_0\ x\ q_1]$
$[q_1\ x\ q_1] \to 0[q_1\ x\ q_1][q_1\ x\ q_1]$

7) for 7th Transition Apply Rule 2

$\delta(q_1, 0, z_0) = (q_1, \epsilon)$

$= [q_1 \; z_0 \; q_1] \rightarrow 0$

After getting CFG we have to determine some productions are repeated or useless production

we can reduce this useless or epsilon productions by elimination.

part 2: PDA to CFG (conversion

$M = \{\{p, q\}, \{0, 1\}, \{X, Z\}, \delta, q, Z\}$

state = Q, Σ, Γ, δ, q_0, Z_0

① $\delta(q, 1, Z) \Rightarrow (q, XZ)$ ... push

② $\delta(q, 1, X) \Rightarrow (q, XX)$ ... push

③ $\delta(q, \epsilon, X) \Rightarrow (q, \epsilon)$ ... pop

4) $\delta(q, 0, X) \Rightarrow (P, X)$ No operation

5) $\delta(P, 1, X) \Rightarrow (P, \epsilon)$ ... pop

6) $\delta(P, 0, Z) \Rightarrow (q, Z)$ ... No operation

Solution:

Initial state ... stack (initial symbol)

$S \rightarrow [q, Z, q]$
$S \rightarrow [q, Z, P]$
} both possible start symbol productions

Transition's to be converted in production's

① $\delta(q, 1, Z) \Rightarrow (q, XZ)$ push

$[q, Z, q] \rightarrow 1\,[q, X, q]\,[q, Z, q]$
$[q, Z, q] \rightarrow 1\,[q, X, P]\,[P, Z, q]$
$[q, Z, P] \rightarrow 1\,[q, X, q]\,[q, Z, P]$
$[q, Z, P] \rightarrow 1\,[q, X, P]\,[P, Z, P]$

② $\delta(q, 1, x) \Rightarrow (q, xx)$ --- push

$[q, x\ q] \Rightarrow 1[q, x, q][q, x, q]$

$[q, x\ q] \rightarrow 1[q, x, p][p, x, q]$

$[q, x\ p] \rightarrow 1[q, x, q][q, x, p]$

$[q, x\ p] \rightarrow 1[q, x, p][p, x, p]$

③ $\delta(q, \varepsilon, x) \Rightarrow (q, \varepsilon)$ --- pop

$[q, x, q] \rightarrow \varepsilon$

④ $\delta(q, 0, x) \Rightarrow (p, x)$ No operation

$[q, x, q] \rightarrow 0[p, x, q]$

$[q, x, p] \rightarrow 0[p, x, p]$

Both p & q states to be covered

⑤ $\delta(p, 1, x) \rightarrow (p, \epsilon)$ pop

$[p, x, p] \rightarrow 1$

⑥ $\delta(p, 0, z) \rightarrow (q, z)$ No operation

$[p, z, q] \rightarrow 0[q, z, q]$

$[p, z, p] \quad 0[q, z, p]$

Original Name	New Name	Original Name	New Name
[q, z, q]	A	[q, x, q]	E
[q, z, p]	B	[q, x, p]	F
[p, z, q]	C	[p, x, q]	G
[p, z, p]	D	[p, x, p]	H

original

$S \rightarrow [q, z, q]$

$S \rightarrow [q, z, p]$

By above Table

$S \rightarrow A$

$S \rightarrow B$

① for first Transition

$A \rightarrow 1EA$

$A \rightarrow 1FC$

$B \rightarrow 1EB$

$B \rightarrow 1FD$

② for 2nd Transition

$E \rightarrow 1EE$

$E \rightarrow 1FG$

$F \rightarrow 1EF$

$F \rightarrow 1FH$

③ for 3rd Transition

$E \rightarrow \epsilon$

④ 4th Transition

$E \rightarrow 0G$

$F \rightarrow 0H$

⑤ For 5th Transition

$H \rightarrow 1$

6th Transition

$C \rightarrow 0A$

$D \rightarrow 0B$

Grammar:

$S \rightarrow A \mid B$

$A \rightarrow 1EA \mid 1FC$

$B \rightarrow 1EB \mid 1FD$

$E \rightarrow 1EE \mid 1FG \mid \epsilon \mid 0G$

$F \rightarrow 1EF \mid 1FH \mid 0H$

$H \rightarrow 1$

$C \rightarrow 0A$

$D \rightarrow 0B$