5675154

B.Tech. DEGREE EXAMINATION JANUARY 2023

Fifth Semester

Information Technology

THEORY OF COMPUTATION

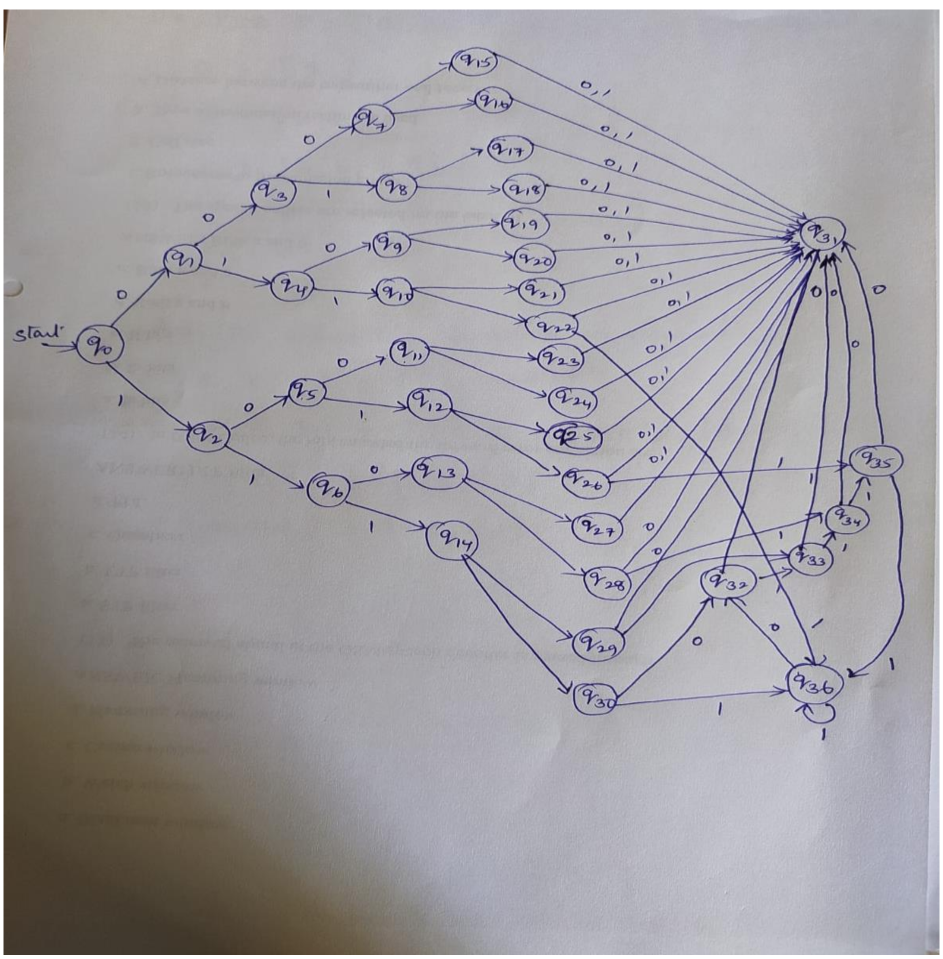
(2013 – 14 Regulations)

Time: Three hours Maximum: 75 marks

SECTION A – (10 * 2 = 20 marks)

Answer ALL the questions.

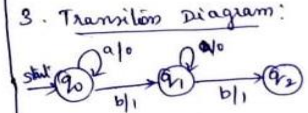
1. Give the DFA accepting the language over the alphabet 0, 1 that has the set of all strings such that each block of 5 consecutive symbol contains at least two 0's.



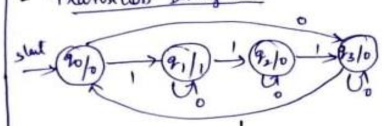
2. Compare the functionalities of Mealy and Moore machines.

Mealy machine	Moore machine
1. Each and every transition contains the output.	Each and every state contains the output.
2. It consists of 6-tuples $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where. Q - finite set of states; Σ - finite set of input symbols; Δ - finite set of output alphabets; δ - set of input transitions; λ - finite set of output transitions; q_0 - starting state	- It consists of 6-tuples. $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$.

