

AUTOMATA THEORY

Finite State Machine	2
Finite State Models	9
Structure of Sequential Machines	36
Finite Automation	41
Closure Properties of Regular Sets	67
Context Free Grammars	81
Pushdown Automation	101
Closure Properties of CFLs	109
Turing Machine	112

NOTE:

WBUT course structure and syllabus of 4th semester has been changed from 2012. Formal Language & Automata Theory (4th Sem, CS) has been redesigned. Taking special care of this matter we are providing the relevant WBUT solutions of old Formal Language & Automata Theory, from 2006 to 2011 along with the complete solutions of new university papers also, so that students can get an idea about university questions patterns.

FINITE STATE MACHINE

Multiple Choice Type Questions

1. The basic limitation of Finite State Machine of that [WBUT 2007, 2010]
a) it cannot remember arbitrary large amount of information
b) it cannot recognize grammars that are regular
c) it sometimes recognize grammars that are not regular
d) all of these
- Answer: (a)
2. A shift register is a [WBUT 2010]
a) Mealy M/c b) Moore M/c c) Turing M/c
d) All of these
- Answer: (d)
3. FSM can recognize [WBUT 2011]
a) a grammar dependent on characteristic of FSM b) on CFG
c) any unambiguous grammar d) only regular grammar
- Answer: (d)
4. Which is true of the following? [WBUT 2013]
a) Merger graph is directed graph
c) Both are directed
- Answer: (b)
5. Compatible pairs are obtained from [WBUT 2014, 2015, 2017]
a) Merger Graph
c) Testing Table
- Answer: (a)
6. A finite automata recognizes [WBUT 2018]
a) Any Language
c) Context Free Language
- Answer: (d)
7. Maximum number of states of a DFA converted from a NFA with n states is [WBUT 2018]
a) n b) n^2 c) 2^n d) None of these
- Answer: (c)
8. Regular Expression is accepted by [WBUT 2018]
a) Finite Automata
c) Turing Machine
d) All of these
- Answer: (d)

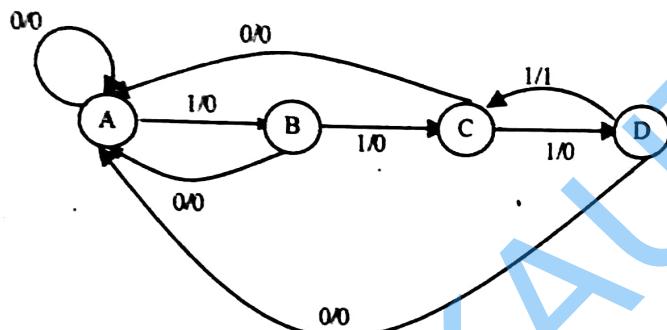
Short Answer Type Questions

1. A long sequence of pulses enters a two-input, two-output synchronous sequential circuit, which is required to produce an output pulse $Z = 1$ whenever the sequence 1111 occurs. Overlapping sequences are accepted; for example, if the input is 01011111....., the required output is 00000011.....

- Draw a state diagram.
- Select the assignment and show the excitation and output tables.

Answer:

[WBUT 2004, 2008]



Present State	Next State, Z	
	Input $x = 0$	Input $x = 1$
A	A, 0	B, 0
B	A, 0	C, 0
C	A, 0	D, 0
D	A, 0	C, 1

y_1y_2	y_1y_2		Z	
	$x=0$	$x=1$	$x=0$	$x=1$
$A \rightarrow 00$	00	01	0	0
$B \rightarrow 01$	00	11	0	0
$C \rightarrow 11$	00	10	0	0
$D \rightarrow 10$	00	11	0	1

2. What are the limitations of sequential circuit?

[WBUT 2009]

Answer:

A sequential circuit can only represent a finite-state machine. In such a machine, an arbitrarily long sequence of input symbols is bound to create a periodic output with a period not more than the number of states in the machine. Thus, computations which needs to remember an arbitrary number of past input symbols, for example in recognizing $a^n b^n$, $n > 0$, cannot be implemented by sequential machines. By the same reasoning, a sequential machine cannot multiply two arbitrary length binary strings.

3. Define Deterministic Finite Automata. What do you mean by NFA with -moves.

[WBUT 2018]

POPULAR PUBLICATIONS

Answer:

Definition of DFA:

DFA is a 5-tuple.

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

$Q \rightarrow$ finite set of states

$\Sigma \rightarrow$ finite set of alphabets

$\delta \rightarrow$ transition function

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

$q_0 \rightarrow$ start state

$F \rightarrow$ sets of final states

NFA with λ -moves:

Let Q be finite set.

Let Σ be a finite set of symbols.

$\therefore \delta$ be transition function defined as,

$$Q \times \Sigma \cup \{\lambda\} \rightarrow 2^Q$$

$q_0 \rightarrow$ be a state in Q

and A be a subset of Q .

$\therefore A$ is set of accepting states.

So, NFA with λ -moves, is a 5-tuple

$$\{Q, \Sigma, q_0, \delta, A\}$$

\therefore Basically, any NFA is also a NFA with λ -move

Long Answer Type Questions

1. Draw the Merger graph, Merger Table, Compatibility graph and then minimize the following:
[WBUT 2011, 2016]

Present State	Next State, o / p			
	i/p = 0	i/p = 1	i/p = 2	i/p = 3
A	-	C, 1	E, 1	B, 1
B	E, 0	-	-	-
C	F, 0	F, 1	-	-
D	-	-	B, 1	-
E	-	F, 0	A, 0	D, 1
F	C, 0	-	B, 0	C, 1

Answer:

Step 1: Draw the arc of compatible pair. Compatible pair means that two present state entries of the next state should not conflict.

Here AB, BC, EF, etc are non-conflicting pair.

FORMAL LANGUAGE AND AUTOMATA THEORY

On other hands DE is a conflicting pair, because output of I_3 is conflicting i.e. B consist the next state output of I_3 is 1 whereas E contain 0. Similarly CE, DF are also conflicting so these are non-compatible pair.

Note: now the merger graph displays all possible pair of state and their implied pair, and since a pair of state is compatible only if its implied pair is.

Step 2: Check and determine whether the implied pair are indeed compatibles. A pair $(S_p S_q)$ is incompatible if no arc is drawn between vertices S_p and S_q . In such a case, if $(S_p S_q)$ is written space of an interrupted arc, the entry $(S_p S_q)$ is crossed off, and the, corresponding arc is ignored.

Here the condition for (BF) to be compatible is that (CE) be compatible, but since there is no arc drawn between C and E, (CE) is incompatible and the arc between B and F is ignored. Thus states B and F are incompatible.

Next, it is necessary to check whether the incompatibility of (BF) does not invalidate any other implied pair, that is, if (BF) is not written in the space of another interrupted arc, and so on. The interrupted arcs which remain in graph, after all the implied pairs have been verified to be compatible are regarded as solid ones.

