Unit - 4

Push Down Automata

Page No.
Date

Pushdown automata is a finite automata with extra memory called stack which helps Pushdown automata to recognize context free language. The PDA will have input tape, finite control & stack. The input tape is divided in many cells. At each cell only one input symbol is placed. Thus certain input string is placed on tape.

The finite control has some pointer which points the current symbol which is to be read. At the end of input string blank symbol is placed which indicates end of input string.

The stack in such a structure in which you can push & pop the item from one end only. In the PDA, we are using the stacks for storing the items temporarily. Inserting the symbol onto the stack is called Push operation. & deleting the symbol from stack is called POP operation.

A PDA can be defined as-

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

$Q \rightarrow$ finite set of states

$\Sigma \rightarrow$ finite set of input symbol

$\Gamma \rightarrow$ finite set of stack alphabet

$\delta \rightarrow$ transition function

$q_0 \rightarrow$ initial state

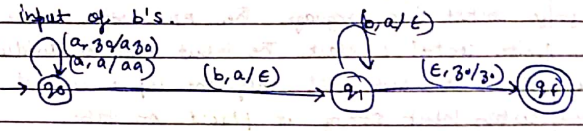
$Z_0 \rightarrow$ initial stack symbol

$F \rightarrow$ set of final states.

Page No. _____
Date

Q. Design a push down automata for $L = \{a^n b^n / n \geq 1\}$.

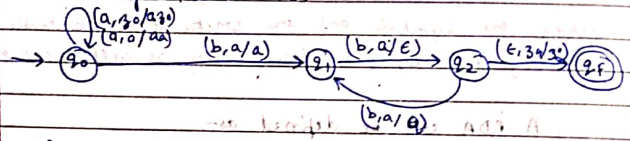
Sol First push all the a's then pop all a's with input of b's.



Push: $\delta(q_0, a, Z_0) \vdash (q_0, aZ_0)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
Pop: $\delta(q_0, b, a) \vdash (q_1, \epsilon)$
$\delta(q_1, b, a) \vdash (q_1, \epsilon)$
check if empty: $\delta(q_1, \epsilon, Z_0) \vdash (q_f, Z_0)$

Q. Design a PDA for $L = \{a^n b^{2n} / n \geq 1\}$

Sol first push all a's then for every two ~~three~~ b insertion of b, remove a's.

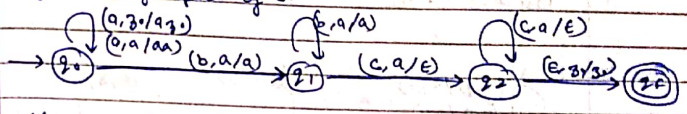


Push: $\delta(q_0, a, Z_0) \vdash (q_0, aZ_0)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
skip: $\delta(q_0, b, a) \vdash (q_1, a)$
Pop: $\delta(q_1, b, a) \vdash (q_2, \epsilon)$
$\delta(q_2, b, a) \vdash (q_1, a)$ (loop)
check if empty: $\delta(q_2, \epsilon, Z_0) \vdash (q_f, Z_0)$

Page No. _____
Date

Q. Design a PDA for $L = \{a^n b^m c^n / n, m \geq 1\}$

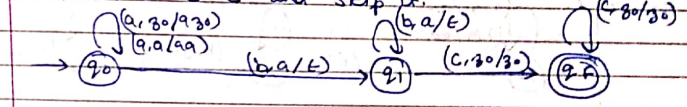
Sol first push all a's then skip b's and remove a after every input of c.



Push: $\delta(q_0, a, Z_0) \vdash (q_0, aZ_0)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
skip: $\delta(q_0, b, a) \vdash (q_1, a)$
$\delta(q_1, b, a) \vdash (q_1, a)$
Pop: $\delta(q_1, c, a) \vdash (q_2, \epsilon)$
$\delta(q_2, c, a) \vdash (q_2, \epsilon)$
check if empty: $\delta(q_2, \epsilon, Z_0) \vdash (q_f, Z_0)$

Q. Design a PDA for $L = \{a^n b^n c^m / n, m \geq 1\}$

Sol first push all a's then every input of b remove a, then take c and skip it.