3. Transition Diagram:



- Transition Diagram.



4. Transition Table

Present state	Next state a = 0: state	output	a = 1: state	output
→ q_0	q_0	0	q_1	1
q_1	q_1	0	q_2	1
q_2	ϕ	ϕ	ϕ	ϕ

- Transition Table.

Present state	Next state a = 0	a = 1	output
q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

3. Give English description of the following language (0+10)*1*.

This is the language of strings in which there are no two consecutive 1's except for possibly a string of 1's at the end.

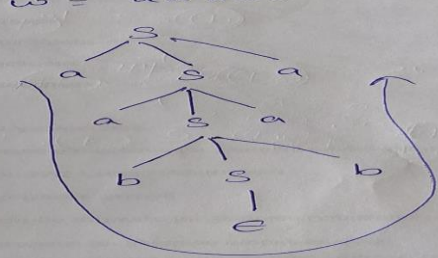
4. Let G be the grammar with S → aSa / bSb / ϵ

Construct Parse tree for the input string w = 'aabbaa'

$S \to aSa \mid bSb \mid \epsilon$

Sol: $S \to aSa$
$S \to bSb$
$S \to \epsilon$

Input string $w = aabbaa$



5. List the primary objectives of Turing machine.

The main advantages of the Turing machine is we have a tape head which can be moved forward or Backward and the input tape can be scanned. The simple logic which we will apply is read out each '0' Mark it by A and then move ahead along with the input tape and find out 1 convert it to B.

6. State when a problem is said to be undecidable and give an example of an undecidable problem.

A problem whose language is recursive is said to be decidable. Otherwise the problem is said to be undecidable. Decidable problem have an algorithm that takes as input an instance of the problem and determine whether the answer to that instance is "yes" or "no".

Eg. Of undecidable problems are (1) Halting problem of the TM.

7. Convert the following CFG to PDA.

S → aAA, A → aS / bS / a.

The PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is defined as

$Q = \{q\}$

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

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

$q_0 = q$

$Z_0 = S$

$F = \{\ \}$

And the transition function is defined as:

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

$\delta(q, \epsilon, A) = \{(q, aS), (q, bS), (q, a)\}$

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

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

8. Does a Pushdown Automata have memory? Justify.

Yes. Finite automata can be used to accept only regular languages. Pushdown automata is a finite Automata with extra memory called stack which helps pushdown automata to recognize Context Free Languages.

9. Differentiate Top down and bottom up parsing approaches.

TOP - DOWN PARSER	BOTTOM - UP PARSER
1. This is top-down (LL) parser.	This is bottom-up (LR) parser.
2. It is attempts to find left most derivations for an input string.	It can be defined as an attempt to reduce the input string to the start symbol of a grammar.
3. In this parsing technique we start parsing from the top to down (start symbol of parse tree to the leaf node of parse tree) in a top-down manner.	In this parsing technique we start parsing from the bottom to top (leaf node of parse tree to start symbol of parse tree) in a bottom-up manner.
4. This parsing techniques uses Left Most Derivation.	This parsing technique uses Right Most Derivation.
5. The main leftmost decision is to select what production rule to use in order to construct the string.	The main decision is to select when to use a production rule to reduce the string to get the starting symbol.
6. Eg. Recursive Descent parser or Predictive Descent parser.	Eg. Shift Reduce parser.

10. Consider the following grammar

S → Aa / b

A → Ac / Sd / ϵ

Eliminate the left recursion.

$S \rightarrow Aa \mid b$

$A \rightarrow Ac \mid Sd \mid \epsilon$

There is no immediate left recursion.

* To obtain the immediate left recursion substitute the S- productions in $A \rightarrow Sd$.

$A \rightarrow Ac \mid Aad \mid bd \mid \epsilon$

* Replaced as

$A \rightarrow bdA' \mid A'$

$A' \rightarrow cA' \mid adA' \mid \epsilon$

* Finally, we obtain

$S \rightarrow Aa \mid b$

$A \rightarrow bdA' \mid A'$

$A' \rightarrow cA' \mid adA' \mid \epsilon$.