The merger graph of Machine M reveals the existence of compatible pairs:

$(AB), (AC), (AD), (BC), (BD), (BE), (CD), (CF), (EF)$

In order to find a minimal set of compatibles, which covers original machine and can be used as a basis for the construction of the minimal machine, it is often useful to find the set of maximal compatibles. Recall that a compatible is maximal if it is not contained in any other compatible. In terms of the merger graph, we are looking for complete polygons which are not contained within any higher-order polygons. [A *complete polygon* is one in which all possible $\binom{n-3}{2}$ $n/2$ diagonals exist, where n is the number of sides in the polygon.] Since the states covered by a complete polygon are all pair wise compatible they constitute a compatible; and if the polygon is not contained; higher-order complete polygon, they constitute a maximal compatible.

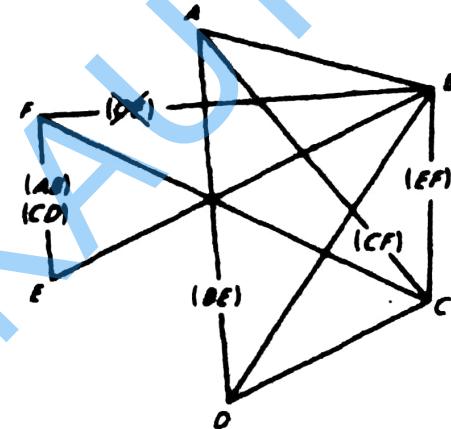
In the merger graph the set of highest-order polygons are the $(ABCD)$ and the arcs (CF) , (BE) , and (EF) . Generally, after a complete polygon of order n has been found, all polygons of order $n-1$ contained in it can be ignored. Consequently, the triangles (ABC) , etc., are not considered. Thus the following set of maximal compatibles for machine M is:

$\{(ABCD), (BE), (CF), (EF)\}$

2. a) A long sequence of pulses enters a synchronous sequential circuit, which is required to produce an output pulse $z=1$ whenever the sequence 1001 occurs. Overlapping sequences are accepted.

i) Draw a state diagram

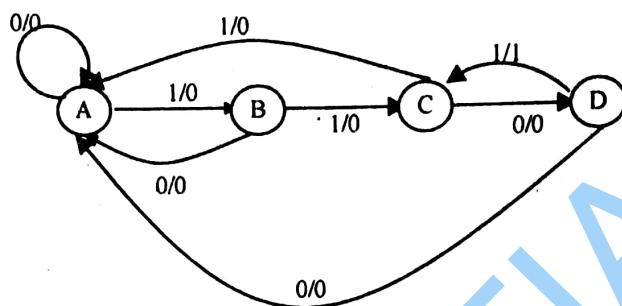
ii) Select an assignment and show the excitation and output tables. [WBUT 2015]



POPULAR PUBLICATIONS

Answer:

a) i)



ii)

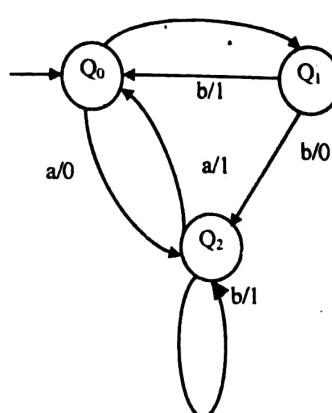
Present State	Next State, Z	
	Input x = 0	Input x = 1
A	A, 0	B, 0
B	A, 0	C, 0
C	D, 0	A, 0
D	A, 0	C, 1

y_1y_2	$Y_1 Y_2$		Z	
	x=0	x=1	x=0	x=1
$A \rightarrow 00$	00	01	0	0
$B \rightarrow 01$	00	11	0	0
$C \rightarrow 11$	10	00	0	0
$D \rightarrow 10$	00	11	0	1

3. Convert the following Mealy machines into Moore machine.

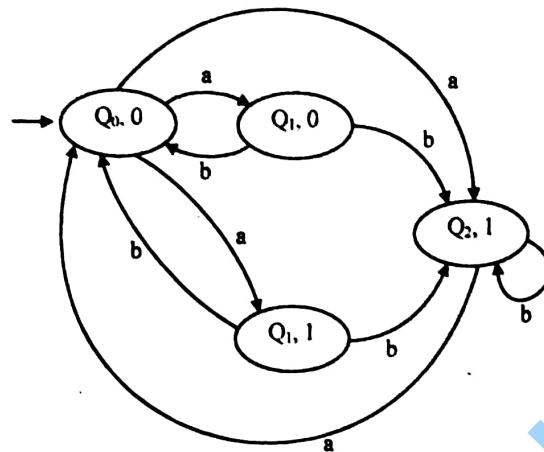
a/0

[WBUT 2015]



FORMAL LANGUAGE AND AUTOMATA THEORY

Answer:



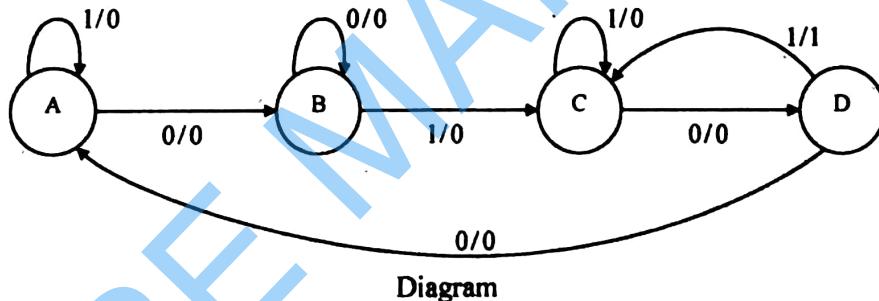
- 4. Draw sequence detector diagram, table, k-map and circuit for the sequence 0101 which generates a 1 as output after every 0101 sequence. Consider overlapping of sequences.** [WBUT 2018]

Answer:

Sequence detector:

i/p: 0 1 0 1 0 1 0 1

o/p: 0 0 0 1 0 1 0 1



Present State	Next State	
	I/P = 0	I/P = 1
A	B, 0	A, 0
B	B, 0	C, 0
C	D, 0	C, 0
D	A, 0	C, 1

