UNIT-3

Pushdown Automata: Definitions Moves
Instantaneous descriptions
Deterministic pushdown automata-Problems related to DPDA
Non - Deterministic pushdown automata-Problems related to NDPDA
Pushdown automata to CFL Equivalence-Problems of PDA to CFG
CFL to Pushdown automata Equivalence
Problems related to Equivalence of CFG

Pushdown Automata: Definitions Moves

- Pushdown Automata
    - Definition
    - Moves of Pushdown Automata
    - Acceptance of Pushdown Automata
    - Instantaneous Description
    - Types
        - Deterministic Pushdown Automata
        - Nondeterministic Pushdown Automata

Automata

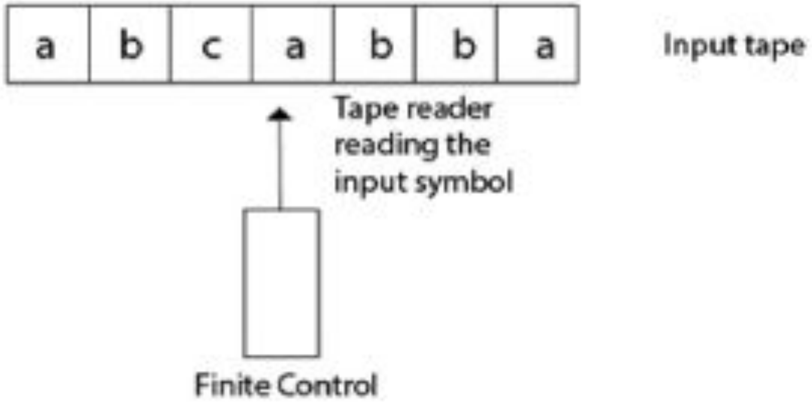
Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	Context-free grammar	Context-free language	Pushdown automaton
Type 3	Regular grammar	Regular language	Finite state automaton

Finite Automata

An automaton with a finite number of states is called a Finite Automaton (FA) or Finite State Machine (FSM). Formal definition of a Finite Automaton An automaton can be represented by a 5-tuple (Q, Σ, δ, q0, F), where:

1. Q: finite set of states
2. ∑: finite set of the input symbol
3. q0: initial state
4. F: **final** state
5. δ: Transition function

$$\delta: Q \times \Sigma \rightarrow Q.$$



NFA Vs PDA: Definitions

NFA	PDA
The language accepted by NFA is the regular language	The language accepted by PDA is Context free language.
NFA has no memory.	PDA is essentially an NFA with a stack (memory).
It can store only limited amount of information.	It stores unbounded limit of information.
A language/string is accepted only by reaching the final state.	It accepts a language either by empty Stack or by reaching a final state.

Pushdown Automata: Definitions

A PDA is a computational machine to recognize a Context free language. Computational power of PDA is between Finite automaton and Turing machines. The PDA has a finite control , and the memory is organized as a stack.

Pushdown Automata: Definitions Conti….

A pushdown automaton consists of seven tuples

$$P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

Where,

Q - A finite non empty set of states

Σ - A finite set of input symbols.

Γ - A finite non empty set of stack symbols.

q_0 - q_0 in Q is the start state

z_0 - Initial start symbol of the stack.

F - $F \subseteq Q$, set of accepting states or final states

δ - Transition function $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$

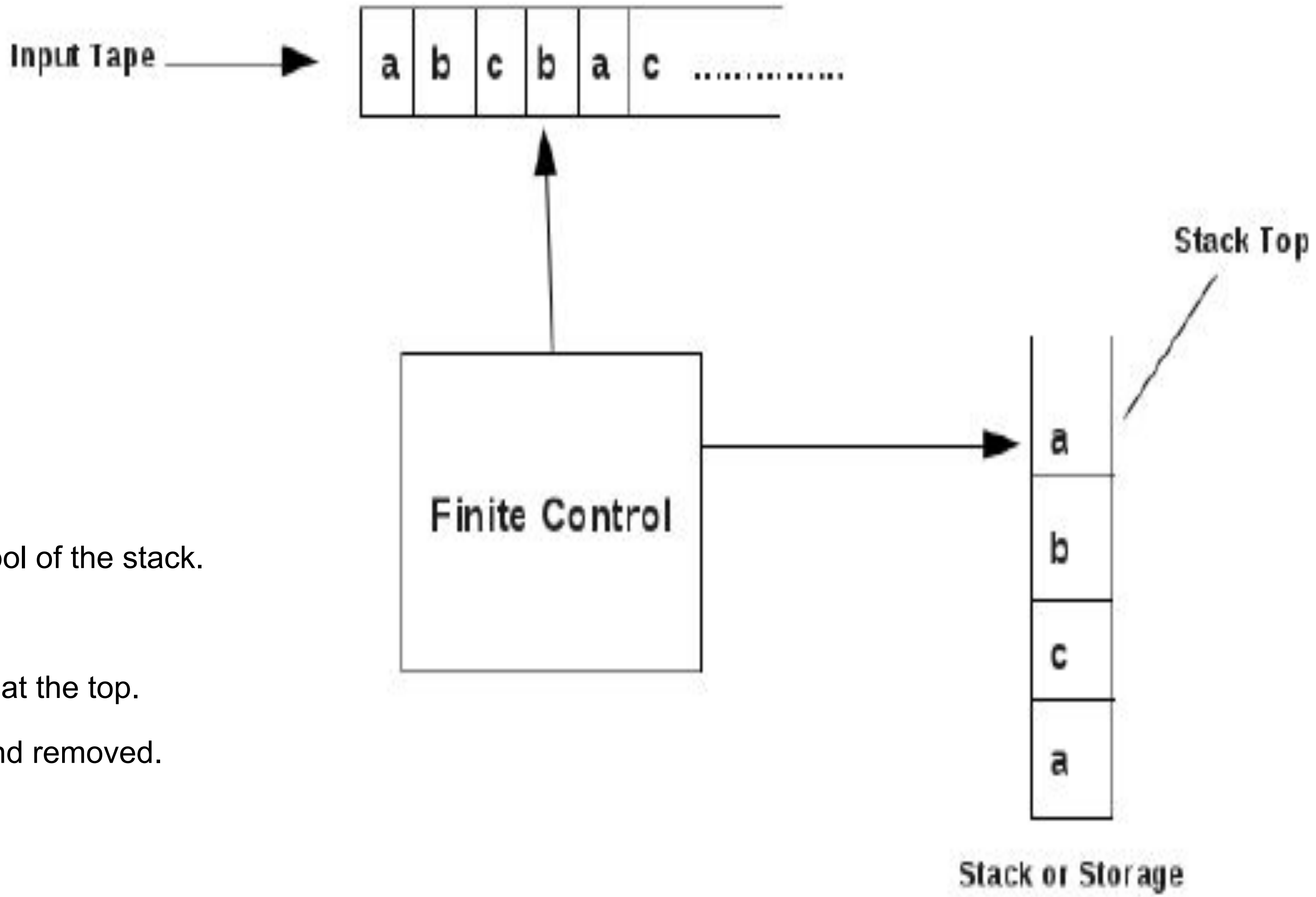
In PDA, the transitional function δ is in the form

$$Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow (Q, \Gamma)$$

δ is a transition function which maps,

$$(Q \times \Sigma^* \times \Gamma^*) \longrightarrow (Q \times \Gamma^*)$$

Basic Structure of PDA



A PDA has three components −

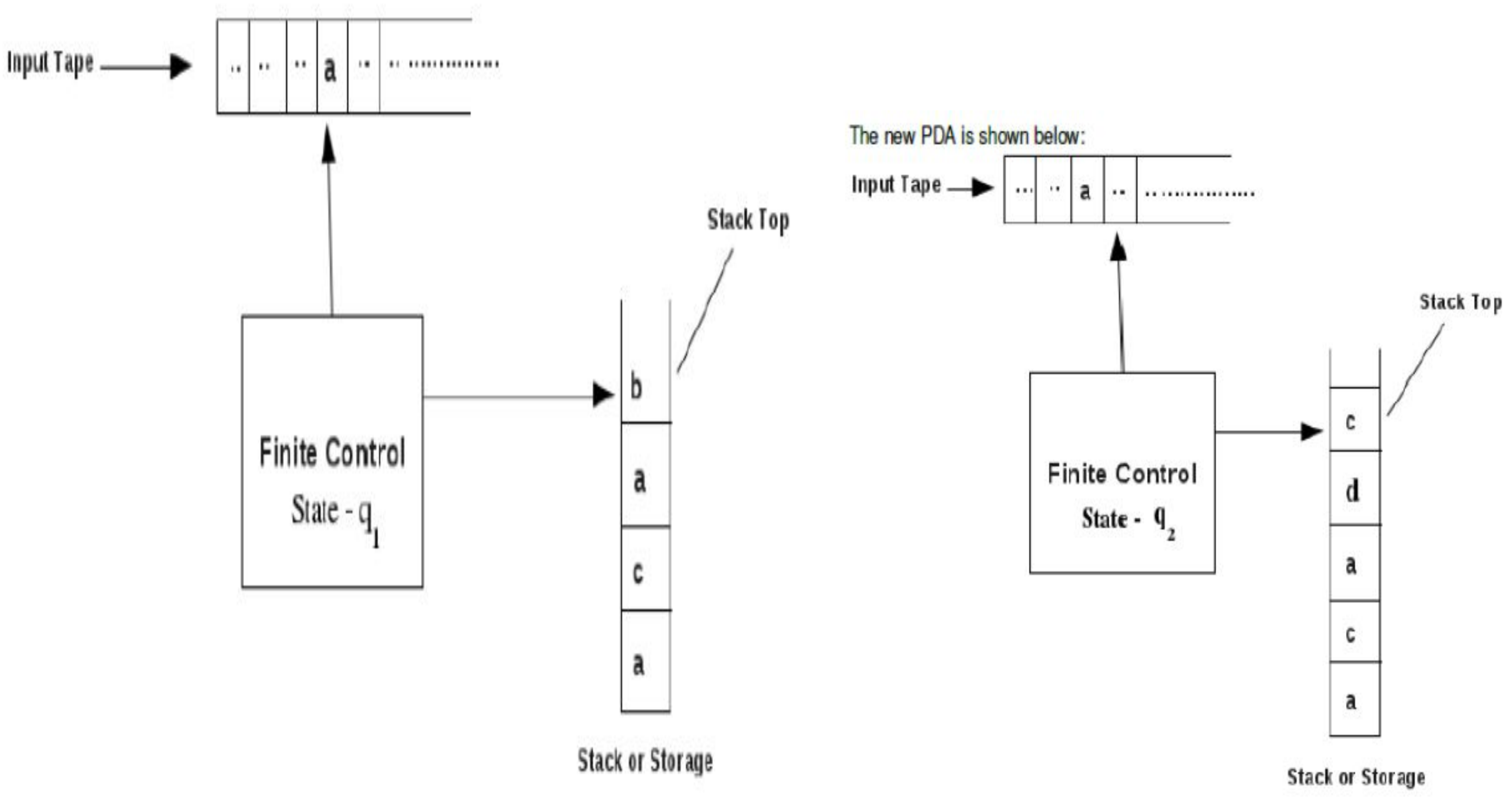
- an input tape,
- a control unit, and
- a stack with infinite size.