SECTION B – (5 * 11 = 55 marks)

Answer ALL questions.

UNIT – I

11. Design a NFA accepts the following strings over the alphabet {0, 1}. The set of all string that begins with 01 and ends with 00. Check for the validity of 011100 and 01100 strings and find its equivalent DFA.

Sol

NFA : Non - Deterministic Finite Automata

- It is transition from more than one input symbols. It is called as NFA.
- 5 - tuples :

$$M = (Q, \Sigma, \delta, q_0, F).$$

where

$Q = \{q_0, q_1, q_2\}$. - set of states

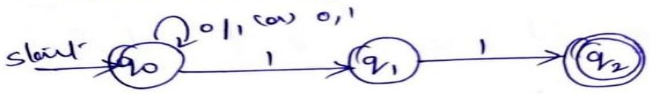
$\Sigma = \{0, 1\}$. - set of input symbols.

$\delta : Q \times \Sigma^* \rightarrow 2^Q$ - mapping function.

$q_0 = \{q_0\}$ - starting state.

$F = \{q_2\}$ - Final state

- Transition diagram for all strings that begins with 01 and ends with 00.



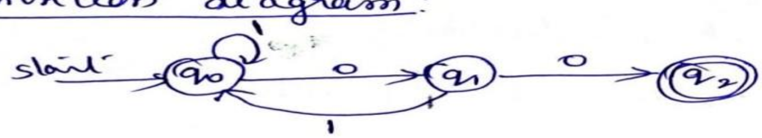
NFA, $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$.

- Transition Table

states	input symbols 0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	ϕ
$* q_2$	ϕ	ϕ

DFA :

Transition diagram:



Transition Table:

state	i/p 0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	$\cancel{\phi}$ q_0
$* q_2$	ϕ	ϕ

check for the validity of string : 0111100 and 01100.

<u>Input string : 0111100.</u>

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

$\delta(q_0, 01) = \delta(\delta, \{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$.

$= \{q_0\} \cup \phi = \{q_0\}$.

$\delta(q_0, 011) = \delta(q_0, 1) = \{q_0\}$

$\delta(q_0, 0111) = \delta(q_0, 1) = \{q_0\}$.

$\delta(q_0, 01111) = \delta(q_0, 1) = \{q_0\}$.

$\delta(q_0, 011110) = \delta(q_0, 0) = \{q_0, q_1\}$.

$\delta(q_0, 0111100) = \delta(\delta, \{q_0, q_1\}, 0)$

$= \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \{q_2\}$.

$= \{q_0, q_1, \underline{\underline{q_2}}\}$.

It is Accepted (q_2 is a final state).

Input string : 01100.

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

$\delta(q_0, 01) = \delta(\delta, \{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$

$= \{q_0\} \cup \phi = \{q_0\}$.

$\delta(q_0, 011) = \delta(q_0, 0) = \{q_0, q_1\}$.

$\delta(q_0, 0110) = \delta(\delta, \{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$

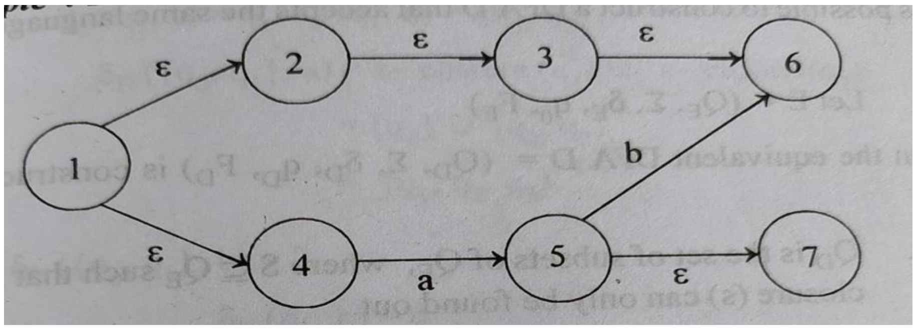
$= \{q_0, q_1\} \cup \{q_2\} = \{\cancel{q_0, q_2}\}$.

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

Accepted.

Or

12. Consider the following NFA - ϵ for an identifier. Consider the ϵ - closure of each state and find its equivalent DFA.