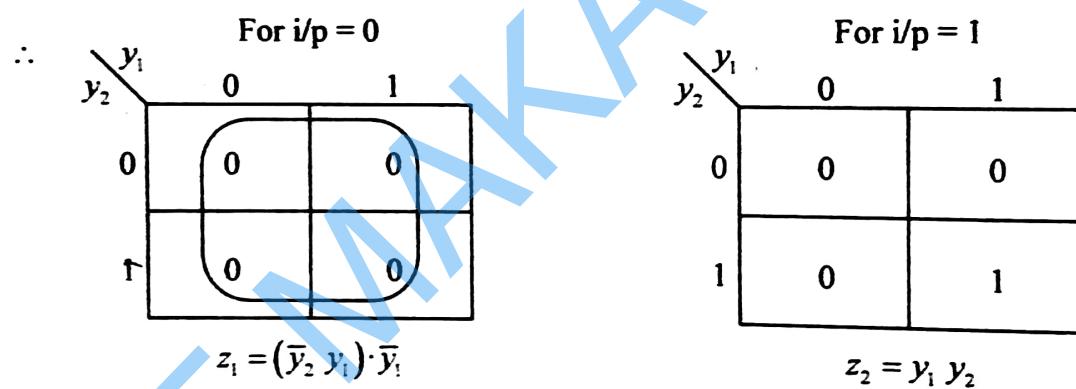
Table

POPULAR PUBLICATIONS

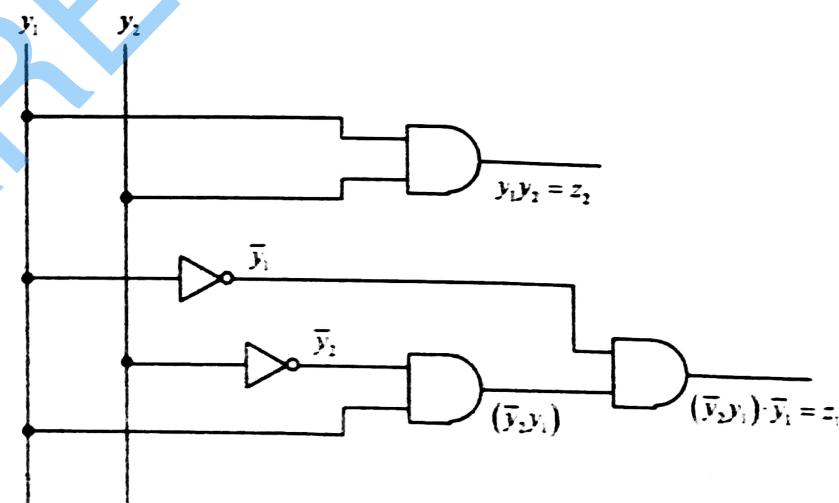
∴ K-map:

Let $A \rightarrow 00, B \rightarrow 01, C \rightarrow 10, D \rightarrow 11$

Present State		Next State			
		i/p = 0		i/p = 1	
	$y_1 y_2$	$y_1 y_2$	z_1	$y_1 y_2$	z_2
A	0 0	0 1,	0	0 0,	0
B	0 1	0 1,	0	1 0,	0
C	1 0	1 1,	0	1 0,	0
D	1 1	0 1,	0	1 0,	1



Circuit:



FINITE STATE MODELS

Multiple Choice Type Questions

1. Choose the correct statements:

- a) moore and mealy machine are FSM with output capability
- b) any given Moore machine has an equivalent Mealy machine
- c) any given Mealy machine has an equivalent Moore machine
- d) moore machine is not an FSM

Answer: (a), (b) and (c)

[WBUT 2006, 2012]

2. Which is true of the following?

- a) Merger graph is a directed graph
- b) Compatible graph is a directed graph
- c) both are directed
- d) None of these

Answer: (b)

[WBUT 2009]

3. Merger table is a substitute of

- a) Merger graph
- b) Compatible graph
- c) Minimized machine
- d) Finite state machine

Answer: (a)

[WBUT 2009]

4. The loop-free testing graph indicates that

- a) the machine has finite memory
- b) the machine has non-finite memory
- c) the machine has finite states
- d) the machine has non-finite states

Answer: (a)

[WBUT 2010]

5. Input sequence of an information lossless machine can be determined from the knowledge of

[WBUT 2010]

- a) only output sequence
- b) output sequence and initial state
- c) output sequence, initial state and final state
- d) initial state

Answer: (a)

6. Moore machine output depends on

[WBUT 2011]

- a) input
- b) present state
- c) input and present state
- d) none of these

Answer: (c)

7. In Moore machine, output is associated with

[WBUT 2017]

- a) present state only
- b) next state only
- c) present state and input
- d) none of these

Answer: (a)

POPULAR PUBLICATIONS

8. Number of vertices of a Merger graph is
 a) The number of state of the machine
 b) Number of compatible pairs
 c) Number of states combinations

Answer: (a)

[WBUT 2018]

- b) Number of compatible pairs
 d) None of the above

9. Number of vertices of a compatible graph is
 a) the number of states of the machine
 b) number of compatible pairs
 c) number of states combinations

Answer: (b)

[WBUT 2018]

Short Answer Type Questions

1.

PS	NS, Z		
	I ₁	I ₂	I ₃
A	C, 0	E, 1	-
B	C, 0	E, -	-
C	B, -	C, 0	A, -
D	B, 0	C, -	E, -
E	-	E, 0	A, -

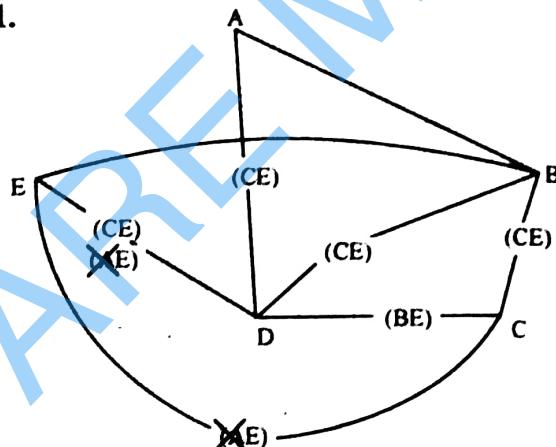
Table-1

For the incompletely specified machine shown in table-1, find a minimum state reduced machine containing the original one. [WBUT 2004, 2008, 2012]

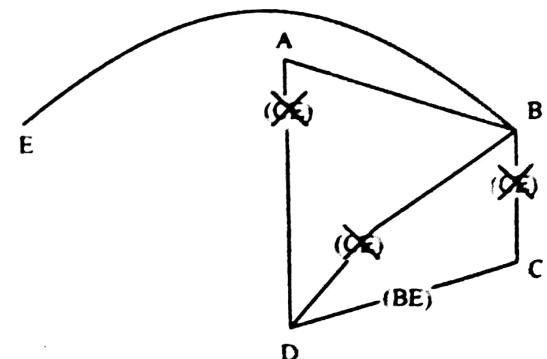
Answer:

Compatible pairs = AB, AD, BC, BD, BE, CD, CE, DE

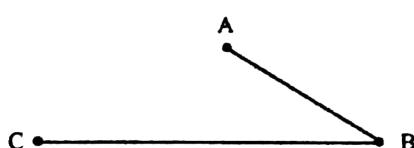
1.



2.



3.

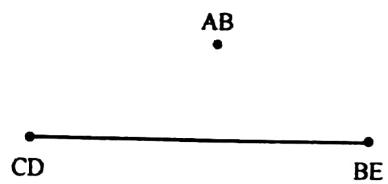


Final merger graph

$$D \xrightarrow{(BE)} C \xrightarrow{(BE)} B$$

FORMAL LANGUAGE AND AUTOMATA THEORY

Compatibility Graph



Merger table

B	✓		
C	x	CE	
D	CE	CE	BE
E	x	✓	CE, AE

A B C D

B	✓		
C	x		CE
D	CE	CE	BE
E	x	✓	x

A B C D

Next step merger table.

B	✓		
C	x	x	
D	x	x	BE
E	x	✓	x

A B C D

Final merger table

2. Minimize the following machine to standard form:

[WBUT 2006, 2008]

PS	NS, z	
	X = 0	X = 1
A	E, 0	C, 0
B	C, 0	A, 0
C	B, 0	G, 0
D	G, 0	A, 0
E	F, 1	B, 0
F	E, 0	D, 0
G	D, 0	G, 0

Answer:

Let's do the partitioning process.