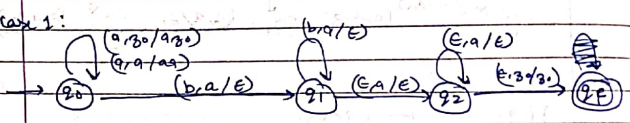
Push: $\delta(q_0, a, Z_0) \vdash (q_0, aZ_0)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
POP: $\delta(q_0, b, a) \vdash (q_1, \epsilon)$
$\delta(q_1, b, a) \vdash (q_1, \epsilon)$
Skip: $\delta(q_1, c, Z_0) \vdash (q_f, Z_0)$
$\delta(q_f, c, Z_0) \vdash (q_f, Z_0)$

Q. Design a PDA for $\{L = a^n b^m \mid n \neq m\}$

Sol. Case 1: $n > m$

Case 2: $n < m$.

for case 1:



Push: $\delta(q_0, a, Z_0) \vdash (q_0, aZ_0)$

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

Pop b: $\delta(q_0, b, a) \vdash (q_1, \epsilon)$

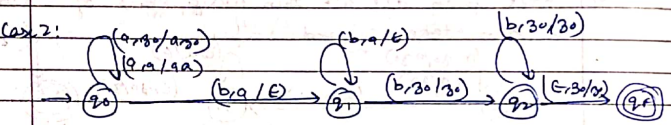
$\delta(q_1, b, a) \vdash (q_1, \epsilon)$

Pop a: $\delta(q_1, \epsilon, a) \vdash (q_2, \epsilon)$

$\delta(q_2, \epsilon, a) \vdash (q_2, \epsilon)$

check if empty: $\delta(q_2, \epsilon, Z_0) \vdash (q_F, Z_0)$

for case 2:



Push: $\delta(q_0, a, Z_0) \vdash (q_0, aZ_0)$

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

Pop: $\delta(q_0, b, a) \vdash (q_1, \epsilon)$

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

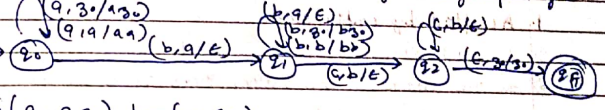
Pop extra b: $\delta(q_1, b, Z_0) \vdash (q_2, Z_0)$

$\delta(q_2, b, Z_0) \vdash (q_2, Z_0)$

check if empty: $\delta(q_2, \epsilon, Z_0) \vdash (q_F, Z_0)$.

Q. Design a PDA for $L = \{a^m b^{(m+n)} c^n \mid m, n \geq 1\}$.

Sol. First push all a's onto stack then remove a's for every input of b when stack is empty, push b's into stack then remove b's for every input of c.



Push a: $\delta(q_0, a, Z_0) \vdash (q_0, aZ_0)$

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

pop b: $\delta(q_0, b, a) \vdash (q_1, \epsilon)$

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

Push b: $\delta(q_1, b, Z_0) \vdash (q_1, bZ_0)$

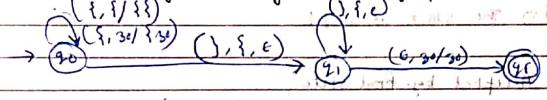
$\delta(q_1, b, b) \vdash (q_1, bb)$

Pop c: $\delta(q_1, c, b) \vdash (q_2, \epsilon)$

$\delta(q_2, c, b) \vdash (q_2, \epsilon)$

check if empty: $\delta(q_2, \epsilon, Z_0) \vdash (q_F, Z_0)$

Q. Design PDA for well formatted of parenthesis.



Push: $\delta(q_0, \{, Z_0) \vdash (q_0, \{Z_0)$

$\delta(q_0, \{, \{) \vdash (q_0, \{\{)$

Pop: $\delta(q_0,), \{) \vdash (q_1, \epsilon)$

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

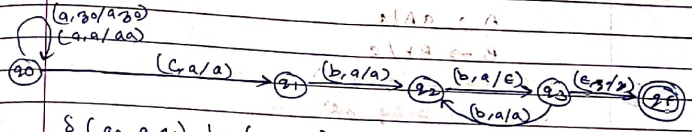
check if empty: $\delta(q_1, \epsilon, Z_0) \vdash (q_F, Z_0)$

There are two ways to declare a string accepted by PDA:
1) Accepted by empty stack
2) Accepted by final state

1. Accepted by empty stack
a) In each PDA, there is a stack attached
b) In the stack at the bottom, there is a symbol called stack bottom symbol.
c) In each move of the PDA, one symbol called stack symbol is either pushed in or popped from the stack. But the symbol z_0 still remains in the stack.
d) A string w may be declared accepted by empty stack after processing all the symbol of w, if the stack is empty after reading the rightmost input character of the string.
e) In general, we can say that a string is accepted by PDA by empty stack if both the following conditions are fulfilled.
i) The string is finished.
ii) The stack is empty.