The stack head scans the top symbol of the stack.

A stack does two operations −

- **Push** − a new symbol is added at the top.
- **Pop** − the top symbol is read and removed.

PDA-Transitions



Representation of State Transition

Delta Function (δ) is the transition function, the use of which will become more clear by taking a closer look at the Three Major operations done on Stack :-

1. Push
2. Pop
3. Skip /No operation

Language Acceptability by PDA

- The input string is accepted by the PDA if:

  - The final state is reached .
  - The stack is empty.

- For a PDA $M=(Q, \Sigma, \Gamma, \delta, q0, Z0, F)$ we define : Language accepted by final state **L(M)** as:

$\{ w \mid (q0, w, Z0) \vdash^{*} (p, \epsilon, \gamma)$ for some p in F and γ in Γ^{*} $\}$.

- Language accepted by empty / null stack **N(M)** is:

$\{ w \mid (q0, w, Z0) \vdash^{*} (p, \epsilon, \epsilon)$ for some p in Q$\}$.

ways of language acceptances by a PDA

PDA accepts its input either by "acceptance by final state" or "Acceptance by Empty stack"

PDA acceptance by empty stack method	**PDA acceptance by empty final method**
PDA accepts when set of strings that cause the PDA to empty its stack	PDA accepts its input by consuming it and entering an accepting state
For each PDA $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ then $L(P) = \{w \mid (q_0,w,z_0) \quad (q,\in,\alpha)\}$	For each PDA $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ then $N(P)$ $= \{w \mid (q_0,w,z_0) \quad (q,\in,\in)\}$

Ex:1Language Acceptability by PDA

Language Acceptability by PDA

Example 1:

Consider a PDA,

$P = (Q, \sum, \Gamma, \delta, q_0, F)$

where

$Q = \{s, q, f\}$

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

$\Gamma = \{a, b, c\}$

$q_0 = \{s\}$

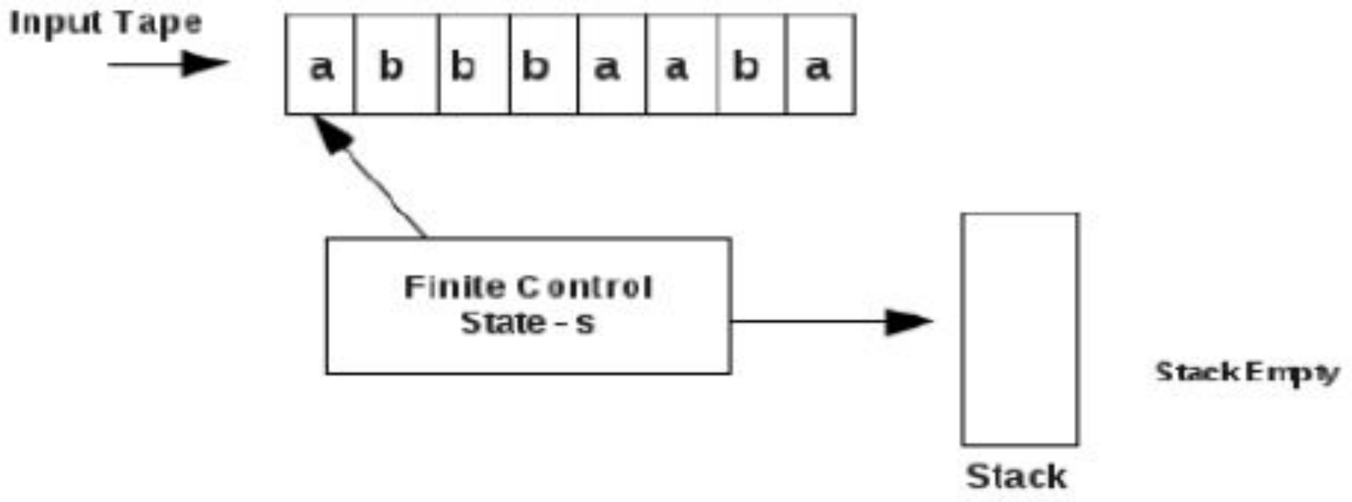
$F = \{f\}$

δ is given as follows:

1. $(s, a, \varepsilon) \longrightarrow (q, a)$
2. $(s, b, \varepsilon) \longrightarrow (q, b)$
3. $(q, a, a) \longrightarrow (q, aa)$
4. $(q, b, b) \longrightarrow (q, bb)$
5. $(q, a, b) \longrightarrow (q, \varepsilon)$
6. $(q, b, a) \longrightarrow (q, \varepsilon)$
7. $(q, b, \varepsilon) \longrightarrow (q, b)$
8. $(q, \varepsilon, \varepsilon) \longrightarrow (f, \varepsilon)$

Check whether the string abbbaaba is accepted by the above pushdown automation.

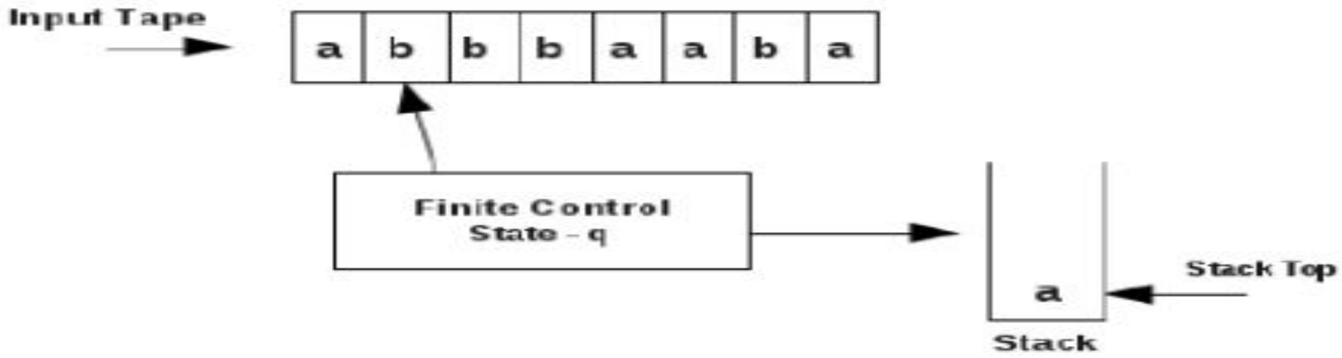
Ex:1Language Acceptability by PDA



1.

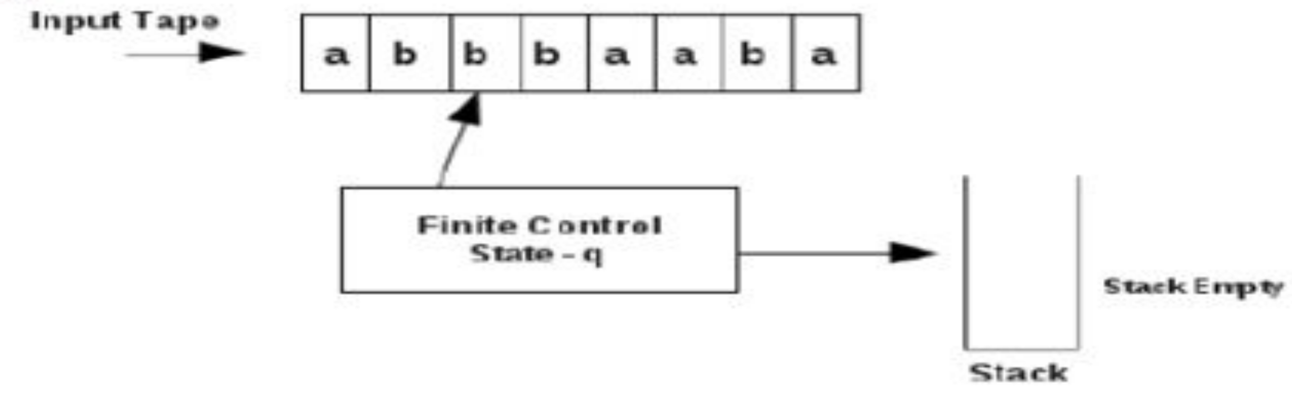
The PDA is in state s, stack top contains symbol ε. Consider the transition,

1. $(s, a, \varepsilon) \longrightarrow (q, a)$

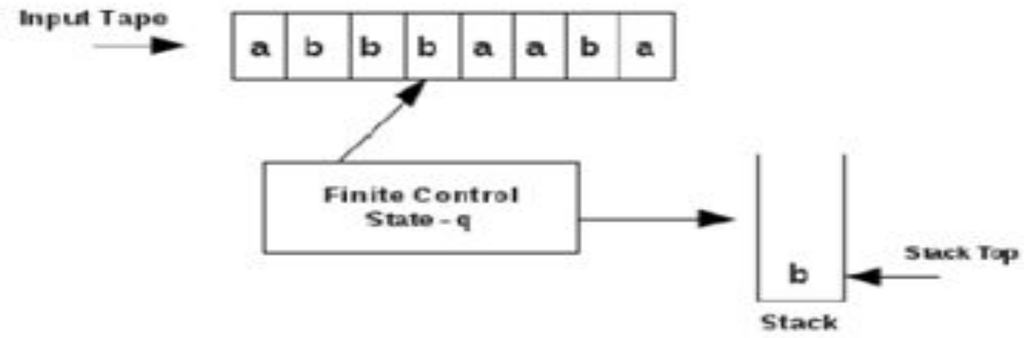


Conti….

2.