POPULAR PUBLICATIONS

Partitions	Reason
$P_0 = (A B C D E F G)$	
$P_1 = (A B C D F G) (E)$	E has $o/p = 1$ on $i/p = 0$
$P_2 = (B C D G) (A F) (E)$	$(A F)$ goes to E on $i/p = 0$
$P_3 = (B D) (C G) (A F) (E)$	$(B D)$ goes to A on $i/p = 1$
$P_4 = (B D) (C G) (A) (F) (E)$	A goes to C but F goes to D on $i/p = 1$
$P_5 = (B D) (C G) (A) (E) (F)$	Final partition.

Let $(A) \Rightarrow \alpha$
 $(B D) \Rightarrow \beta$
 $(C G) \Rightarrow \gamma$
 $(F) \Rightarrow \delta$
 $(E) \Rightarrow \theta$

Reduced machine		
PS	NS	Z
α	$\delta, 0$	$\gamma, 0$
β	$\gamma, 0$	$\alpha, 0$
γ	$\beta, 0$	$\gamma, 0$
δ	$\theta, 1$	$\beta, 0$
θ	$\delta, 0$	$\gamma, 0$

3. What do you mean by Inverse machine? Write the definition of a lossless machine. [WBUT 2008]

Answer:

An *inverse* M' is a machine which, when excited by the output sequence of a machine M , produces (as its output) the input sequence to M , after at most a finite delay. Evidently, a deterministic inverse can be constructed only if M is lossless, and it can be constructed so that it produces M 's input sequence after just a finite delay if and only if M is lossless of finite order.

A machine is to be (*information*) lossless if the knowledge of the initial state, the output sequence, and the final state is sufficient to determine uniquely the input sequence.

4. What is the basic difference between Mealy machine and Moore machine?

[WBUT 2009]

OR,

Define and compare Moore and Mealy machines.

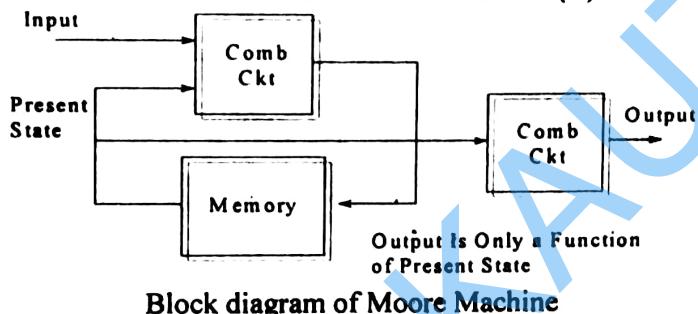
[WBUT 2015]

Answer:

Moore Machine

The limitation of FA is that its output is limited to a binary signal "accept/don't accept". Models in which the output is chosen from some other alphabet and in which the output is associated with the state is called Moore machine.

A Moore machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where Q , Σ , δ and q_0 are as in a DFA. Δ is the output alphabet and λ is a mapping from Q to Δ giving the output associated with each state. Any Moore machine gives output $\lambda(q_0)$ in response to input ϵ . The DFA may be viewed as a special case of a Moore machine, where the output alphabet is $\{0, 1\}$ and state q is "accepting" if and only if $\lambda(q) = 1$.



Mealy Machine

A Mealy machine is a sequential machine in which output is associated with each state transition.

The output of the vending machine example, presented in class, is that of a Mealy machine.

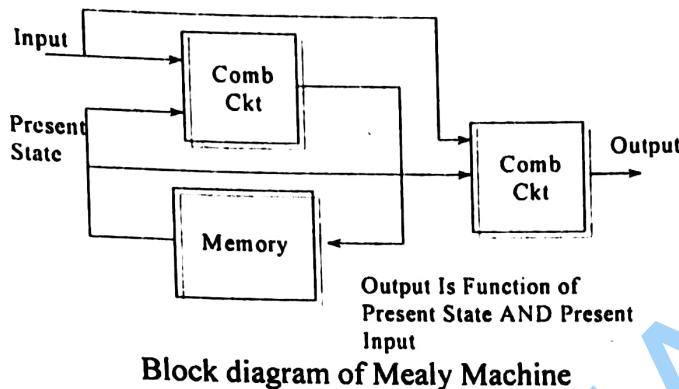
A Mealy machine is a six-tuple $M = (Q, \Sigma, \Delta, \lambda, \delta, q_0)$, where Q is a finite non-empty set of states, Σ is the input alphabet, Δ is the output alphabet, $\lambda : Q \times \Sigma \rightarrow \Delta$ is the state transition function, and $\delta : Q \times \Sigma \rightarrow Q$ is the output function.

The λ function gives the output symbol associated with the transition from state q on input symbol α , i.e.,

$$\lambda(q, \alpha) = z \in \Delta$$

Notice the six-tuple does not have a set of final states. That is because sequential machines do not have final states.

The length of the output string is equal to the length of the input string. The state transition diagram for a Mealy machine M is almost identical to the diagram for finite automata; the output symbol is denoted on the edge by appending to the input symbol, a slash ("/") followed by the output symbol. Sequential machines are used to design sequential circuits.



OR Part also:

Moore vs. Mealy Machine

Moore Machine: Output only depends on present state

Mealy Machine: Output depends on both the present state and the present input.

- 5. In response to an unknown input sequence, the machine given below produces the output sequence 1110000010. Find the input sequence to the machine if it is known that its initial state is A and final state is F.** [WBUT 2009]

PS	NS, z	
	$x = 0$	$x = 1$
A	B, 1	C, 0
B	D, 1	B, 1
C	E, 1	B, 0
D	A, 0	E, 0
E	F, 0	D, 1
F	D, 0	A, 1

Answer:

This problem is best solved by tracing backwards from state F. We see that the only F, 0 entry is for E and $x = 0$. Hence the second-last state is E. Proceeding similarly. We get

Output: 1 1 1 0 0 0 0 0 1 0

Possible states F A B D E F D A C E F

A B E

B C

But we know that the first state is A and since the first output is 1, the next state is B. Also, since the next output is 1, the next state is B or D. But we have found that this state is B or E. Hence, the state is B. So, the state transition are:

$A \rightarrow B \rightarrow B \rightarrow D \rightarrow E \rightarrow F \rightarrow D \rightarrow A \rightarrow C \rightarrow E \rightarrow F$

Hence, the input string is: 0101000110

- 6. Is the following machine information lossless? If yes, find the order of losslessness.** [WBUT 2009]

PS	NS, z	
	$x = 0$	$x = 1$
A	A, 0	B, 0
B	C, 0	D, 0

FORMAL LANGUAGE AND AUTOMATA THEORY

C	D, 1	C, 1
D	B, 1	A, 1

Answer:

The first step of this testing procedure is to check each row of the transition table, whether there is two identical next state entries associated with the same output symbol. From the above table it is clear that, there is no identical next state entries.

Now we have to construct the output successor table.

Testing table for Machine-M

PS	$z = 0$	$z = 1$
A	(AB)	—
B	(CD)	—
C	—	(CD)
D	—	(AB)
AB	(AB) (CD)	—
CD	—	(AB) (CD)
AC	—	—
AD	—	—
BC	—	—
BD	—	—

The lower part of the testing table is constructed from the compatible pair of states of the upper part of the table. In the lower part of the table all the implied pairs have been taken as a row heading.