2. Accepted by final state
a) A string may be declared accepted by final state if after total traversal of the string w, the PDA enters into its final state.
b) In general, we can say that a string is accepted by a PDA by final state if both the following conditions are fulfilled.
i) The string is finished
ii) The PDA enters into its final state.

a. Construct PDA for the following:
$L = \{ a^n c b^{2n} \mid n >= 1 \}$



$\delta(q_0, a, z_0) \vdash (q_0, aaz_0)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
$\delta(q_0, c, a) \vdash (q_0, a)$
$\delta(q_1, b, a) \vdash (q_2, a)$
$\delta(q_2, b, a) \vdash (q_3, \epsilon)$
$\delta(q_3, b, a) \vdash (q_2, a)$
$\delta(q_3, \epsilon, z_0) \vdash (q_f, z_0)$

Deterministic Pushdown Automata (DPDA):
The model should be deterministic in the sense that it should not show more path moving to different state for the same i/ps.
Infact, we have observe all the model, they show that for each specific input, there is only one specific path like as $L = a^n b^n$, as well as example of checking the well formedness of parenthesis.
The concept of DPDA is similar to DFA.

Non-Deterministic Pushdown Automata (NDPA):
The non DPDA is very similar to NFA.
- The CFG which accepts DPDA accepts ND PDA as well as.
- Similarly, there are some CFG which can be accepted by NPDA

Page No.
Date

Q. Convert the following CFG into PDA:

S → aSa/aA/Bb

A → aA/a

B → Bb/a

Sol. $\delta(q, \epsilon, S) = (q, aSa)$

$\delta(q, \epsilon, S) = (q, aA)$

$\delta(q, \epsilon, S) = (q, Bb)$

$\delta(q, \epsilon, A) = (q, aA)$

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

$\delta(q, \epsilon, B) = (q, Bb)$

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

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

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

Q. Consider the CFG where production P are as follows:

S → aABB / aAA

A → aBB / a

B → bBB / A

Sol $\delta(q, \epsilon, S) = (q, aABB)$

$\delta(q, \epsilon, S) = (q, aAA)$

$\delta(q, \epsilon, A) = (q, aBB)$

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

$\delta(q, \epsilon, B) = (q, bBB)$

$\delta(q, \epsilon, B) = (q, A)$

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

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

Page No.
Date

Q. Construct an equivalent PDA for the given CFG:

S → aAB / bBA

A → bS/a

B → aS/b

Shown ID for the string abb aaa bbbb ab for the PDA generated with stack description.

Sol $\delta(q, \epsilon, S) = (q, aAB)$

$\delta(q, \epsilon, S) = (q, bBA)$

$\delta(q, \epsilon, A) = (q, bS)$

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

$\delta(q, \epsilon, B) = (q, aS)$

$\delta(q, \epsilon, B) = (q, b)$

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

for string abb aaa bbbb ab

$\delta(q, abbaaabbbbab) \vdash (q, abb\ aaabbbbab, aAB)$

$\vdash (q, bbaaabbbbab, AB)$

$\vdash (q, bbaaabbbbab, bSB)$

$\vdash (q, baaabbbbab, SB)$

$\vdash (q, baaabbbbab, bBAB)$

$\vdash (q, aaabbbbab, BAB)$

$\vdash (q, aaabbbbab, aSAB)$

$\vdash (q, aabbbbab, SAB)$

$\vdash (q, aabbbbab, aABAB)$

$\vdash (q, abbbbab, ABAB)$

$\vdash (q, abbbbab, aBAB)$

$\vdash (q, bbbbab, BAB)$

$\vdash (q, bbbab, bAB)$

$\vdash (q, bbab, AB)$

$\vdash (q, bbab, bSB)$

$\vdash (q, bab, SB)$

$\vdash (q, bab, BAB)$

$\vdash (q, bab, bAB)$

$\vdash (q, ab, AB)$

$\vdash (q, b, B)$

$\vdash (q, b, b)$

$\vdash (q, \epsilon)$

Page No.
Date

Q. Construct a PDA from the following CFG.

$S \to XS / \epsilon$

$X \to aXb / Xb / ab$

Sol $\delta(q, \epsilon, S) = (q, XS)$

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

$\delta(q, \epsilon, X) = (q, aXb)$

$\delta(q, b, X) = (q, Xb)$

$\delta(q, b, X) = (q, ab)$

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

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

And

$Q = \{q\}$ $\quad \Sigma = \{a, b\}$, $\quad \Gamma = \{S, X, a, b\}$

$q_0 = \{S\}$ $\quad F = \{q\}$

Q Convert the grammar

$S \to a\,Sb / A$

$A \to bAa / S / \epsilon$

to PDA that accepts the same language by empty stack.