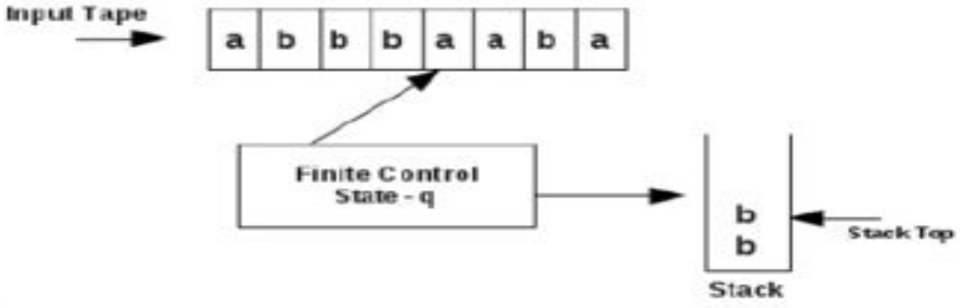


3. PDA now is,

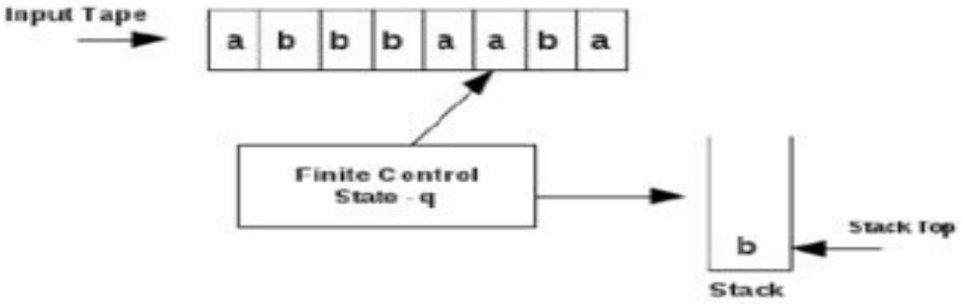


Conti….

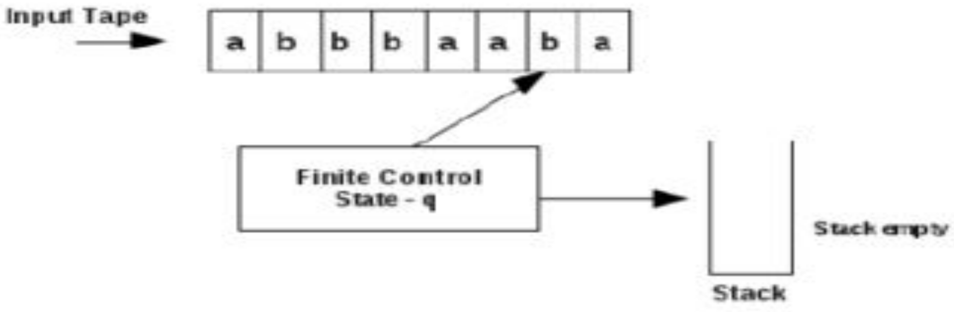
4.



5. PDA now is,



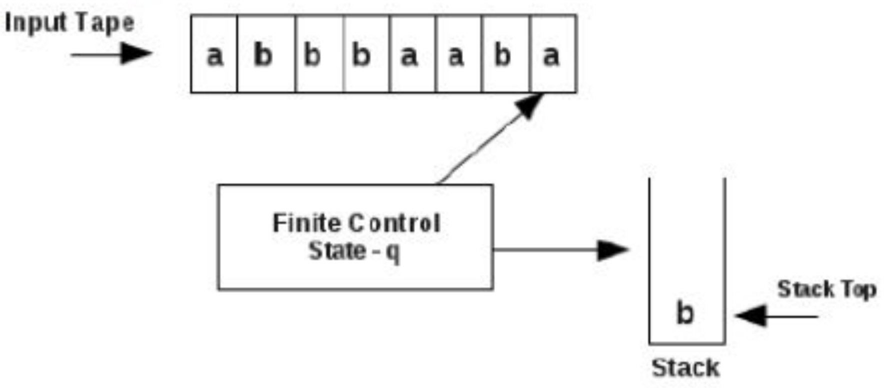
6.



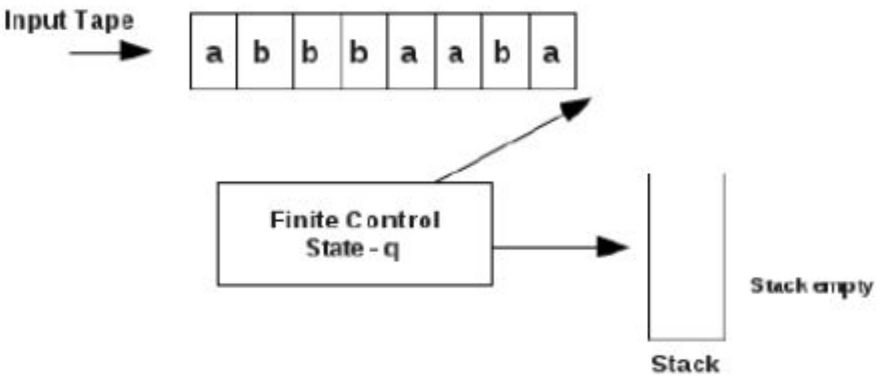
Conti....

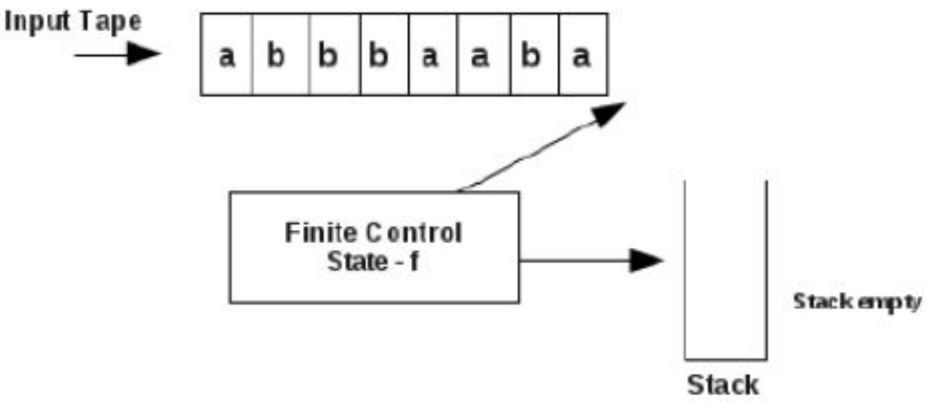
PDA now is,

7.



8. PDA now is,





So the string *abbbaaba* is accepted by the above pushdown automation.

Example :2

Example 2:

1.

Consider a PDA,

$$P = (Q, \sum, \Gamma, \delta, q_0, F)$$

where

$$Q = \{s, f\}$$

$$\sum = \{a, b, c\}$$

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

$$q_0 = \{s\}$$

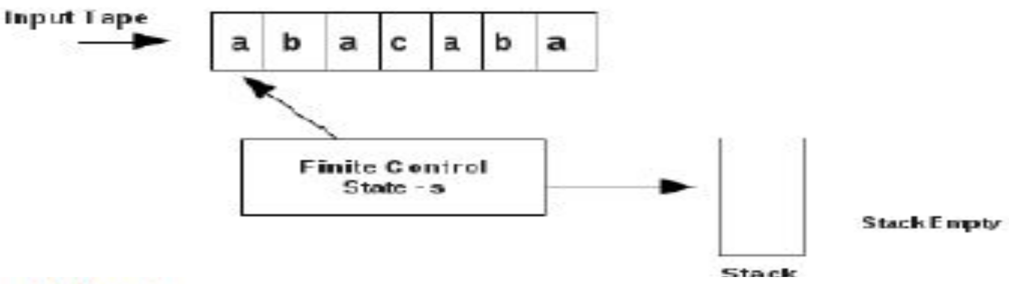
$$F = \{f\}$$

δ is given as follpws:

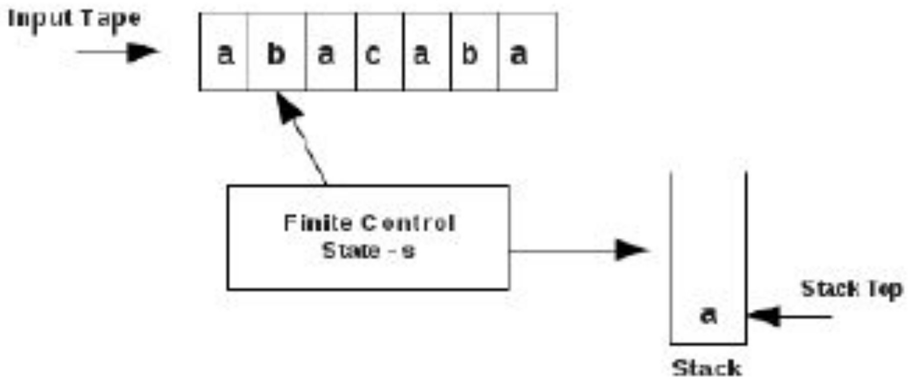
1. $(s, a, \varepsilon) \longrightarrow (s, a)$
2. $(s, b, \varepsilon) \longrightarrow (s, b)$
3. $(s, c, \varepsilon) \longrightarrow (f, \varepsilon)$
4. $(f, a, a) \longrightarrow (f, \varepsilon)$
5. $(f, b, b) \longrightarrow (f, \varepsilon)$

Check whether the string $abacaba$ is accpeted by the above pushdown automation.