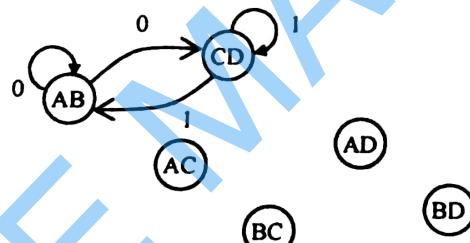


Fig: Testing graph for machine M

From the lower part of the testing table it is clear that, there is no compatible pair with repeated entries. This is necessary and sufficient for a machine to be lossless.

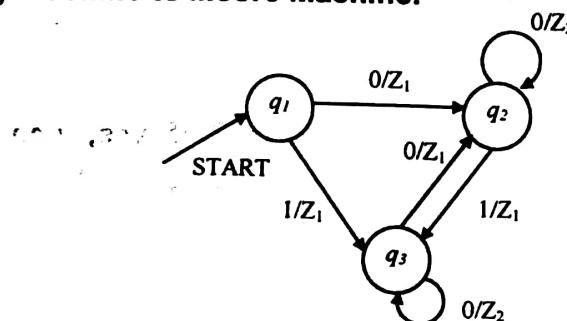
So clearly the given machine is lossless.

In the above testing graph the length of the largest path is $l = 2$.

Hence the order of losslessness, $\mu = l + 2 = 2 + 2 = 4$

7. Convert Mealy Machine to Moore Machine.

[WBUT 2011, 2012, 2016]



FAT-15

Answer:

According to the above diagram the Melay machine is transition diagram will be,

PS	NS, z	
	i/p = 0	i/p = 1
q ₁	{q ₂ , z ₁ }	{q ₃ , z ₁ }
q ₂	{q ₁ , z ₂ }	{q ₃ , z ₁ }
q ₃	{q ₂ , z ₂ }	{q ₁ , z ₁ }



PS	NS, z	
	i/p = 0	i/p = 1
q ₁	{q ₂ , z ₁ }	{q ₃ , z ₁ }
q ₂₀	{q ₂₀ , z ₂ }	{q ₃ , z ₁ }
q ₂₁	{q ₂₀ , z ₂ }	{q ₃ , z ₁ }
q ₃₀	{q ₃₀ , z ₂ }	{q ₂₁ , z ₁ }
q ₃₁	{q ₃₀ , z ₂ }	{q ₂₁ , z ₁ }

The corresponding Moore Machine will be,

Present State	NS		Output
	i/p = 0	i/p = 1	
q ₁	q ₂	q ₃	z ₁
q ₂₀	q ₂	q ₃	z ₂
q ₂₁	q ₂	q ₃	z ₁
q ₃₀	q ₃	q ₂	z ₂
q ₃₁	q ₃	q ₂	z ₁

8. Design a two input two output sequence detector which generates an output '1' every time the sequence 1001 is detected. And for all other cases output '0' is generated. Overlapping sequences are also counted.

[Denote State Graph, State Table and perform State assignment]

[WBUT 2014, 2015]

Answer:

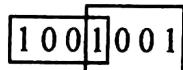
Before designing this circuit some clarifications regarding sequence detector is needed. Let's assign the input string as 1001001.

We have to design the circuit such a way that it will take one input at a time. The input can be either '0' or '1' (two types of input). The output will be also two types, either '0' or '1'. The circuit can store (memorize) the input string up to four clock pulses (t_1 to t_4).

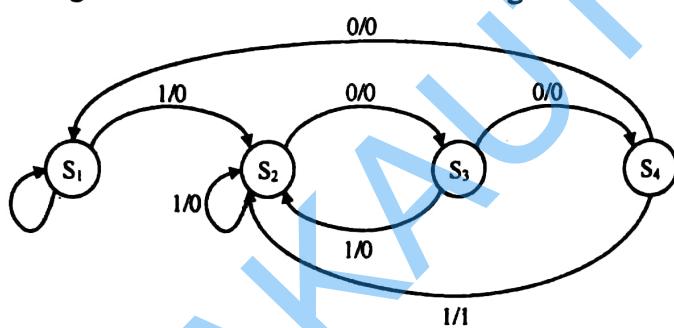
If the input string is placed according to clock pulses, the output is as follows:

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
I/P	1	0	0	1	0	0	1
O/P	0	0	0	1	0	0	1

The first input at t_1 is 1 and as there is no input from t_{i-3} to t_1 , the input sequence does not equal to 1001. So the output will be 0. Similar cases occur for the inputs upto t_3 . However at time t_4 the input from t_{i-3} to t_4 becomes 1001, hence the output '1' is produced at time t_4 . At time t_5 and t_6 the input string from t_{i-3} to t_i are 0010 and 0100, respectively. Therefore, the output '0' is produced. At t_7 clock pulse the input string is 1001, hence output '1' is produced. As the output '1' at t_4 is overlapped from t_1 to t_4 and from t_4 to t_7 , this is called overlapping condition.



In this case the state diagram is to be drawn first according the following process:



In state S_1 , input may be either '0' or '1'. If given input '0', there is no chance to get 1001. Hence it loops on S_1 with output '0'. If the input is '1', there is a chance to get 1001 then machine moves to S_2 producing output '0'.

In S_2 again the input may be either '0' or '1'. If it is '1', the input becomes 11. There is no chance to get 1001 by taking previous inputs. But again there is a chance to get 1001 by considering the given input '1'. Hence it will be in state S_2 . (If it goes to S_1 then there will be loss of clock pulse, i.e., from S_1 by taking input '1', again it has to come to S_2 , i.e., one extra input, i.e., one clock pulse is needed and hence the output will not be in right pattern). If the input is '0', the input becomes 10 – by considering the previous input and there is chance to get 1001, so it will come to S_3 .

In S_3 , if it gets input '0', the input becomes 100 by considering the previous input and it has a chance to get 1001, so it can shift to S_4 . But if it gets '0', it has no chance to get 1001 considering the previous input, but there is a chance to get 1001 by considering the given input '1'. So it will shift to S_2 as we know by getting '1' in S_1 the machine comes to S_2 .

In S_4 if it gets '0', the input will become 1000, but it does not match with 1001. Therefore, it has to start from the beginning i.e., S_1 . As overlapping condition is

POPULAR PUBLICATIONS

accepted. hence from the last '1' of 1001 if it gets 001 only, it will give an output '1'. Therefore, it will go to S_2 .

State table:

A state table can easily be constructed from the above state diagram.

PS	NS, O/P	
	X	0
S_1	$S_1, 0$	$S_2, 0$
S_2	$S_3, 0$	$S_2, 0$
S_3	$S_4, 0$	$S_2, 0$
S_4	$S_1, 0$	$S_2, 1$

State assignment:

The states must be assigned to some binary numbers to make a digital circuit. This is called state assignment. As the number of states is four, only two digit number is sufficient to represent the four states ($2^2 = 4$).

Let's assign

S_1 to 00

S_2 to 01,

S_3 to 11,

S_4 to 10.

After doing this state assignment the state table becomes.