Sol $\delta(q, \epsilon, S) = (q, aSb)$

$\delta(q, \epsilon, S) = (q, A)$

$\delta(q, \epsilon, A) = (q, bAa)$

$\delta(q, \epsilon, A) = (q, S)$

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

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

$\delta(q, b, b) = (q, \epsilon)$.

And

$Q = \{q\}$ $\quad \Sigma = \{a, b\}$ $\quad \Gamma = \{S, A, a, b\}$

$q_0 = \{S\}$ $\quad F = \{q\}$

Page No.
Date

PDA to CFG Conversion

The production P are induced by moves of PDA as follows:

1) S productions are given by

$S \to [q_0, z_0, q]$ for every $q \in Q$

2) Each erasing move $\delta(q, a, z) = (q', \epsilon)$ induces production $[q\ z\ q'] \to a$

3) Each non erasing move

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

induces production of form

$[q\ z\ q_1] \to a\,[q_1, z_1, q_2][q_2, z_2, q_3] \ldots [q_m\ z_m\ q_n]$

where each state $q_1, q_2 \ldots q_m$ can be any state in Q.

Q. Convert the given PDA M to equivalent CFG. where δ is given as follows:

$(q_0, 1, z_0) = (q_0, X z_0)$

$(q_0, 1, X) = (q_0, XX)$

$(q_0, 0, X) = (q_0, X)$

$(q_0, \epsilon, X) = (q_1, \epsilon)$

$(q_1, 1, X) = (q_1, \epsilon)$

$(q_1, 0, X) = (q_1, XX)$

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

Sol Step 1: $\delta(q_0, 1, z_0) = (q_0, X z_0)$

$[q_0, 1, q_0] = 1\,[q_0\ X\ q_0]\,[q_0\ z_0\ q_0]$

$[q_0, 1, q_0] = 1\,[q_0\ X\ q_1]\,[q_1\ z_0\ q_0]$

$[q_0, 1, q_1] = 1\,[q_0\ X\ q_0]\,[q_0\ z_0\ q_1]$

$[q_0, 1, q_1] = 1\,[q_0\ X\ q_1]\,[q_1\ z_0\ q_1]$

Page No.
Date

Step 2: $\delta(q_0, 1, X) = (q_0, XX)$

$[q_0, X, q_0] = 1 [q_0 X q_0] [q_0 X q_0]$
$[q_0, X, q_0] = 1 [q_0 X q_1] [q_1 X q_0]$
$[q_0, X, q_1] = 1 [q_0 X q_0] [q_0 X q_1]$
$[q_0, X, q_1] = 1 [q_0 X q_1] [q_1 X q_1]$

Step 3: $\delta(q_0, 0, X) = (q_0, X)$

$[q_0, X, q_0] = 0 [q_0, X, q_0]$
$[q_0, X, q_1] = 0 [q_0, X, q_1]$

Step 4: $\delta(q_0, \epsilon, X) = (q_1, \epsilon)$
$[q_0 X q_1] \rightarrow \epsilon$

Step 5: $\delta(q_1, \epsilon, X) = (q_1, \epsilon)$
$[q_1 X q_1] \rightarrow \epsilon$

Step 6: $\delta(q_1, 0, X) = (q_1, XX)$

$[q_1, X, q_0] = 0 [q_1, X, q_0] [q_0, X, q_0]$
$[q_1, X, q_0] = 0 [q_1, X, q_1] [q_1, X, q_0]$
$[q_1, X, q_1] = 0 [q_1, X, q_0] [q_0, X, q_1]$
$[q_1, X, q_1] = 0 [q_1, X, q_1] [q_1, X, q_1]$

Step 7: $\delta(q_1, 0, Z_0) = (q_1, \epsilon)$

$[q_1, Z_0, q_1] \rightarrow 0$

Page No.
Date

Q. Construct given PDA:
$\delta(q_0, a, Z_0) = (q_0, aZ_0)$
$\delta(q_0, 1, Z_0) = (q_0, bZ_0)$
$\delta(q_0, 0, Z_0) = (q_0, aa)$
$\delta(q_0, 1, b) = (q_0, bb)$
$\delta(q_0, 0, b) = (q_0, \epsilon)$
$\delta(q_0, 1, a) = (q_0, \epsilon)$
$\delta(q_0, \epsilon, Z_0) = (q_0, \epsilon)$