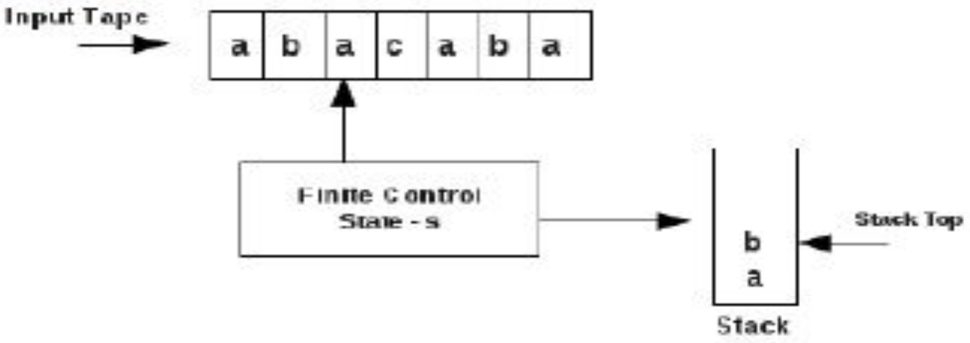
Ex:2Language Acceptability by PDA



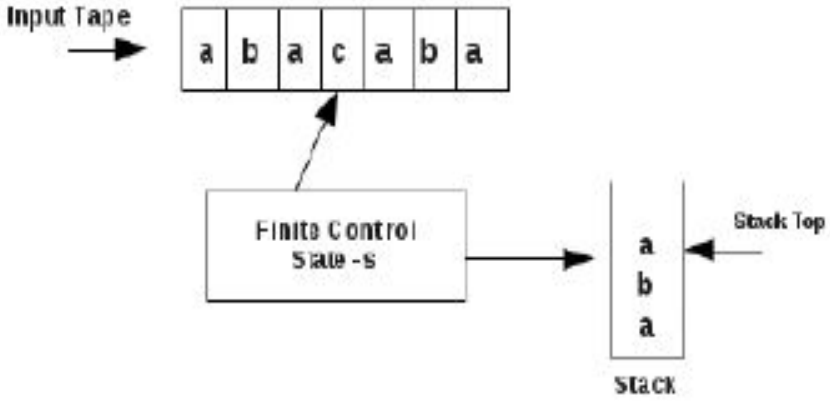
1. PDA now is,



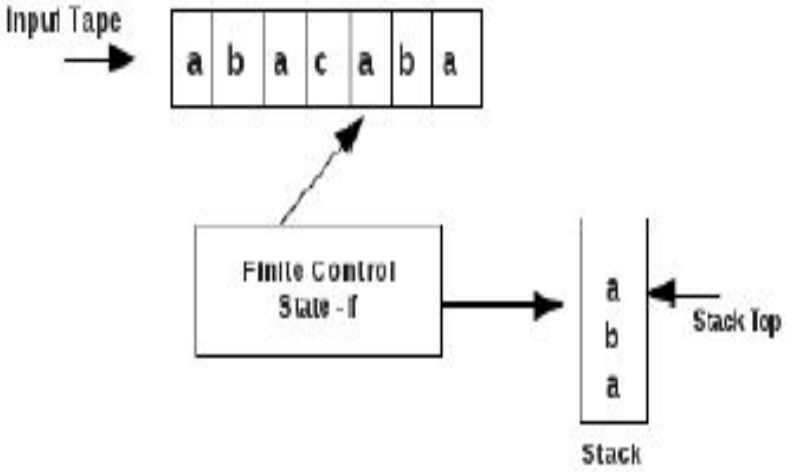
2. PDA now is,



3. PDA now is,

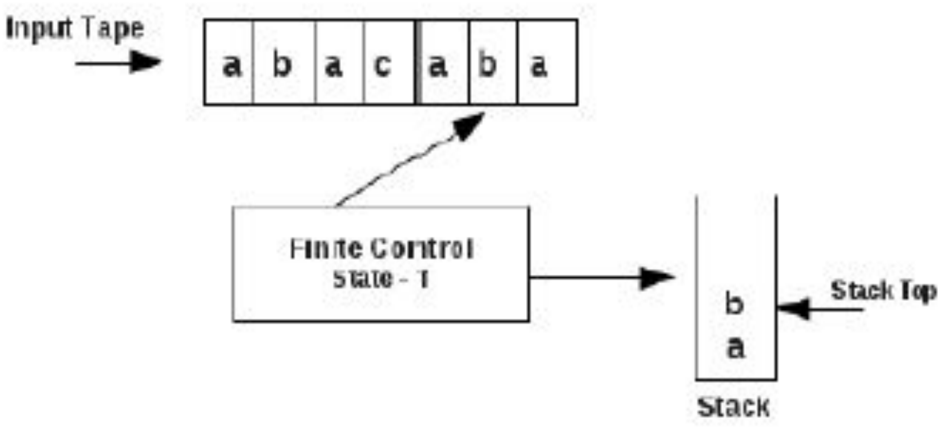


4. PDA now is,

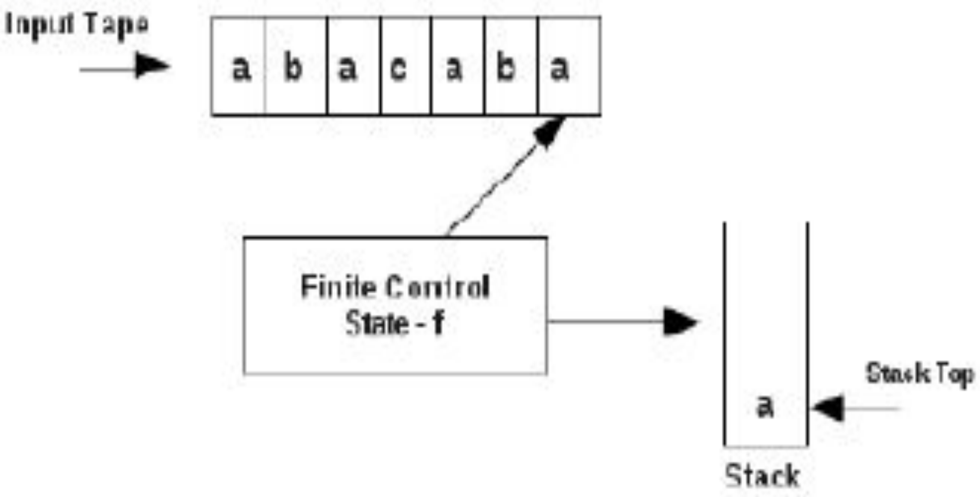


Conti….

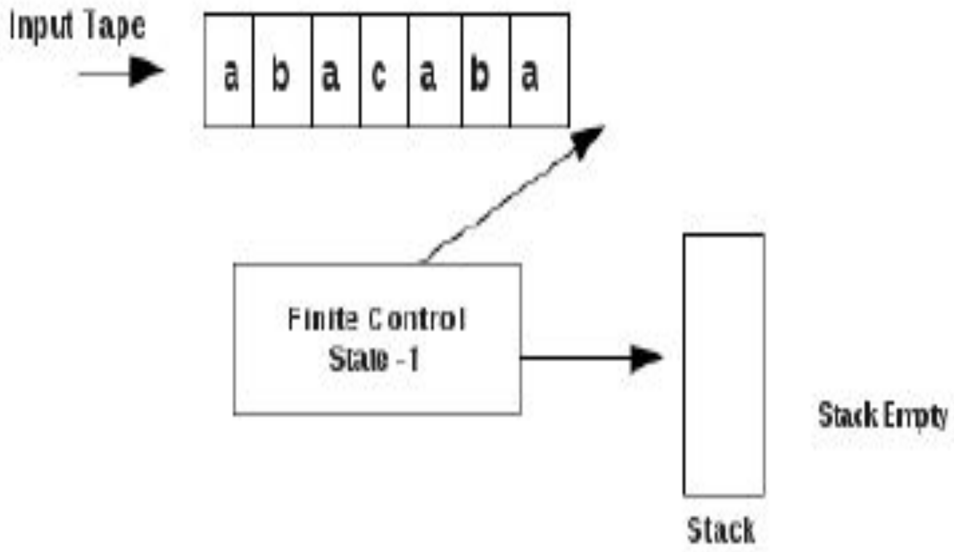
5. PDA now is,



6. PDA now is,



7.



Now there are no more symbols in the input string, stack is empty. PDA is in final state, f.

So the string *abacaba* is accepted by the above pushdown automation.

Assignment Exercises:

Language Acceptability by PDA

Example 1:

Consider a PDA,

$$P = (Q, \sum, \Gamma, \delta, q_0, F)$$

where

$Q = \{s, q, f\}$

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

$\Gamma = \{a, b, c\}$

$q_0 = \{s\}$

$F = \{f\}$

δ is given as follows:

1. $(s, a, \varepsilon) \longrightarrow (q, a)$
2. $(s, b, \varepsilon) \longrightarrow (q, b)$
3. $(q, a, a) \longrightarrow (q, aa)$
4. $(q, b, b) \longrightarrow (q, bb)$
5. $(q, a, b) \longrightarrow (q, \varepsilon)$
6. $(q, b, a) \longrightarrow (q, \varepsilon)$
7. $(q, b, \varepsilon) \longrightarrow (q, b)$
8. $(q, \varepsilon, \varepsilon) \longrightarrow (f, \varepsilon)$

Check whether the string abbbaaba is accepted by the above pushdown automation.

Instantaneous Description

- ID describe the configuration of a PDA at a given instant.ID is a triple such as
- (q, w ,γ) , where q is a state , w is a string of input symbols and is a string of stack symbols.

- Instantaneous Description (ID) is an informal notation of how a PDA "computes" a input string and make a decision that string is accepted or rejected.
- The relevant factors of pushdown configuration notation by a triple (q, w, γ) where;

- q is the current state of the control unit
- w is the unread part of the input string or the remaining input alphabets
- γ is the current contents of the PDA stack.
- Conventionally, we show leftmost symbol indicating the top of the stack γ and the bottom at the right end. Such a triple notation is called an instantaneous description or ID of the pushdown automata.

- It is also useful to represent as part of the configuration the portion of the input that remains.

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 a PDA (Q, ∑, S, δ, q_0, I, F).

A transition can be mathematically represented by the following turnstile notation −(p, aw, Tβ) ⊢ (q, w, αb) .

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 **'α'**.

Note − If we want zero or more moves of a PDA, we have to use the symbol (⊢*) for it.

Conti..

- Write down the IDs or moves for input string w = "aabb" of PDA as M = ($\{q_0, q_1, q_2\}$, {a, b}, $\{a, b, Z_0\}$, δ, q0, Z_0, $\{q_2\}$), where δ is defined by following rules:

- $\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$ Rule (1)
  $\delta(q_0, a, a) = \{(q_0, aa)\}$ Rule (2)
  $\delta(q_0, b, a) = \{(q_1, \lambda)\}$ Rule (3)
  $\delta(q_1, b, a) = \{(q_1, \lambda)\}$ Rule (4)
  $\delta(q_1, \lambda, Z_0) = \{(q_2, \lambda)\}$ Rule (5)
  $\delta(q_0, \lambda, Z_0) = \{(q_2, \lambda)\}$ Rule (6)

Conti..

- check string w is accepted by PDA or not?
- **Solution:** Instantaneous Description for string w = "aabb"

- (q_0, aabb, Z_0) |- (q_0, abb, aZ_0) as per Rule (1)
  |- (q_0, bb, aaZ_0) as per Rule (2)
  |- (q_1, b, aZ_0) as per Rule (3)
  |- (q_1, λ, Z_0) as per Rule (3)
  |- $(q2, \lambda, \lambda)$ as per Rule (5)

- Finally PDA reached a configuration of (q_2, λ, λ) i.e. the input tape is empty or input string w is completed, PDA stack is empty and PDA has reached a final state. So the string 'w' is **accepted**.

Write down the IDs or moves for input string w = “aaabb” of PDA.
Also check it is accepted by PDA or not?