PS(y_1y_2)	NS(Y_1Y_2)		O/P(z)	
	X	0	1	0
00	00	01	0	0
01	11	01	0	0
11	10	01	0	0
10	00	01	0	1

The digital function can easily be derived from this state assignment table:

$Y_1 = X'y_2$	Y_1		Y_2		z			
	X	0	X	0	1	X	0	1
$Y_1 = X'y_2$	y_1y_2	0	y_1y_2	0	1	y_1y_2	0	0
$Y_2 = X + y'_1y_2$	00	0	0	00	0	1	00	0
$z = Xy_1y'_2$	01	1	0	01	1	1	01	0
	11	1	0	11	0	1	11	0
	10	0	0	10	0	1	10	0

9. Minimize the machine using equivalent partitioning

[WBUT 2015]

Present state	Next state, output	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

Answer:

$$P_0 = (A, B, C, D, E, F)$$

$$P_1 = (A, C, E) (B, D, F)$$

$$P_2 = (A, C, E) (B, D) (F)$$

$$P_3 = (A, C) (E) (B, D) (F)$$

Say: $(AC) \rightarrow \alpha$, $E \rightarrow \beta$, $(BD) \rightarrow \lambda$, $F \rightarrow \delta$

The standard form of the reduced machine is:

PS	NS	
	x = 0	x = 1
α	$\beta, 0$	$\lambda, 1$
β	$\alpha, 0$	$\delta, 1$
λ	$\delta, 0$	$\lambda, 1$
δ	$\lambda, 0$	$\alpha, 1$

10. Design a two-input two-output sequence detector which generates an output '1' every time the sequence 1010 is detected. And for all other cases output '0' is generated. Overlapping sequences are also counted. Draw only state table and state diagram.

[WBUT 2016]

Answer:

The input string 1010100 is placed according to the clock pulses, and it looks like the following

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
I/P	1	0	1	0	1	0	0
O/P	0	0	1	0	1	0	

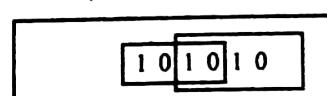


Fig: Overlapping Sequence

And the output becomes as given earlier. Overlapping portion is 10, as shown in above figure. The state diagram is in following:

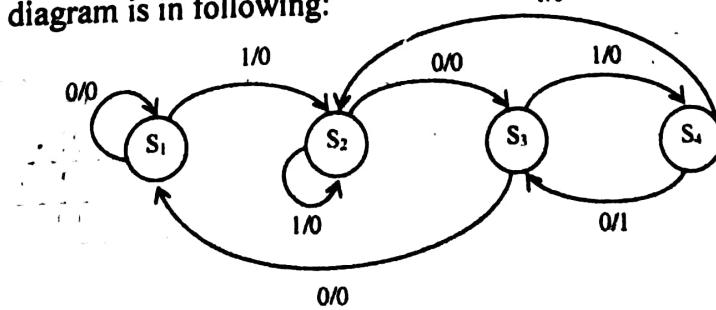


Fig: State diagram

POPULAR PUBLICATIONS

State Table: From the previous state diagram, a state table as follows can easily be constructed.

Present State	Next State, O/P	
	X=0	=1
S ₁	S ₁ , 0	S ₂ , 0
S ₂	S ₃ , 0	S ₂ , 0
S ₃	S ₁ , 0	S ₄ , 0
S ₄	S ₃ , 0	S ₂ , 0

11. Find the string which is applied on state 'D' producing output string 10011110 and final state 'D' for the machine given below:

PS	NS, o/p	
	x = 0	x = 1
A	A, 1	C, 1
B	E, 0	B, 1
C	D, 0	A, 0
D	C, 0	B, 0
E	B, 1	A, 0

[WBUT 2016]

Answer:

First we need to prove the machine is information lossless. For this we need to construct a testing table for information lossless. If the machine is information lossless then and only a single input string can be found for a single beginning state and single final state.

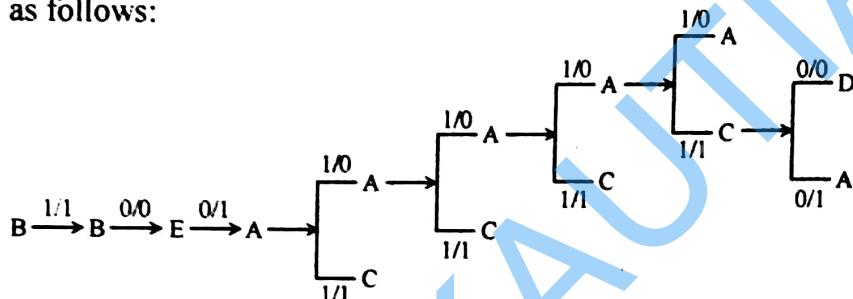
The testing table for information lossless:

PS	NS	
	z=0	z=1
A	-	(AC)
B	E	B
C	(AD)	-
D	(BC)	-
E	A	B
AC	-	-
AD	-	-
BC	(AE)(DE)	-
AE	-	(AB)(BC)
DE	(AB)(AC)	-
AB	-	(AB)(BC)

The testing table does not contain any repeated entry. The machine is information lossless machine. The output successor table for the given machine:

PS	NS, i/p	
	$z=0$	$z=1$
A	-	(A, 0), C, 1)
B	E, 0	B, 1
C	(D, 0), (A, 1)	-
D	(C, 0), (B, 1)	-
E	A, 1	B, 0

Transition is as follows:



Beginning state B and final state D is obtained from one path with input string 10100010.

12. Design a two-input two-output sequence detector which generates an output 1 every time the sequence 1101 is detected. And for all other cases output 0 is generated. Overlapping sequences are also counted. [WBUT 2017]

Answer:

The sequence is 1101. We have to start from S_1 . If we get input 0, then there is no chance, so it is confined in S_1 producing output 0. If we get input 1 then there is a chance to get 1101, so the control moves to S_2 producing output 0 (as we have not got 1101 as input). In S_2 if we get 0 then there is a chance to get 1101 considering the last 1, so the control will be confined in S_2 producing output 0. In S_2 if we get 1 then there is a chance to get 1101 so the control moves to S_3 producing output 0.

In S_3 , if we get 0, there is a chance to achieve the string 1101, so the control moves to S_4 producing output 0. But in S_3 if we get 1 then there is no chance to get the string 1101, so in this case we have to start again from the beginning i.e., S_1 . In S_4 also, if we get 0 then there is no chance of achieving the input 1101 so the control moves to the beginning producing output 0, but if we get 1 then the string 1101 is achieved, thus producing the output 1. As overlapping sequence is also accepted, the control moves to S_2 , so that by getting 101, the sequence detector can produce 1.



The state diagram is given below –

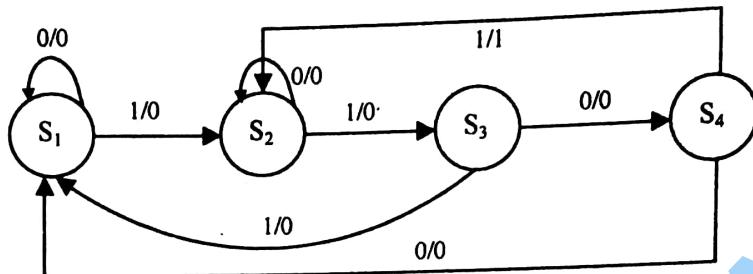


Figure 1

The state table for the sequence detector,

PS	NS		O/P
	X	0	1
S ₁	S ₁ , 0	S ₂ , 0	
S ₂	S ₃ , 0	S ₂ , 0	
S ₃	S ₄ , 0	S ₁ , 0	
S ₄	S ₁ , 0	S ₂ , 1	

Long Answer Type Questions

1. a) In response to an unknown input sequence, the machine of table-4 produces the output sequence 1 1 1 0 0 0 0 0 1 0.