Sol: (i) $\hat{\delta}(1, \epsilon) = \{1, 2, 3, 4, 6\}$.

(ii) $\hat{\delta}(4, \epsilon) = \{4\}$.

(iii) $\hat{\delta}(5, \epsilon) = \{5, 7\}$.

(iv) $\hat{\delta}(4, a) = \epsilon\text{-closure}(\delta(\hat{\delta}(4, \epsilon), a)$

$= \epsilon\text{-closure}(\delta(\{4\}, a) = \epsilon\text{-closure}(\{5\})$

$= \{5, 7\}$.

(v) $\hat{\delta}(4, ab) = \epsilon\text{-closure}(\delta(\hat{\delta}(4, a), b)$

$= \epsilon\text{-closure}(\delta(\{5, 7\}, b)$

$= \epsilon\text{-closure}(\delta(5, b) \cup \delta(7, b)$

$= \epsilon\text{-closure}(\{6\} \cup \phi)$

$= \epsilon\text{-closure}(\{6\})$

$= \{6\}$.

UNIT – II

13. Construct the following grammar in CNF.

S → abSba / bAaB / bb

A → aa / aSAb

B → Aa / abb

Sol: chomsky Normal form (CNF).

1. No null and unit productions.
2. Let $G_1 = (N'_1, \{a, b\}, S, P_1^*)$ where.

   P_1 and N' are

   (i) $A \to aa$, $B \to abb$, $S \to bb$ · are added to P_1.

   (ii) $S \to abSba$; $S \to bAaB$

   $A \to aSAb$. $B \to Aa$, yield.

   $S \to C_a C_b S C_b C_a$, $S \to C_b A C_a B$,

   $A \to C_a S A C_b$, $B \to A C_a$, $C_a \to a$, $C_b \to b$.

   $N' = \{S, A, B, C_a, C_b\}$.

   (iii) P_1 consists of $S \to C_a C_b S C_b C_a$, $S \to C_b A C_a B$,

   $A \to C_a S A C_b$, $B \to A C_a$, $C_a \to a$, $C_b \to b$.
   $A \to aa$, $B \to abb$.

   $S \to C_1 C_2 S C_2 C_1$, $S \to C_2 C_3 B$, $A \to C_1 S A C_2$.

   $B \to C_3$, $C_1 \to C_a$, $C_2 \to C_b$, $C_3 \to A C_a$.

   The resulting production in P_1 added to P_2.

   $G_2 = (\{S, A, B, C_a, C_b, C_1, C_2, C_3\}, \{a, b\}, P_2, S)$.

   where P_2 consists of $S \to C_1 C_2 S C_2 C_1$, $S \to C_2 C_3 B$, $A \to C_1 S A C_2$,
   $B \to C_3$, $C_1 \to C_a$, $C_2 \to C_b$, $C_3 \to A C_a$., $A \to aa$, $B \to abb$.

   G_2 is in CNF//.

Or

14. Convert the following grammar G into Greibach Normal Form (GNF).

S → AB

A → BS / b

B → SA / a

Sol: Greibach Normal Form (GNF).

Sol

$S \rightarrow AB$

$A \rightarrow BS \mid b.$

$B \rightarrow SA \mid a.$

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 \mid b.$

$A_3 \rightarrow A_1 A_2 \mid a.$

$A_3 \rightarrow A_1 A_2 \mid a.$

$A_3 \rightarrow A_2 A_3 A_2 \mid a.$

$A_3 \rightarrow A_3 \overset{\alpha}{\boxed{A_1 A_3 A_2}} \mid \overset{\beta}{b A_3 A_2} \mid a$

$$\boxed{\begin{array}{l} A \rightarrow A\alpha \mid \beta \\ A \rightarrow \beta A' \mid \beta \\ B' \rightarrow \alpha A' \mid \alpha. \end{array}}$$

$\boxed{A_3 \rightarrow b A_3 A_2 A' \mid a A' \mid b A_3 A_2 \mid a.}$

$A_1 \rightarrow A_1 A_3 A_2 A' \mid A_1 A_3 A_2$

$A_2 \rightarrow A_3 A_1 \mid b.$

$\boxed{A_2 \rightarrow b A_3 A_2 A' A_1 \mid a A' A_1 \mid b A_3 A_2 A_1 \mid a A_1 \mid b.}$