- **Solution:** Instantaneous Description for string w = “aaabb”

  (q_0, aaabb, Z_0) |- (q_0, aabb, aZ_0)   as per transition Rule (1)
  |- (q_0, abb, aaZ_0)   as per transition Rule (2)
  |- (q_0, bb, $aaaZ_0$)   as per transition Rule (2)
  |- (q_1, b, aaZ_0)   as per transition Rule (3)
  |- (q_1, λ, aZ_0)   as per transition Rule (3)
  |- There is no defined move.

- So the pushdown automaton stops at this move and the string is not accepted because the input tape is empty or input string w is completed but the PDA stack is not empty. So the string ‘w’ is **not accepted**.

Moves of PDA And Types

- *Moves of a Pushdown Automata*: A PDA chooses its next move based on its current state, the next input symbol, and the symbol at the top of its stack. It may also choose to make a move independent of the input symbol and without consuming that symbol from the input. Being nondeterministic, the PDA may have some finite number of choices of move; each is a new state and a string of stack symbols with which to replace the symbol currently on top of the stack.

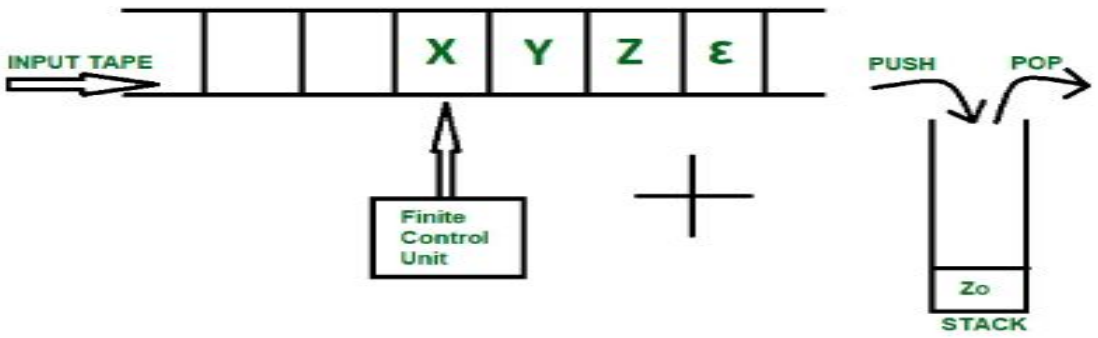
Deterministic PDA :-

$\delta : Q \times \Sigma \times \Gamma = Q \times \Gamma^*$

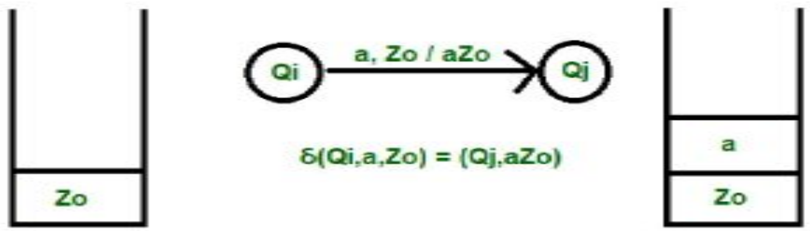
Non-Deterministic PDA :-

$\delta : Q \times \Sigma \times \Gamma = 2^\wedge(Q \times \Gamma^*)$

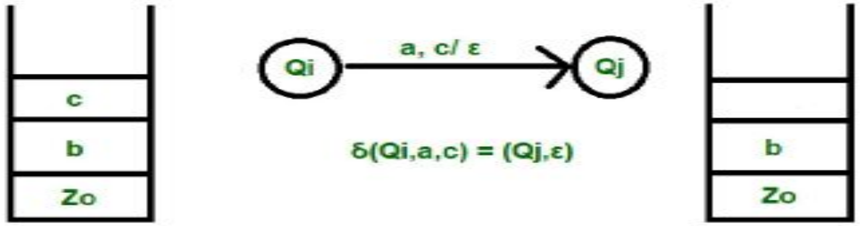
Moves of PDA



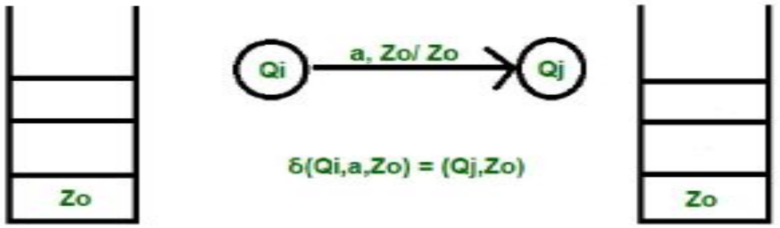
Push



Pop



Skip/ no operation



<u>Moves</u>: The interpretation of

$$\delta(q, a, Z) = \{(P_1, \gamma_1), (P_2, \gamma_2) \ldots (P_M, \gamma_M)\}$$

where q, P_i – states a – input symbol Z – stack symbol
γ_i – a symbol in Γ^*

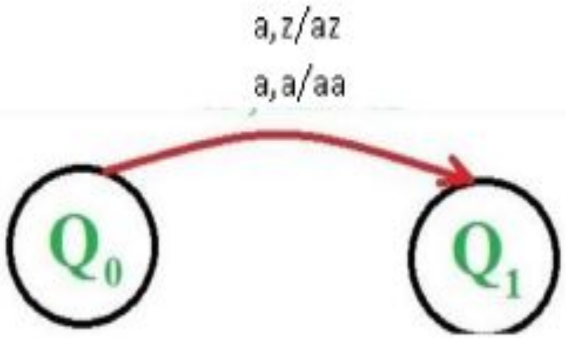
PDA enters state P_i, replaces the symbol Z by the string γ_i and advances the input head one symbol.

Representation of State Transition

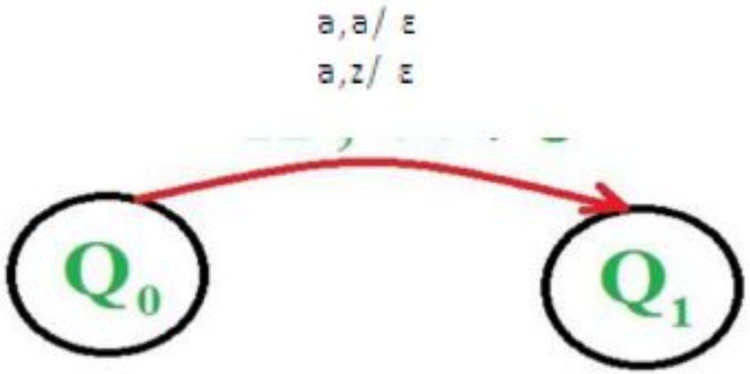
Input , Top of stack / new top of stack

Initially stack is empty , denoted by Z_0

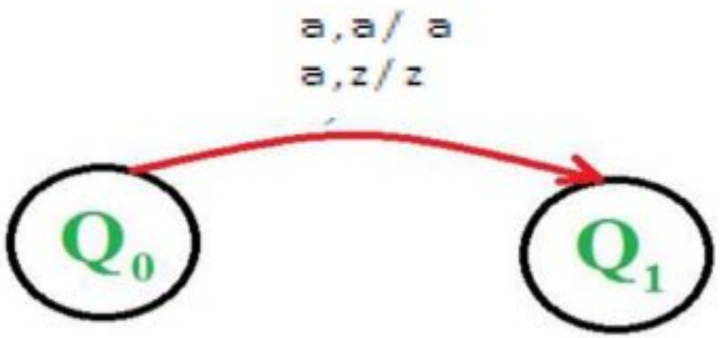
Representation of Push in a PDA



Representation of Pop in a PDA



Representation of Ignore in a PDA

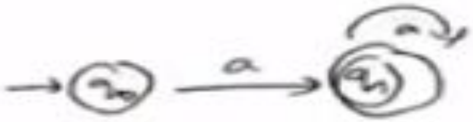


Give examples of languages handled by PDA.

- L={ anbn | n>=0 },here n is unbounded , hence counting cannot be done by finite memory. So we require a PDA ,a machine that can count without limit.
- L= { wwR | w Є {a,b}* } , to handle this language we need unlimited counting capability .

Design FA for accepting a language $\{a^n \mid n>=1\}$

$L = \{a, aa\ aaa, aaaa, \ldots\ldots\}$



Design FA for accepting a language $\{a^nb^m \mid m,n>=0\}$

Design FA for accepting a language $\{a^nb^m \mid m,n>=0\}$

L= $\{\varepsilon$,a,b,ab,aa,bb,aaa,aab,abb,bbb,aabb,aaabbb,aaaabbbb, ……$\}$

Constraints/limitations

1. a followed by b
2. N >=0

Ex:1

Problem: PDA Constructions

1) Design a PDA for accepting a language $L = \{a^n b^n \mid n \geq 1\}$

Solution:

Logic: First we will push all a's onto the stack. Then reading every single b each a is popped from the stack.

If we read all b and remove all a's and if we get stack empty then that string will be accepted.

Instantaneous Description:

$\left.\begin{array}{l} \delta(q_0, a, z_0) = \{(q_0, az_0)\} \\ \delta(q_0, a, a) = \{(q_0, aa)\} \end{array}\right\}$ pushing the elements onto stack

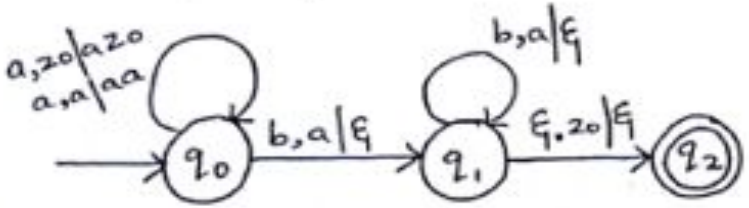
$\left.\begin{array}{l} \delta(q_0, b, a) = \{(q_1, \epsilon)\} \\ \delta(q_1, b, a) = \{(q_1, \epsilon)\} \\ \delta(q_1, \epsilon, z_0) = \{(q_2, \epsilon)\} \end{array}\right\}$ Popping the elements

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, \{q_2\})$

where $Q = \{q_0, q_1, q_2\}$

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

$\Gamma = \{a, z_0\}$



Example: Let $n = 2$, String $w = a^2 b^2 = aabb$

$\delta(q_0, aabb, z_0) \vdash (q_0, aabb, z_0)$

$\vdash (q_0, abb, az_0)$

$\vdash (q_0, bb, aaz_0)$

$\vdash (q_1, b, az_0)$

$\vdash (q_1, \epsilon, z_0)$

$\vdash (q_2, \epsilon)$ Accept state.