Find the input sequence to the machine if it is known that its initial state is A and final state is F.

PS	NS, Z	
	x = 0	x = 1
A	B, 1	C, 0
B	D, 1	B, 1
D	E, 1	B, 0
F	A, 0	E, 0
F	F, 0	D, 1
F	D, 0	A, 1

Table-4

b) Find the minimal inverse machine of the machine given in table-5. [WBUT 2004]

PS	NS, Z	
	x = 0	x = 1
A	C, 0	D, 1
B	D, 0	C, 1
C	A, 0	B, 0
D	C, 1	D, 1

Table-5

OR,

Show that the following FSM is information lossless of finite order:

PS	NS, Z	
	x = 0	x = 1
A	C, 0	D, 1
B	D, 0	C, 1
C	A, 0	B, 0
D	C, 1	D, 1

Table-5

Also find its order of information losslessness.

[WBUT 2008]

Answer:

- a) We use the backward sequencing to get

A	B	B	D	E	F	D	A	C	E	F
1	1	1	0	0	0	0	0	0	1	

Hence, the input sequence is

0 1 0 1 0 0 0 1 0 0

- b)

D	C, 1	D, 1
---	------	------

Machine-M

This machine is loss less of order 3. If we know the initial state and the values of three successive outputs, which are produced due to transition from this initial state, then we can determine the first input to the machine.

Now the present states of the inverse machine can be defined by a set of triples, which is denoted by $(S(t), Z(t+1), Z(t+2))$ where $S(t)$ is the possible states of the original machine M and $Z(t+1), Z(t+2)$ are 2 outputs.

For the above machine we have the following triples:

$$\begin{array}{llll} (A,0,0) & (B,0,1) & (C,0,0) & (D,1,0) \\ (A,1,1) & (B,1,0) & (C,0,1) & (D,1,1) \end{array}$$

The rest of the triples are not defined.

Thus the inverse machine has present states defined by the eight triples shown above.

Now the next state of the inverse machine can also be defined by triples.

The first member of this triple is the state to which the machine M will go consuming the first input symbol.

The second member is nothing but the third member of the present state of the inverse machine and finally the third member is the present output of M.

The state table of the corresponding inverse machine is shown below:

Machine- M^i

PS	NS, X	
	z = 0	z = 1
(A,0,0)	(C,0,0),0	(C,0,1),0
(A,1,1)	(D,1,0),1	(D,1,1),1
(B,0,1)	(D,1,0),0	(D,1,1),0
(B,1,0)	(C,0,0),1	(C,0,1),1

POPULAR PUBLICATIONS

(C,0,0)	(A,0,0),0	(B,0,1),1
(C,0,1)	(B,1,0),1	(A,1,1),0
(D,1,0)	(C,0,0),0	(C,0,1),0
(D,1,1)	(D,1,0),1	(D,1,1),1

2. a) Find the equivalence partition for the machine shown in Table. Show a standard form of the corresponding reduced machine and find a minimum-length sequence that distinguishes state A from state B.

[WBUT 2007, 2012, 2015]

PS	NS, Z	
	X = 0	X = 1
A	B, 1	H, 1
B	F, 1	D, 1
C	D, 0	E, 1
D	C, 0	F, 1
E	D, 1	C, 1
F	C, 1	C, 1
G	C, 1	D, 1
H	C, 0	A, 1

Answer:

$$P_0 = (A \ B \ C \ D \ E \ F \ G \ H)$$

$$P_1 = (A \ B \ E \ F \ G) (C \ D \ H)$$

$$P_2 = (A \ B) (E \ F \ G) (C \ D \ H)$$

$$P_3 = (A) (B) (E \ F \ G) (C \ D \ H)$$

$$P_4 = (A) (B) (E \ F \ G) (C \ D) (H)$$

$$P_5 = (A) (B) (E \ F \ G) (C \ D) (H) \leftarrow \text{Final equivalent partition}$$

Let $\alpha = (A)$, $\beta = (B)$, $\gamma = (E \ F \ G)$,

$\delta = (C \ D)$, $\epsilon = (H)$.

Hence the standard form machine is

PS	NS, Z	
	x = 0	x = 1
α	$\beta, 1$	$\epsilon, 1$
β	$\gamma, 1$	$\delta, 1$
γ	$\delta, 0$	$\delta, 1$
δ	$\delta, 0$	$\gamma, 1$
ϵ	$\delta, 1$	$\alpha, 1$

There are no 1-length or 2-length sequence that distinguishes A and B.

The sequence 000 distinguishes A and B

In state A, the output is 111

In state B, the output is 110

- b) For the incompletely specified machines shown in Table-2 find a minimum-state reduced machine containing the original one.

[WBUT 2007]

Table - 2

PS	NS, Z	
	I_1	I_2
A	E, 0	B, 0
B	F, 0	A, 0
C	E, -	C, 0
D	F, 1	D, 0
E	C, 1	C, 0
F	D, -	B, 0

Answer:

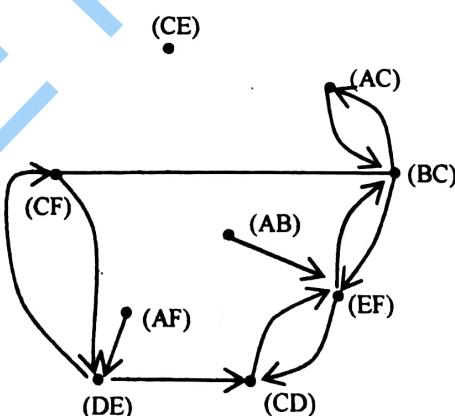
B	EF			
C	BC	EF, AC		
D	x	x	EF	
E	x	x	✓	CD, CF
F	DE	AB, DF	BC, DE	x BC, CD

A B C D E

Procedure

- Write down all compatible pair in respective cell.
- An 'x' is inserted in cell where the states have conflicting outputs. Here A and D have conflicting output. Similarly AE, BD, BE having conflicting output.
- Insert a check mark in cell CE because state E contain State C.
- Cross those cells having incompatible pair. Here cell BF contain incompatible pair DF so ignored the cell BF.

Compatibility Graph



The set $\{(AB), (AC), (BC), (CD), (EF)\} \rightarrow \{(ABC), (CD), (EF)\}$ is a minimal cover of the above machine.

3. a) For the following machine shown in Table determine whether or not it is lossless. If it is lossy, find the shortest output sequence produced by two different input sequences with the same initial and final states. If it is lossless, determine its order.