Sol. Step 1: $\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$[q_0, Z_0, q_0] = a [q_0, a, q_0] [q_0, Z_0, q_0]$

Step 2: $\delta(q_0, 1, Z_0) = (q_0, bZ_0)$
$[q_0, Z_0, q_0] = 1 [q_0, b, q_0] [q_0, Z_0, q_0]$

Step 3: $\delta(q_0, 0, Z_0) = (q_0, aa)$
$[q_0, Z_0, q_0] = 0 [q_0, a, q_0] [q_0, a, q_0]$

Step 4: $\delta(q_0, 1, b) = (q_0, bb)$
$[q_0, b, q_0] = 1 [q_0, b, q_0] [q_0, b, q_0]$

Step 5: $\delta(q_0, 0, b) = (q_0, \epsilon)$
$(q_0, b, q_0) \rightarrow 0$

Step 6: $\delta(q_0, 1, a) = (q_0, \epsilon)$
$(q_0, a, q_0) \rightarrow 1$

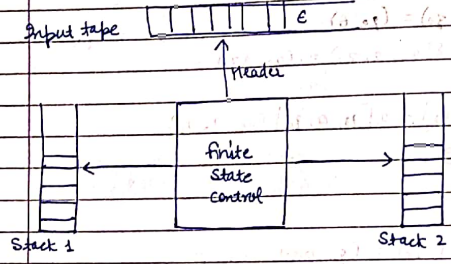
Step 7: $\delta(q_0, \epsilon, Z_0) = (q_0, \epsilon)$
$(q_0, Z_0, q_0) = \epsilon$

Two stack PDA

A two stack PDA is similar to a PDA, but it has two stacks instead of one. In each transition, we must denote the pop and push action on both stacks.

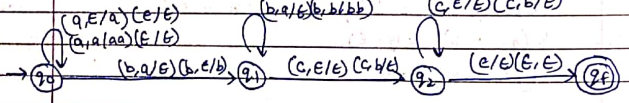
A two-stack PDA has six tuple $\{Q, \Sigma, \delta, q_0, \Gamma, F\}$

$\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \times \Gamma \longrightarrow Q \times \Gamma^* \times \Gamma^*$



Q. Design two stack PDA for the language $L = \{a^n b^n c^n / n \geq 0\}$

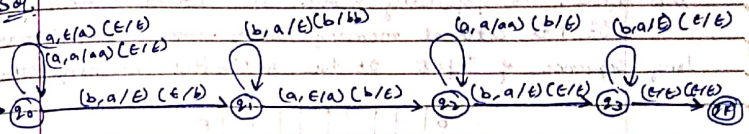
Sol. Mechanism: Push a into stack 1, for every input of b, pop a from stack 1 and push b into stack 2 then for every input of c, pop b.



$\delta(q_0, a, \epsilon)(q_0, \epsilon, \epsilon) = (q_0/a)(q_0/\epsilon)$
$\delta(q_0, a, a)(q_0, b, \epsilon) = (q_0/aa)(q_0/\epsilon)$
$\delta(q_0, b, a)(q_0, b, \epsilon) = (q_1/\epsilon)(q_1/b)$
$\delta(q_1, b, a)(q_1, b, b) = (q_1/\epsilon)(q_1/bb)$
$\delta(q_1, c, \epsilon)(q_1, c, b) = (q_2, \epsilon)(q_2, \epsilon)$
$\delta(q_2, c, \epsilon)(q_2, c, b) = (q_2, \epsilon)(q_2, \epsilon)$
$\delta(q_2, \epsilon, \epsilon)(q_2, \epsilon, \epsilon) = (q_2, \epsilon)(q_2, \epsilon)$

Q. Construct a two-stack PDA that will accept the language $L = \{a^n b^n a^n b^n / n \geq 1\}$

Sol:

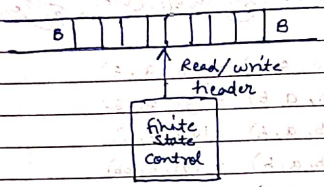


$\delta(q_0, a, \epsilon, \epsilon) = (q_0, a, \epsilon)$
$\delta(q_0, a, a, \epsilon) = (q_0, aa, \epsilon)$
$\delta(q_0, b, a, \epsilon) = (q_1, \epsilon, b)$
$\delta(q_1, b, a, b) = (q_1, \epsilon, bb)$
$\delta(q_1, a, \epsilon, b) = (q_2, a, \epsilon)$
$\delta(q_2, a, a, b) = (q_2, aa, \epsilon)$
$\delta(q_2, b, a, \epsilon) = (q_3, \epsilon, \epsilon)$
$\delta(q_3, b, a, \epsilon) = (q_3, \epsilon, \epsilon)$
$\delta(q_3, \epsilon, \epsilon, \epsilon) = (q_f, \epsilon, \epsilon)$.

Unit 5
Turing Machine

Page No. / Date

Alan Turing is father of such a model which has computing capability of general purpose computer. Hence this model is popularly known as Turing Machine.
It has external memory which remembers arbitrarily long sequence of input. It has unlimited memory capability.



It has seven tuples $M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$

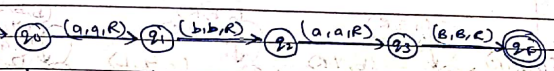
Q is a finite set of states
Σ is a finite set of input symbols
Γ is a finite set of tape alphabet
δ is transition or mapping function
q_0 is initial state
B is blank symbol
F is final state.

Instantaneous Description

$(q_0, a) \longrightarrow (q_1, A, L/R)$

Page No. / Date

Q. Construct a turing machine which accepts a a's over $\Sigma = \{a, b\}$.

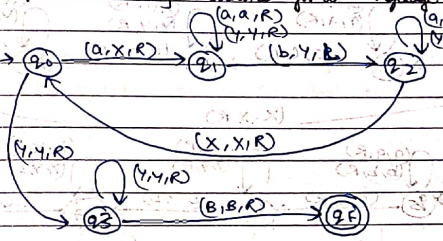


	a	b	B
q_0	(q_1, a, R)	—	—
q_1	—	(q_2, b, R)	—
q_2	(q_3, a, R)	—	—
q_3	—	—	(q_f, B, R)

Representation = (Next State, o/p to be printed, Direction)

Q. Design a turing machine for a language $L = \{a^n b^n / n \geq 1\}$.

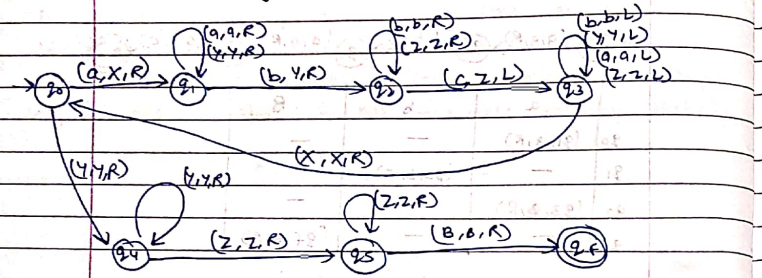
Sol



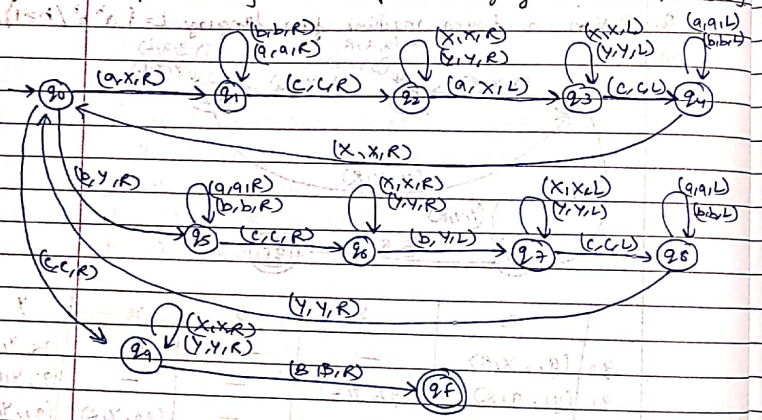
	a	b	B	X	Y
q_0	(q_1, X, R)	—	—	—	(q_3, Y, R)
q_1	(q_1, a, R)	(q_2, Y, L)	—	—	(q_1, Y, R)
q_2	(q_2, a, L)	—	—	(q_0, X, R)	(q_2, Y, L)
q_3	—	—	(q_f, B, R)	—	(q_3, Y, R)