Ex:2

2) Construct a PDA for $L = \{wcw^R \mid w \text{ in } (0+1)^*\}$

Solution:

Logic → For each move, the PDA writes a symbol on the top of the stack.

→ If the tape head reaches the input symbol c, stop pushing onto the stack.

→ Compare the stack symbol with the i/p symbol. if it matches pop the stack symbol

→ Repeat the process till reaches the final state or empty stack.

Instantaneous Description:

$\delta(q_0,0,Z_0) = \{(q_0,0Z_0)\}$
$\delta(q_0,1,Z_0) = \{(q_0,1Z_0)\}$
$\delta(q_0,0,0) = \{(q_0,00)\}$
$\delta(q_0,0,1) = \{(q_0,01)\}$
$\delta(q_0,1,0) = \{(q_0,10)\}$
$\delta(q_0,1,1) = \{(q_0,11)\}$
} PUSH

$\delta(q_0,c,0) = \{(q_1,0)\}$
$\delta(q_0,c,1) = \{(q_1,1)\}$
} Accept the separator C

$\delta(q_1,0,0) = \{(q_1,\epsilon)\}$
$\delta(q_1,1,1) = \{(q_1,\epsilon)\}$
$\delta(q_1,\epsilon,Z_0) = \{(q_2,\epsilon)\}$
} POP

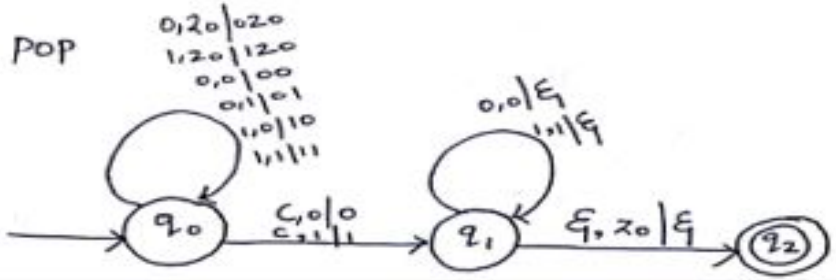
Here

$PDA = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \{q_2\})$

$Q = \{q_0, q_1, q_2\}$

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

$\Gamma = \{0,1,Z_0\}$



Example:

$\delta(q_0,100c001,Z_0) \vdash (q_0,100c001,Z_0)$
$\vdash (q_0,00c001,1Z_0)$
$\vdash (q_0,0c001,01Z_0)$
$\vdash (q_0,c001,001Z_0)$
$\vdash (q_1,001,001Z_0)$
$\vdash (q_1,01,01Z_0)$
$\vdash (q_1,1,1Z_0)$
$\vdash (q_1,\epsilon,Z_0)$
$\vdash (q_2,\epsilon)$ Accept State.

Ex:3

3) Construct PDA for the language $L=\{a^n b^{2n} \mid n \geq 1\}$.

Solution:

Logic: $L = \{$ 'n' number of a's followed by '2n' number of b's $\}$

If we read single 'a' push two a's onto the stack.

If we read 'b' then for every single 'b' only one 'a' should get popped from the stack.

Instantaneous Description:

$$\left.\begin{aligned} \delta(q_0, a, Z_0) &= \{(q_0, aaZ_0)\} \\ \delta(q_0, a, a) &= \{(q_0, aaa)\} \end{aligned}\right\} \text{PUSH}$$

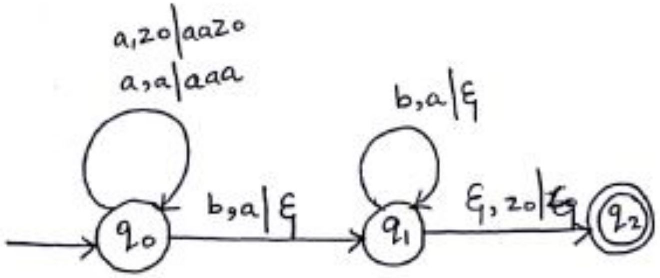
$$\left.\begin{aligned} \delta(q_0, b, a) &= \{(q_1, \epsilon)\} \\ \delta(q_1, b, a) &= \{(q_1, \epsilon)\} \\ \delta(q_1, \epsilon, Z_0) &= \{(q_2, \epsilon)\} \end{aligned}\right\} \text{POP}$$

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \{q_2\})$

$Q = \{q_0, q_1, q_2\}$

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

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



Example: Let $n=2$. $L = \{a^2 b^4\}$ string $w = aabbbb$

$\delta(q_0, aabbbb, Z_0) \vdash (q_0, aabbbb, Z_0)$

$\vdash (q_0, abbbb, aaZ_0)$

$\vdash (q_0, bbbb, aaaaZ_0)$

$\vdash (q_1, bbb, aaaZ_0)$

$\vdash (q_1, bb, aaZ_0)$

$\vdash (q_1, b, aZ_0)$

$\vdash (q_1, \epsilon, Z_0)$

$\vdash (q_2, \epsilon)$ Accept State.

Ex:4

4) Construct the PDA for the language $L=\{a^{2n}b^{n} \mid n\geq 1\}$. Trace your PDA for the input with $n=2$.

Solution:

Logic: When we read single 'b', single 'a' popped from the stack. For reading ϵ also single 'a' popped from the stack.

Instantaneous description:

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

$\delta(q_0,a,a)=\{(q_0,aa)\}$ PUSH

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

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

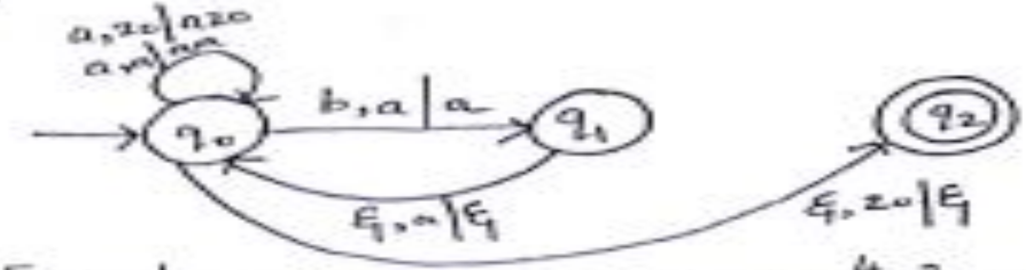
$\delta(q_0,\epsilon,z_0)=\{(q_2,\epsilon)\}$ POP

PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,z_0,\{q_2\})$

$Q=\{q_0,q_1,q_2\}$

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

$\Gamma=\{a,z_0\}$



Example: Let $n=2$ string $w=a^4b^2=aaaabb$

$\delta(q_0,aaaabb,z_0) \vdash (q_0,aaaabb,z_0)$

$\vdash (q_0,aaabb,az_0)$

$\vdash (q_0,aabb,aaz_0)$

$\vdash (q_0,abb,aaaz_0)$

$\vdash (q_0,bb,aaaaz_0)$

$\vdash (q_1,b,aaaz_0)$

$\vdash (q_0,b,aaz_0)$

$\vdash (q_1,\epsilon,az_0)$

$\vdash (q_0,\epsilon,z_0)$

$\vdash (q_2,\epsilon)$ Accept state.

Ex:5

5) Construct the DPDA for the language $L = \{0^n 1^m \mid n < m \text{ and } n, m \geq 1\}$

Solution:

ID:

$\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$

$\delta(q_0, 0, 0) = \{(q_0, 00)\}$

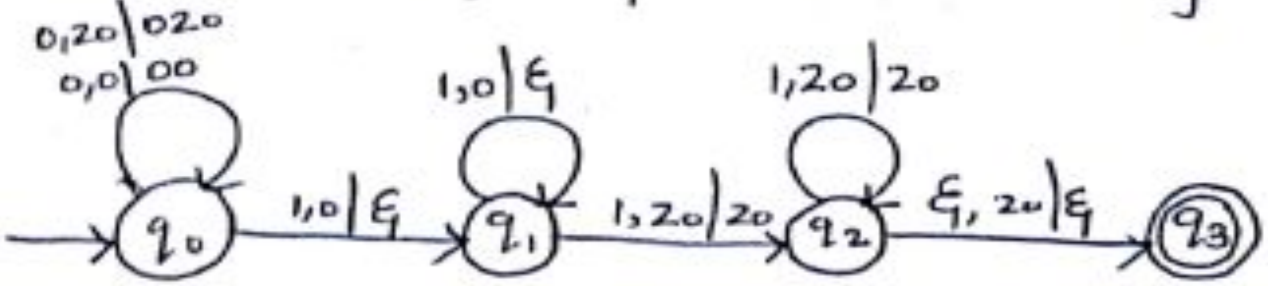
$\delta(q_0, 1, 0) = \{(q_1, \epsilon)\}$

$\delta(q_1, 1, 0) = \{(q_1, \epsilon)\}$

$\delta(q_1, 1, Z_0) = \{(q_2, Z_0)\}$

$\delta(q_2, 1, Z_0) = \{(q_2, Z_0)\}$

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



Example: $\delta(q_0, 00111, Z_0) \vdash (q_0, 00111, Z_0)$

$\vdash (q_0, 0111, 0Z_0)$

$\vdash (q_0, 111, 00Z_0)$

$\vdash (q_1, 11, 0Z_0)$

$\vdash (q_1, 1, Z_0)$

$\vdash (q_2, \epsilon, Z_0)$

$\vdash (q_3, \epsilon)$ Accept state

Assignment Exercises:

construct the PDA for the language $L=\{a^n b^m a^n \mid m,n \geq 1\}$

construct the PDA for the language $L=\{a^n b^m C^m d^n \mid m,n \geq 1\}$

Deterministic pushdown Automata

Deterministic pushdown automata

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ is deteministic if and only if it satisfies the following condition.

(i) $\delta(q, a, x)$ has at most one element

(ii) If $\delta(q, a, x)$ is nonempty for some $a \in \Sigma$ then $\delta(q, \epsilon, x)$ must be empty.

Non-Deterministic pushdown Automata

The non-deterministic pushdown automata is very much similar to NFA. The CFG which accepts deteministic PDA accepts non-deteministic PDAs as well.

Similarly there are some CFG's which can be accepted only by NDPA and not by DPDA. Thus NDPA is more powerful than DPDA.