$A_1 \rightarrow A_2 A_3.$

$\boxed{A_1 \rightarrow b A_3 A_2 A' A_1 A_3 \mid a A' A_1 A_3 \mid b A_3 A_2 A_1 A_3 \mid a A_1 A_3 \mid b A_3}$

$A_1 \to bA_3A_2A'A_1A_3A_3A_2A' \mid aA'A_1A_3A_3A_2A' \mid$
$bA_3A_2A_1A_3A_3A_2A' \mid aA_1A_3A_3A_2A' \mid$
$aA_3A_3A_2A' \mid bA_3A_2A'A_1A_2A_3A_2 \mid$
$aA'A_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2 \mid$
$aA_1A_3A_3A_2 \mid bA_3A_3A_2.$

It is an GNF //

UNIT – III

15. Design a Turing machine to accept the language L = {0^n1^n n≥1}. Draw the transition diagram.

(Also specify the instantaneous description to trace the string 0011).

Solution :

Given a finite sequence of 0's and 1's on its tape. The turing machine is designed using the following way.

(i) M replaces the leftmost 0 by x, moves right to the leftmost 1, replacing it by y.

(ii) Then M moves left to find the rightmost x, and moves one cell right to the leftmost 0 and repeats the cycle.

(iii) While searching for a 1, if a blank is encountered, then M halts without accepting.

(iv) After changing a 1 to a y, if M finds no more 0's, then M checks that no more 1's remain, accepting the string else not.

Assume the set of states $Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

$\vdash = (0, 1, x, y, B\}$

$F = \{q_4\}$

let q_0 be the initial state and at state q_0, it replaces the leftmost 0 by x, and changes it to q_1. At q_1, M searches right for 1's, skipping over 0's and y's.

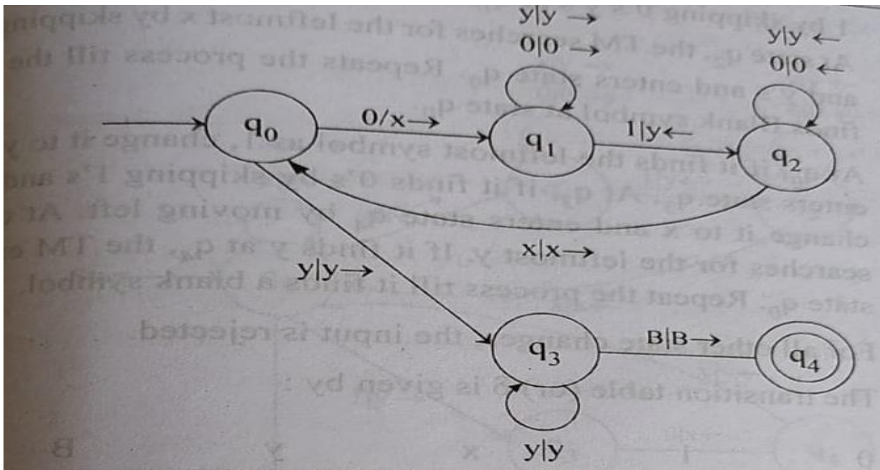


Fig. 4.15 Transition diagram for $0^n 1^n$.

$\therefore \quad M = (Q, \Sigma, \vdash, \delta, q_0, B, \{q_4\})$

where

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

$\vdash = (0, 1, x, y, B\}$

q_0 = Initial state

q_4 = Final state

δ is given in the table.

If M finds a 1, it changes it to y, entering state q_2. From q_2, it searches left for an x and moves right to change the state to q_0.

At q_0, if y is encountered, it goes to state q_3 and checks that no 1's remain. If the y's are followed by a B, state q_4 is entered and then accepted. And for all others, M rejects.