Page No.
Date

Q. Design a turing machine for the language $L = \{a^n b^n c^n, n \geq 1\}$

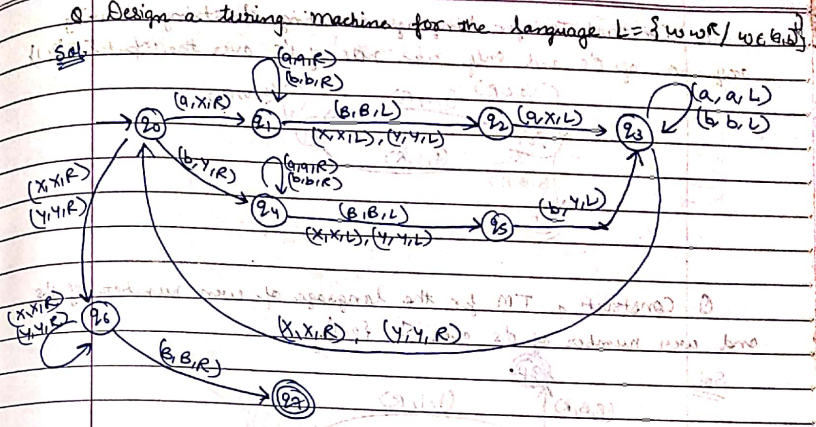


Q. Design a turing machine for the language $L = \{w c w / w \in \{a,b\}^*\}$



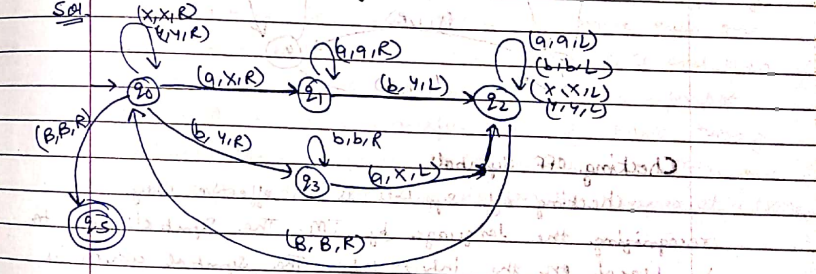
Q. Design a turing machine for the language $L = \{w w^R / w \in \{a,b\}^*\}$

Sol:



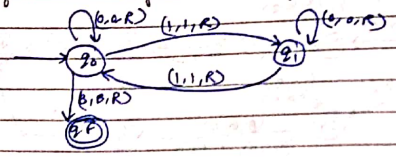
Q. Design a turing machine for the language which has equal no's of a's and b's.

Sol:



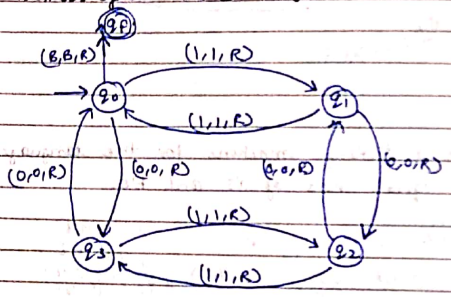
Q. Construct TM for language consisting of strings having any no. of 0's and only even nos of 1's over the input $\Sigma = \{0, 1\}$.

Sol.



Q. Construct a TM for the language of even number of 1's and even number of 0's over $\Sigma = \{0, 1\}$.

Sol.

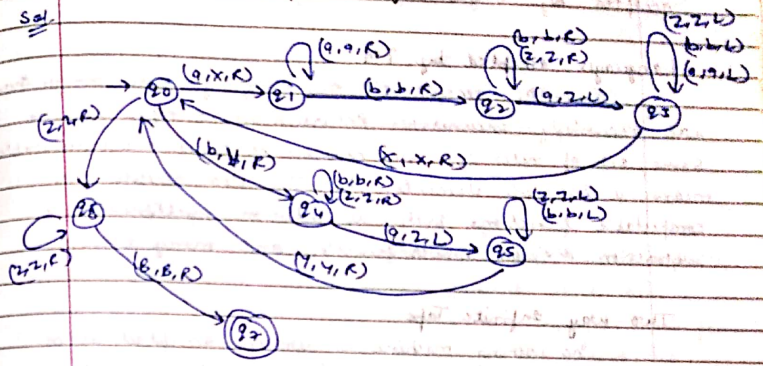


Checking off Symbols

Checking off symbols is an effective way of recognizing the language by TM. The symbols are to be placed on the input tape. The symbol which is read is marked by any special character. The tape head can be moved to the right or left.