Deterministic pushdown automata-Problems related to DPDA

Example 1:

Consider the PDA,

$$P = (Q, \sum, \Gamma, \delta, q_0, F)$$

where

$Q = \{s, f\}$

$\sum = \{a, b, c\}$

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

$q_0 = \{s\}$

$F = \{f\}$

δ is given as follpws:

1. $(s, a, \varepsilon) \longrightarrow (s, a)$
2. $(s, b, \varepsilon) \longrightarrow (s, b)$
3. $(s, c, \varepsilon) \longrightarrow (f, \varepsilon)$
4. $(f, a, a) \longrightarrow (f, \varepsilon)$
5. $(f, b, b) \longrightarrow (f, \varepsilon)$

Above is a deterministic pushdown automata (DPDA) because it satisfies both conditions of DPDA.

Deterministic pushdown automata-Problems related to DPDA

Construct a DPDA for a CFL $L = \{a^n b^n : n \geq 0\}$

Let Z_0 be the bottom element of a stack.

Let assume 3 state $\rightarrow q_0 \rightarrow$ Push the n a element to s

$q_1 \rightarrow$ Pop the n b elements

$q_2 \rightarrow$ Final acceptance.

$\therefore$ DPDA = $\{Q, \{q_0, q_1, q_2\}, \{a, b\}, \{a, b, Z_0\}, \delta, q_0, Z_0, \{$

δ is defined by following set of rules:

Push

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

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

Pop

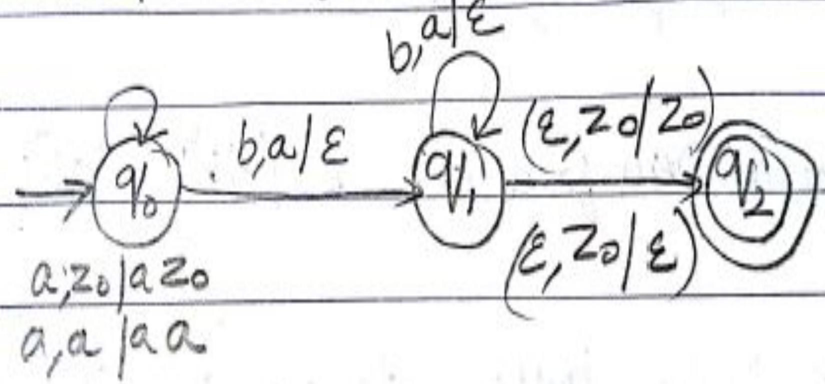
$\delta(q_0, b, a) = \{(q_1, \varepsilon)\}$

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

$\delta(q_1, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$ / $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$

$\Downarrow$ Acceptance by empty stack

$\Downarrow$ Acceptance by final state



Deterministic pushdown Automata-Problems related to DPDA

Consider the following PDA.

$P = (Q, \sum, \Gamma, \delta, q_0, F)$

where

$Q = \{q_0, q_f\}$

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

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

$q_0 = \{q_0\}$

$F = \{q_f\}$

δ is given as follipws:

1. $(q_0, \varepsilon, \varepsilon) \longrightarrow (q_f, \varepsilon)$
2. $(q_0, a, \varepsilon) \longrightarrow (q_0, 0)$
3. $(q_0, b, \varepsilon) \longrightarrow (q_0, 1)$
4. $(q_0, a, 0) \longrightarrow (q_0, 00)$
5. $(q_0, b, 0) \longrightarrow (q_0, \varepsilon)$
6. $(q_0, a, 1) \longrightarrow (q_0, \varepsilon)$
7. $(q_0, b, 1) \longrightarrow (q_0, 11)$

Above is a non deterministic pushdown automata (NPDA).

Consider the transitions, 1, 2 and 3.

1. $(q_0, \varepsilon, \varepsilon) \longrightarrow (q_f, \varepsilon)$
2. $(q_0, a, \varepsilon) \longrightarrow (q_0, 0)$
3. $(q_0, b, \varepsilon) \longrightarrow (q_0, 1)$

Here $(q_0, \varepsilon, \varepsilon)$ is not empty, also,

(q_0, a, ε) and (q_0, b, ε) are not empty.

NDPDA-Problems related to DPDA

A PDA is said to be non- deterministic, if

1. $\delta(q, a, b)$ may contain multiple elements, or
2. if $\delta(q, \varepsilon, b)$ is not empty, then

   $\delta(q, c, b)$ is not empty for some input symbol, c.

Example 1:

Consider the following PDA.

$$P = (Q, \sum, \Gamma, \delta, q_0, F)$$

where

$Q = \{q_0, q_1, q_2, q_3\}$

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

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ is given as follpws:

1. $(q_0, a, 0) \longrightarrow (q_1, 10), (q_3, \varepsilon)$
2. $(q_0, \varepsilon, 0) \longrightarrow (q_3, \varepsilon)$
3. $(q_1, a, 1) \longrightarrow (q_1, 11)$
4. $(q_1, b, 1) \longrightarrow (q_2, \varepsilon)$
5. $(q_2, b, 1) \longrightarrow (q_2, \varepsilon)$
5. $(q_2, \varepsilon, 0) \longrightarrow (q_3, \varepsilon)$

Above is a non deterministic pushdown automata (NPDA)

NDPDA-Problems related to DPDA

Consider the following PDA,

$$P = (Q, \sum, \Gamma, \delta, q_0, F)$$

where

$Q = \{q_0, q_f\}$

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

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

$q_0 = \{q_0\}$

$F = \{q_f\}$

δ is given as follpws:

1. $(q_0, \varepsilon, \varepsilon) \longrightarrow (q_f, \varepsilon)$
2. $(q_0, a, \varepsilon) \longrightarrow (q_0, 0)$
3. $(q_0, b, \varepsilon) \longrightarrow (q_0, 1)$
4. $(q_0, a, 0) \longrightarrow (q_0, 00)$
5. $(q_0, b, 0) \longrightarrow (q_0, \varepsilon)$
6. $(q_0, a, 1) \longrightarrow (q_0, \varepsilon)$
7. $(q_0, b, 1) \longrightarrow (q_0, 11)$

Above is a non deterministic pushdown automata (NPDA).

Consider the transitions, 1, 2 and 3.

1. $(q_0, \varepsilon, \varepsilon) \longrightarrow (q_f, \varepsilon)$
2. $(q_0, a, \varepsilon) \longrightarrow (q_0, 0)$
3. $(q_0, b, \varepsilon) \longrightarrow (q_0, 1)$

Here $(q_0, \varepsilon, \varepsilon)$ is not empty, also,

(q_0, a, ε) and (q_0, b, ε) are not empty.

Assignment Exercises:

$$L = \{ wcw^R \mid w \text{ is in } (0+1)^* \}$$

$$L = \{ a^n b^m a^n \mid m, n \geq 1 \}$$

Pushdown automata to CFL

Equivalence : pushdown automata to CFL.

Let, $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, q_n)$ is a PDA there exists CFG G which is accepted by PDA P. The G can be defined as,

$$G = (V, T, P, S)$$

Where S is a start symbol, T – Terminals V – Non-terminals

For getting production rules p, we follow the following algorithm.

Algorithm for getting production rules of CFG

1. If q_0 is start state in PDA and q_n is final state of PDA then $[q_0\ Z\ q_n]$ becomes start state of CFG.
2. The production rule for the ID of the form $\delta(q_i, a, z_0) = (q_{i+1}, Z_1 Z_2)$ can be obtained as,

$$\delta\ (q_i\ z_0\ q_{i+K}) \rightarrow a\ (q_{i+1}\ Z_1\ q_m)\ (q_m\ Z_2\ q_{i+K})$$

Where q_{i+K}, q_m represents the intermediate states, z_0, Z_1, Z_2 are Stack symbols and a is input symbol.

3. The production rule for the ID of the form.

$\delta(q_i, a, z_0) = (q_{i+1}, \epsilon)$ can be converted as $(q_i\ z_0\ q_{i+1}) \rightarrow a$

Pushdown automata to CFL

For every CFG, there exists a pushdown automation that accepts it.

To design a pushdown automation corresponding to a CFG, following are

Step 1:

Let the start symbol of the CFG is S. Then a transition of PDA is,

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

Step 2:

For a production of the form, $P \longrightarrow AaB$, a transition of PDA is,

$$\delta(q, \varepsilon, P) \longrightarrow (q, AaB)$$

For a production of the form, $P \longrightarrow a$, a transition of PDA is,

$$\delta(q, \varepsilon, P) \longrightarrow (q, a)$$

For a production of the form, $P \longrightarrow \varepsilon$, a transition of PDA is,

$$\delta(q, \varepsilon, P) \longrightarrow (q, \varepsilon)$$

Step 3:

For every terminal symbol, a in CFG, a transition of PDA is,

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

Thus the PDA is,

$$P = (Q, \sum, \Gamma, \delta, q_0, F)$$

where

$Q = \{p, q\}$

$\sum$ = set of terminal symbols in the CFG

Γ = set of terminals and non-terminals in the CFG

$q_0 = p$

$F = q$

δ is according to the above rules.

Pushdown automata to CFL Equivalence-Problems of PDA to CFG

SRM INSTITUTE OF SCIENCE & TECHNOLOGY (Deemed to be University u/s 3 of UGC Act, 1956)

Design a pushdown automata that accepts the language corresponding to the regular expression, $(a|b)^*$.

First, we need to find the CFG corresponding to this language. We learned in a previous section, the CFG corresponding to this is,

$S \longrightarrow aS|bS|a|b|\varepsilon$

where S is the start symbol.

Next we need to find the PDA corresponding to the above CFG.

Step 1:

Here start symbol of CFG is S. A transition is,

$$\delta(p,\varepsilon,\varepsilon) \longrightarrow (q,S)$$

Consider the production, $S \longrightarrow aS|bS|a|b|\varepsilon$

Transitions are

$$\delta(q,\varepsilon,S) \longrightarrow (q,aS)$$

$$\delta(q,\varepsilon,S) \longrightarrow (q,bS)$$

$$\delta(q,\varepsilon,S) \longrightarrow (q,a)$$

$$\delta(q,\varepsilon,S) \longrightarrow (q,b)$$

$$\delta(q,\varepsilon,S) \longrightarrow (q,\varepsilon)$$

Step 3:

The terminal symbols in the CFG are, a, b.

Then the transitions are,

$$\delta(q,a,a) \longrightarrow (q,\varepsilon)$$

$$\delta(q,b,b) \longrightarrow (q,\varepsilon)$$

Pushdown automata to CFL Equivalence-Problems of PDA to CFG

Thus the PDA is,

$$P = (Q, \sum, \Gamma, \delta, q_0, F)$$

where

$Q = \{p, q\}$

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

$\Gamma = \{S, a, b\}$

$q_0 = p$

$F = q$

δ is given as follows:

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, aS)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, bS)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, a)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, b)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, \varepsilon)$$

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

$$\delta(q, b, b) \longrightarrow (q, \varepsilon)$$

CFL to Pushdown automata Equivalence

Equivalence : CFL to pushdown automata

Algorithm:

(1) convert the CFG to Greibach Normal form.