	0	1	x	y	B
q_0	(q_1, x, R)	–	–	(q_3, y, R)	–
q_1	$(q_1, 0, R)$	(q_2, y, L)	–	(q_1, y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, x, R)	(q_2, y, L)	–
q_3	–	–	–	(q_3, y, R)	(q_4, B, R)
q_4	–	–	–	–	–

Eg :

(i) $q_0 0011 \vdash x q_1 011 \vdash x 0 q_1 11 \vdash x q_2 0 y 1 \vdash q_2 x 0 y 1 \vdash$

$x q_0 0 y 1 \vdash x x q_1 y 1 \vdash x x y q_1 1 \vdash x x q_2 y y \vdash x q_2 x y y \vdash$

$x x q_0 y y \vdash x x y\, q_3 y \vdash x x y y q_3 \vdash x x y y B\, q_4$.

Accepted.

Or

16. Show that the union of two recursive language is recursive and union of two Recursively enumerable language is recursive.

Recursive languages:

We refer to a language *L* as recursive if there exists a turing machine *T* for it. In this case, the turing machine accepts every string in language *L* and rejects all strings that don't match the alphabet of *L*.

In other words, if string *S* is part of the alphabet of language *L*, then the turing machine *T* will accept it otherwise the turing machine halts without ever reaching an accepting state.

Recursively enumerable languages.

Here if there is a turing machine *T* that accepts a language *L*, the language in which an enumeration procedure exists is referred to as a recursively enumerable language.

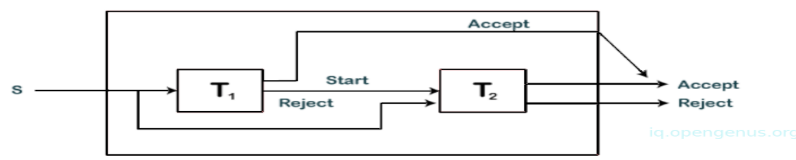
Note that some recursive languages are enumerable and some enumerable languages are recursive.

The relationship between recursive and recursively enumerable languages.

If the languages *L1* and *L2* are recursive, their union *L1 U L2* is also recursive.

Proof:

We have two turing machines *T1* and *T2* that recognize languages *L1* and *L2*. We construct a turing machine *T* as shown:



T simulates *T1* and *T* accepts input *S* is *T1* accepts it also. On the other hand,
if *T1* rejects, *T* simulates *T2* and accepts if *T2* accepts.
Both *T1* and *T2* are algorithms and therefore they will halt at some point. We conclude that *T* accepts *L1 U L2*.

UNIT – IV

17. Convert the grammar S →0S1 / A, A → 1A0 / S / ϵ into PDA that aspects the same language by empty Stack. Check whether 1001 belongs to N(M).

Sol The CFG can be first simplified by eliminating unit productions:

$S \rightarrow 0S1 \mid 1S0 \mid \epsilon$

Now will will convert this CFG to GNF:

$S \rightarrow 0SX \mid 1SY \mid \epsilon$

$X \rightarrow 1$

$Y \rightarrow 0$

The PDA can be:

$\delta(q, \epsilon, S) = \{(q, 0SX) \mid (q, 1SY) \mid (q, \epsilon)\}$

$\delta(q, \epsilon, X) = \{(q, 1)\}$

$\delta(q, \epsilon, Y) = \{(q, 0)\}$

$\delta(q, 0, 0) = \{(q, \epsilon)\}$

$\delta(q, 1, 1) = \{(q, \epsilon)\}$

---

check whether 1001 belongs to N(M).

$\delta(q, 1001, S) \vdash \delta(q, 1001, 0S1)$

$\vdash \delta(q, 001, 10S1)$

$\vdash \delta(q, \cancel{0}1, \cancel{0}10S1)$

$\vdash \delta(q, \cancel{1}, \cancel{1}0S1)$ $\quad S \rightarrow 0S1$

$\vdash \delta(q, \epsilon, S)$ $\quad S \rightarrow \epsilon$

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

Accepted.

Or

18. Construct a PDA for the language.

A push down automata is similar to deterministic finite automata except that it has a few more properties than a DFA.The data structure used for implementing a PDA is stack. A PDA has an output associated with every input. All the inputs are either pushed into a stack or just ignored. User can perform the basic push and pop operations on the stack which is use for PDA. One of the problems associated with DFAs was that could not make a count of number of characters which were given input to the machine. This problem is avoided by PDA as it uses a stack which provides us this facility also.