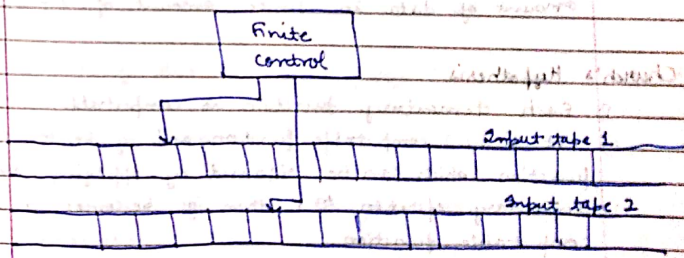
Q. Design a turing machine which accepts the language $L = \{ a^n b^m a^{n+m} \mid n, m > 0 \}$

Sol.



Multi-Tape Turing Machine

The multi-tape turing machine is a type of turing machine in which there are more than one input tapes. Each tape is divided into cells and each cell can hold any symbol of finite tape alphabet. This TM is more powerful than the basic turing machine. Because finite control reads more than one input tape and more symbols can be scanned at a time.



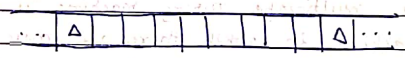
If L is accepted by a multitape turing machine, it is accepted by single tape turing machine also.

Languages accepted by TM

The TM accepts all the languages even though they are recursively enumerable. Recursive means repeating the same set of rules for any number of times and enumerable means a list of elements. The TM also accepts the computable functions such as addition, multiplication, subtraction, division, power function and many more.

Two-way Infinite Tape

The turing machine is widely accepted as a model of computation because of its versatality in the power of computation.



Universal Turing Machine

1) It is a kind of turing machine which can simulate any other turing machine. In other words, UTM is a single machine used to compute any computable sequence.
2) It has an ability to manipulate an unbounded amount of data in finite amount of time.

Church's Hypothesis

1) Each elementary function is computable.
2) Let f be computable function and g be the another function which can be obtained by applying an elementary operation to f, then g becomes a computable function.

Recursive and Recursively Enumerable Languages

Naturally such turing machine always halts on valid input and enters in accept state, but on invalid input, it halts without entering in halt state. The languages of this category are particularly called as recursive languages.

The languages can be modeled by turing machines but there is no guarantee that TM will eventually halt. In the sense that we can not predict that turing machine will halt or will enter in an infinite loop for certain input. Such type of languages are called as recursively enumerable language. (RE)

Decidable and Undecidable Languages

If a language is recursive, then it is called decidable languages and if the language is not recursive then such a language is called undecidable language.

1. If L is recursive language, L' is also a recursive language.
2. If L and its complement L' both are RE then L is a recursive language.
3. Union of two recursive languages is recursive.
4. Union of two RE language is RE.
5. Recursively enumerable language are closed under intersection.

Tractable and non-tractable problems

Tractable problems are kind of problems that can be solved by polynomial time algorithm.

Ex - 1) Searching an ordered or unordered list.
2) Sorting a list.
3) finding minimum spanning tree

Non-tractable problems are kind of problems that can not be solved by polynomial time problems.

Ex - 1) finding permutations of n numbers.
2) Tower of Hanoi.

Halting Problem

Given any functional matrix, input data tape and initial configuration, then is it possible to determine whether the process will ever halt? This is called halting problem.

i) Halt: The machine starting at this configuration will halt after a finite number of states.
ii) No Halt: The machine starting at this configuration never reaches a halt state, no matter how long it runs.

Undecidability

The class of problems which can be answered as "yes" are called solvable or decidable, otherwise the class of problems is said to be unsolvable or undecidable.

Post Correspondence Problem

The undecidability of strings is determined with the help of Post's Correspondence Problem (PCP). It consists of two lists of strings that are of equal length over the input Σ. The two lists are $A = w_1, w_2 \ldots w_n$ and $B = x_1, x_2 \ldots x_n$ then there exists a non empty set of integers $i_1, i_2, i_3 \ldots i_n$ such that,

$$w_1, w_2, w_3 \ldots w_n = x_1, x_2, x_3 \ldots x_n.$$

Q. Consider the PCP as below: $A = \{1; 0; 010; 11\}$ and $B = \{10; 10; 01; 1\}$. Find the solution.

Sol.

A = 1 0 1 010 010 11
B = 10 10 10 01 01 1

Solution: 1 2 1 3 3 4

Chomsky Hierarchy