(2) The δ function is to be developed for the grammar of the form

$$A \rightarrow aB \text{ as } \delta(q_i, a, A) \rightarrow \delta(q_i, B)$$

(3) Finally add the rule

$$\delta(q_i, \epsilon, z_0) \rightarrow (q_i, \epsilon)$$

where z_0 – Stack symbol (Accepting state)

CFL to Pushdown automata Equivalence -Problems

Problem 1: Construct PDA for the following grammar.

$S \to AB,\ B \to b,\ A \to CD,\ C \to a,\ D \to a$

Solution:

GNF form:

$S \to AB$

$\to CDB$

$\to aDB$

$A \to CD$

$\to aD$

$B \to b$

$C \to a$

$D \to a$

Equivalent PDA is

$\delta(q_1, a, S) \to (q_1, DB)$

$\delta(q_1, a, A) \to (q_1, D)$

$\delta(q_1, b, B) \to (q_1, \epsilon)$

$\delta(q_1, a, C) \to (q_1, \epsilon)$

$\delta(q_1, a, D) \to (q_1, \epsilon)$

Example: $\delta(q_1, aab, S) \vdash \delta(q_1, ab, DB)$

$\vdash \delta(q_1, b, B)$

$\vdash \delta(q_1, \epsilon, z_0)$

$\vdash \delta(q_f, \epsilon)$ Accepting state.

CFL to PDA - Problems

<u>Problem 2</u>: construct an unrestricted PDA equivalent of the grammar given below:

$S \rightarrow aAA, \quad A \rightarrow aS \mid bS \mid a$

<u>Solution</u>: The given grammar is already in GNF. Hence the PDA can be.

$\delta(q_1, a, S) \rightarrow (q_1, AA)$

$\delta(q_1, a, A) \rightarrow (q_1, S)$

$\delta(q_1, b, A) \rightarrow (q_1, S)$

$\delta(q_1, a, A) \rightarrow (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) \rightarrow (q_1, \epsilon)$ Accept.

The simulation of abaaaa is,

$\delta(q_1, abaaaa, S) \vdash \delta(q_1, baaaa, AA)$

$\vdash \delta(q_1, aaaa, SA)$

$\vdash \delta(q_1, aaa, AAA)$

$\vdash \delta(q_1, aa, AA)$

$\vdash \delta(q_1, a, A)$

$\vdash \delta(q_1, \epsilon, z_0)$

$\vdash \delta(q_1, \epsilon)$ Accept.

CFL to PDA - Problems

Problem3: Consider GNF $G = (\{S, T, C, D\}, \{a, b, c, d\}, S, P)$ where P is.

$S \rightarrow cCD \mid dTC \mid \epsilon$ $\qquad C \rightarrow aTD \mid c$

$T \rightarrow cDc \mid cST \mid a$ $\qquad D \rightarrow d\epsilon \mid d$

Present a PDA that accepts the language generated by this grammar.

Solution: Let PDA $M = \{\{q\}, \{c, a, d\}, \{S, T, C, D, c, d, a\}, \delta, q, S, \phi\}$

The production rules δ is given by

$\delta(q, \epsilon, S) = \{(q, cCD), (q, dTC), (q, \epsilon)\}$

$\delta(q, \epsilon, C) = \{(q, aTD), (q, c)\}$

$\delta(q, \epsilon, T) = \{(q, cDC), (q, cST), (q, a)\}$

$\delta(q, \epsilon, D) = \{(q, dC), (q, d)\}$

$\left.\begin{array}{l} \delta(q, c, c) = \{(q, \epsilon)\} \\ \delta(q, d, d) = \{(q, \epsilon)\} \\ \delta(q, a, a) = \{(q, \epsilon)\} \end{array}\right\}$ Acceptance by Empty Stack

Simulation for String "caadd"

$\delta(q, \epsilon, S) \vdash \delta(q, caadd, S)$

$\vdash \delta(q, aadd, CD)$

$\vdash \delta(q, add, TDD)$

$\vdash \delta(q, dd, DD)$

$\vdash \delta(q, d, D)$

$\vdash (q, \epsilon)$ Accept.

Assignment Exercises:

Problem 4: Find the PDA equivalent to given CFG with following productions.
$S \rightarrow A$, $A \rightarrow BC$, $B \rightarrow ba$, $C \rightarrow aC$

Problem 5: Convert the grammar $S \rightarrow aSb \mid A$, $A \rightarrow bSa \mid S \mid \epsilon$ to a PDA That accepts the same language by empty stack.

Problem 6: Convert the grammar $S \rightarrow 0S1 \mid A$, $A \rightarrow 1A0 \mid S \mid \epsilon$ into PDA That accepts the same language by empty stack. check whether 0101 belongs to $N(M)$.

Problem 7: Construct CFL for the grammar $S \rightarrow aSbb \mid a$ and also construct its Corresponding PDA.

Sln: $\{L = a^n b^m \mid m > n\} \longrightarrow$ PDA

Pumping Lemma for Context Free Languages (CFLs)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY (Deemed to be University u/s 3 of UGC Act, 1956)

Pumping lemma for CFL

Lemma: Let L be any CFL. Then there is a constant n, depending only on L, such that z is in L and $|z| \geq n$, then we can write $z = uvxyz$ such that

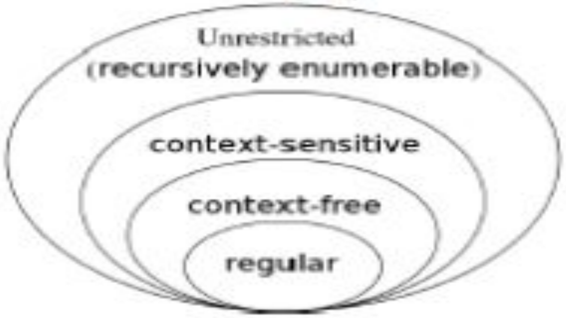
(i) $|vxy| \leq n$

(ii) $|vy| \geq 1$ (or) $|vy| \neq \epsilon$

(iii) for all $i \geq 0$, $uv^ixy^iz \in L$.

Pumping Lemma for Context Free Languages (CFLs)

Consider the following figure:



From the diagram, we can say that not all languages are context free languages. All context free languages are context sensitive and unrestricted. But all context sensitive and unrestricted languages are not context free from the above diagram.

We have a mechanism to show that some languages are not context free. The pumping lemma for CFLs allow us to show that some languages are not context free.

Pumping lemma for CFLs

Let G be a CFG. Then there exists a constant, n such that if W is in $L(G)$ and $|W| \geq n$, then we may write $W = uvxyz$ such that,

1. $|vy| \geq 1$ that is either v or y is non-empty,
2. $|vxy| \leq n$ then for all $i \geq 0$,

   uv^ixy^iz is in $L(G)$.

This is known as pumping lemma for context free languages.

Pumping Lemma for Context Free Languages (CFLs)

Pumping lemma for CFLs can be used to prove that a language, L is not context free.

We assume L is context free. Then we apply pumping lemma to get a contradiction.

Following are the steps:

Step 1:

Assume L is context free. Let n be the natural number obtained by using the pumping lemma.

Step 2:

Choose $W \in L$ so that $|W| \geq n$. Write $W = uvxyz$ using the pumping lemma.

Step 3:

Find a suitable k so that $uv^kxy^kz \notin L$. This is a contradiction, and so L is not context free.

Ex:1 Pumping Lemma for Context Free Languages (CFLs)

RM
E OF SCIENCE & TECHNOLOGY
e University u/s 3 of UGC Act, 1956)

Show that $L = \{a^p | p$ is a prime number$\}$ is not a context free language.

This means, if $w \in L$, number of characters in w is a prime number.

Step 1:

Assume L is context free. Let n be the natural number obtained by using the pumping lemma.

Step 2:

Choose $W \in L$ so that $|W| \geq n$. Write $W = uvxyz$ using the pumping lemma.

Let p be a prime number greater than n. Then $w = a^p \in L$.

We write $W = uvxyz$

Step 3:

Find a suitable k so that $uv^kxy^kz \notin L$. This is a contradiction, and so L is not context free.

By pumping lemma, $uv^0xy^0z = uxz \in L$.

So $|uxz|$ is a prime number, say q.

Let $|vy| = r$.

Then $|uv^qxy^qz| = q + qr$.

Since, $q + qr$ is not prime, $uv^qxy^qz \notin L$. This is a contradiction. Therefore, L is not context free.

Ex:2

Prove that $L = \{a^i b^i c^i \mid i \geq 1\}$ is not context free language.

Solution:

(i) Let us assume that L is regular / CFL

(ii) Let $w = a^i b^i c^i$ where i is constant

(iii) W can be written as $uvxyz$ where

(a) $|vxy| \leq n$

(b) $|vy| \neq \epsilon$

(c) for all $i \geq 0$, $uv^i xy^i z \in L$

Since $vy \neq \epsilon$, Either $v = ab \mid bc \mid ca$ (or)

$y = ab \mid bc \mid ca$.

If $i = 2$, $uv^i xy^i z = uv^2 xy^2 z$ becomes

case (i) If $v = ab$ and $y = c$

$uv^2 xy^2 z = (ab)^2 c^2 \Rightarrow uv^i xy^i z \notin L$

case (ii) If $v = a$ and $y = bc$

$uv^2 xy^2 z = a^2 (bc)^2 \Rightarrow uv^i xy^i z \notin L$

Hence L is not a CFL.

Ex:3

Prove that $L = \{0^i 1^j 2^i 3^j \mid i \geq 1, j \geq 1\}$ is not CFL.

Solution:

(i) Let us assume that L is CFL

(ii) Let $w = 0^n 1^n 2^n 3^n$ where n is constant

(iii) Let w can be rewritten as, $uvxyz$ where,

(a) $|vxy| \leq n$

(b) $|vy| \neq \epsilon$

(c) for all $i \geq 0$, $uv^ixy^iz \in L$.

Case(i) if $v = 01$ and $y = 2$

$i = 2$, $uv^2xy^2z = (01)^2 2^2 3^1$ —— ①

case(ii) if $v = 12$ and $y = 3$

$i = 2$, $uv^2xy^2z = (12)^2 (3)^2 0^1$ —— ②

From ① and ② $uv^ixy^iz \notin L$

$\therefore$ Given L is not a CFL.

End OF UNIT-III