A Pushdown Automata (PDA) can be defined as –

M = (Q, Σ, Γ, δ, q0, Z, F) where

Q is a finite set of states

Σ is a finite set which is called the input alphabet

Γ is a finite set which is called the stack alphabet

δ is a finite subset of Q X (Σ ∪ {ε} X Γ X Q X Γ*) the transition relation.

q0 ∈ Q is the start state

Z ∈ Γ is the initial stack symbol

F ⊆ Q is the set of accepting states

Construct a PDA for language L = $\{0^n1^m2^m3^n \mid n>=1, m>=1\}$

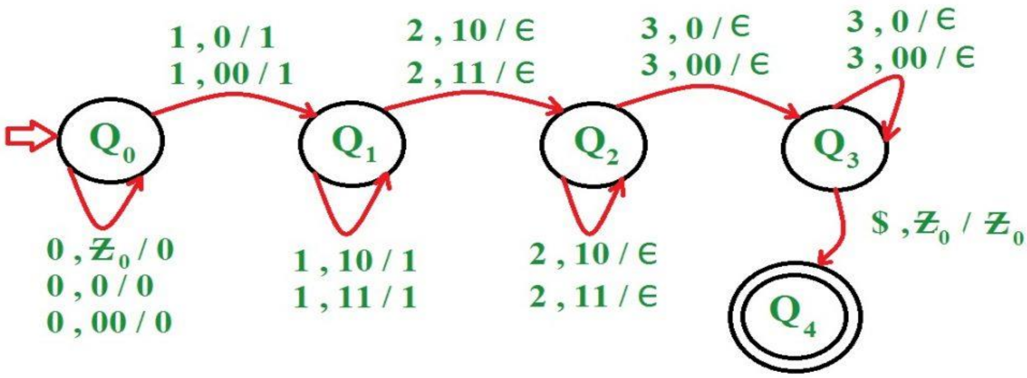
Approach used in this PDA –
First 0's are pushed into stack. Then 1's are pushed into stack.
Then for every 2 as input a 1 is popped out of stack. If some 2's are still left and top of stack is a 0 then string is not accepted by the PDA. Thereafter if 2's are finished and top of stack is a 0 then for every 3 as input equal number of 0's are popped out of stack. If string is finished and stack is empty then string is accepted by the PDA otherwise not accepted.

Step-1: On receiving 0 push it onto stack. On receiving 1, push it onto stack and goto next state

Step-2: On receiving 1 push it onto stack. On receiving 2, pop 1 from stack and goto next state

Step-3: On receiving 2 pop 1 from stack. If all the 1's have been popped out of stack and now receive 3 then pop a 0 from stack and goto next state

Step-4: On receiving 3 pop 0 from stack. If input is finished and stack is empty then goto last state and string is accepted



UNIT – V

19. Find whether the following grammar is LL(1) or not.

S → abSa / aa / aaAb

A → baAb / b

S → absa / aaAb.

A → baAb | b.

FIRST(S) = {a, a}.

FIRST (A) = {b, b}.

FIRST {a} = {a}

FIRST {b} = {b}

FOLLOW(S) = {$, a}.

FOLLOW (A) = {b}.

	a	b	$
S	S → abSA		S → ε
A		A → baAb.	

Or

20. Consider the following grammar

S → AS

S → b

A → SA

A → a

Construct SLR parsing table and process the input string.

Sol Augmented Grammar :-

$S' \rightarrow S$

$S \rightarrow AS$

$S \rightarrow b$

$A \rightarrow SA$

$A \rightarrow a$

step 1 :

Canonical collection of LR(0) items

$I_0 : S' \rightarrow \cdot S$
$S \rightarrow \cdot AS$
$S \rightarrow \cdot b$
$A \rightarrow \cdot SA$
$A \rightarrow \cdot a$

I_1: goto (I_0, S)

$I_1 : S' \rightarrow S\cdot$
$S \rightarrow A\cdot S$
$A \rightarrow \cdot SA$
$A \rightarrow \cdot a$
$S \rightarrow \cdot AS$
$S \rightarrow \cdot b$

goto (I_0, A)

$I_2 : S \rightarrow A\cdot S$
$S \rightarrow \cdot AS$
$A \rightarrow \cdot SA$
$A \rightarrow \cdot a$
$S \rightarrow \cdot AS$
$S \rightarrow \cdot b$

goto (I_0, b)

$I_3 : S \rightarrow b\cdot$

goto (I_1, A)

$I_4 : A \rightarrow SA\cdot$
$A \rightarrow \cdot SA$
$S \rightarrow \cdot AS$
$S \rightarrow \cdot b$
$A \rightarrow \cdot SA$
$A \rightarrow \cdot a$

goto (I_0, a)

$I_4 : A \rightarrow a\cdot$

goto (I_1, S)

$I_6 : A \rightarrow S\cdot A$
$A \rightarrow \cdot SA$
$A \rightarrow \cdot a$
$S \rightarrow \cdot AS$
$S \rightarrow \cdot b$

goto (I_1, S)

$I_7 : A \rightarrow SA\cdot$
$S \rightarrow A\cdot S$
$S \rightarrow \cdot AS$
$S \rightarrow \cdot b$
$A \rightarrow \cdot SA$

goto (I_1, a)

$A \rightarrow a\cdot \Rightarrow I_4$.

goto (I_1, b)

$S \rightarrow b\cdot \Rightarrow I_3$.

goto (I_2, S)

$S \rightarrow AS\cdot$
$S \rightarrow A\cdot S$
$A \rightarrow \cdot SA$
$A \rightarrow \cdot a$
$S \rightarrow \cdot AS$
$S \rightarrow \cdot b$

goto (I_2, A)

$S \rightarrow A\cdot S \Rightarrow I_2$

goto $(I_2, a) = I_4$

goto $(I_2, b) = I_3$.

goto $(I_5, S) = I_7$

goto $(I_5, A) = I_2$

goto $(I_5, a) = I_4$

goto $(I_6, A) = I_5$

goto $(I_6, S) = I_6$

goto $(I_6, a) = I_4$

goto $(I_6, b) = I_3$.

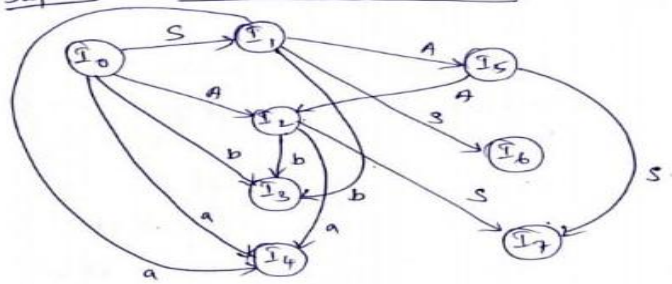
goto $(I_7, A) = I_5$

goto $(I_7, S) = I_6$

goto $(I_7, a) = I_4$

goto $(I_7, b) = I_3$.

step 2 : Design Finite Automata.



$FIRST(S) = \{b, a\}$.

$FIRST(A) = \{a, b\}$.

$FOLLOW(S) = \{\$, a, b\}$

$FOLLOW(A) = \{a, b\}$.

	Action			Goto	
	a	b	\$	S	A
0	S_4	S_3		1	2
1	S_4	S_3	Accepted	6	5
2	S_4	S_3		7	2
3	r_2	r_2	r_2		
4	r_4	r_4			
5	S_4/r_3	S_3/r_3		7	2
6	S_4	S_5		6	5
7	S_4/r_1	S_3/r_1	r_1	6	5

Stack implementation of SLR parser

Stack	Input Buffer	Action
$0	abab$	shift 4
$0a4	bab$	reduce A→a
$0A2	bab$	shift 3
$0A2b3	ab$	reduce S→b
$0A2S7	ab$	reduce S→AS
$0S1	ab$	shift 4
$0S1a4	b$	reduce A→a
$0S1A5	b$	reduce A→SA
$0A2	b$	shift 3
$0A2b3	$	reduce S→b
$0A2S7	$	reduce S→AS
$ ~~S~~ 0S1	$	reduce Accepted