PS	NS, Z	
	X = 0	X = 1
A	A, 0	B, 0
B	C, 0	D, 0
C	D, 1	C, 1
D	B, 1	A, 1

b) Design a minimal inverse of the machine shown in Table

[WBUT 2007]

PS	NS, Z	
	X = 0	X = 1
A	C, 0	D, 1
B	D, 0	C, 1
C	A, 0	B, 0
D	C, 1	D, 1

Answer:

a) As another illustration, the above test is applied to machine M_9 of Table (i). This machine is shown to be lossless of order 3, since its testing graph (Fig. a) is loop-free and the longest path is of length 1.

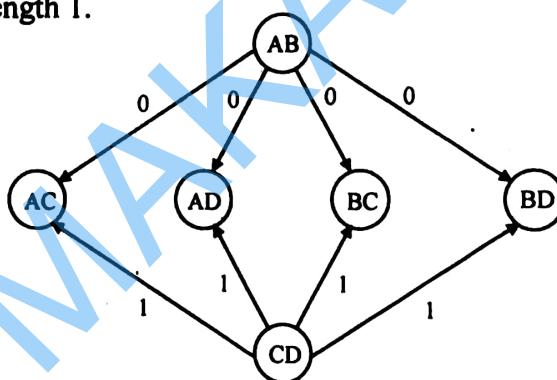


Fig: Testing graph for machine M_9

Table (i): Machine M_9

PS	NS, z	
	x = 0	x = 1
A	A, 0	B, 0
B	C, 0	D, 0
C	D, 1	C, 1
D	B, 1	A, 1

Table (ii): Testing table for

PS	z = 0	
	(AB)	(B, 0)
A	(AB)	B, 0
B	(CD)	D, 0
C	-	C, 1
D	-	A, 1
AB	(AC)(AD) (BC)(BD)	-
CD	-	(AC)(AD) (BC)(BD)

(b) we have the following triples:

$$\begin{array}{ll} (A, 0, 0) & (B, 0, 1) \\ (A, 1, 1) & (B, 1, 0) \end{array}$$

$$\begin{array}{ll} (C, 0, 0) & (D, 1, 0) \\ (C, 0, 1) & (D, 1, 1) \end{array}$$

The rest of the triples are not defined.

Thus the inverse machine has present states defined by the eight triples shown above. Now the next state of the inverse machine can also be defined by triples.

FORMAL LANGUAGE AND AUTOMATA THEORY

The first member of this triple is the state to which the machine M will go consuming the first input symbol.

The second member is nothing but the third member of the present state of the inverse machine and finally the third member is the present output of M.

The state table of the corresponding inverse machine is shown below: -
Machine-M'

PS	NS, X	
	$z = 0$	$z = 1$
(A,0,0)	(C,0,0),0	(C,0,1),0
(A,1,1)	(D,1,0),1	(D,1,1),1
(B,0,1)	(D,1,0),0	(D,1,1),0
(B,1,0)	(C,0,0),1	(C,0,1),1
(C,0,0)	(A,0,0),0	(B,0,1),1
(C,0,1)	(B,1,0),1	(A,1,1),0
(D,1,0)	(C,0,0),0	(C,0,1),0
(D,1,1)	(D,1,0),1	(D,1,1),1

4. a) What do you mean by k -equivalent states?

[WBUT 2009, 2011]

b) Minimize the following machine by partitioning the distinguishable states:

Present state	$i/p = 0$		$i/p = 1$	
	Next State	o/p	Next State	o/p
A	E	0	D	1
B	F	0	D	0
C	E	0	B	1
D	F	0	B	0
E	G	0	F	1
F	B	0	C	0
G	C	1	H	0
H	A	1	G	0

c) Give definition of lossy and lossless machine.

[WBUT 2009]

Answer:

a) Two states S_i and S_j of a finite state machine are said to be K equivalent if there exists no input sequence of length K or less that produces different outputs depending upon whether S_i or S_j is the initial state.

b) Initial partition $P_0 = (ABCDEFGH)$

Partition of 1-distinguishable states = $P_1 = (ACE)(BDF)(GH)$

Partition of 2-distinguishable states = $P_2 = (AC)(E)(BD)(F)(GH)$

Partition of 3-distinguishable states = $P_3 = \text{None}$

So, minimal machine has 5 states.

Let us map: $(AC) \rightarrow P$ $(E) \rightarrow Q$ $(BD) \rightarrow R$ $(F) \rightarrow S$ and $(GH) \rightarrow T$

POPULAR PUBLICATIONS

The minimal machine is:

PS	NS , z	
	$x = 0$	$x = 1$
P	Q, 0	R, 1
Q	T, 0	S, 1
R	S, 0	R, 0
S	R, 0	P, 0
T	P, 1	T, 0

c) A machine M is said to be (information) lossless if the knowledge of the initial state, the output sequence and the final state is sufficient to determine uniquely the input sequence.

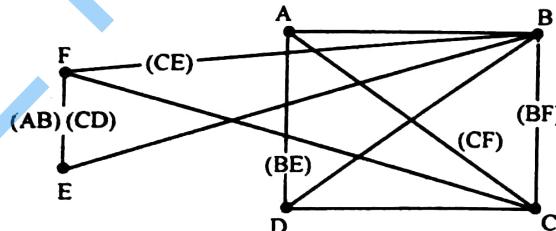
A machine that is not lossless is said to be lossy.

5. Draw the merger graph, merger table, compatibility graph and then minimize the following machine:
[WBUT 2009]

Present state	Next State, o/p		Next State, o/p	
	$i/p = 0$	$i/p = 1$	$i/p = 2$	$i/p = 3$
A	-	C, 1	E, 1	B, 1
B	E, 0	-	-	-
C	F, 0	F, 1	-	-, 1
D	-	-	B, 1	-
E	-	F, 0	A, 0	D, -
F	C, -	..	B, 0	C, 1

Answer:

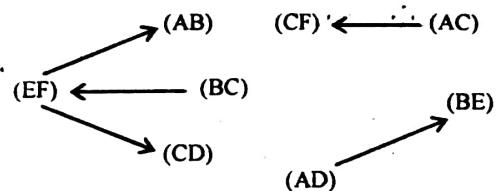
Merger Graph



Merger Table

	✓				
B					
C	CF	EF			
D	BE	✓	✓		
E	✗	✓	✗	✗	
F	✗	CE	✓	AB CD	
	A	B	C	D	E

Compatibility Graph



FORMAL LANGUAGE AND AUTOMATA THEORY

From the compatibility graph, we get $\{(AB), (CD), (EF)\}$ as a closed covering of compatibles.

Minimal Machine

PS	NS, z			
	i/p = 0	i/p = 1	i/p = 2	i/p = 3
$(AB) \rightarrow \alpha$	$\gamma, 0$	$\beta, 1$	$\gamma, 1$	$\alpha, 1$
$(CD) \rightarrow \beta$	$\gamma, 0$	$\gamma, 1$	$\alpha, 1$	$-, 1$
$(EF) \rightarrow \gamma$	$\beta, -$	$\gamma, 0$	$\alpha, 0$	$\beta, 1$

6. a) What is information lossless machine?

[WBUT 2010]

b) Consider the machine shown in the following table:

[WBUT 2010, 2014]

Present State	Next State	
	X = 0	X = 1
A	A, 1	C, 1
B	E, 0	B, 1
C	D, 0	A, 0
D	C, 0	B, 0
E	B, 1	A, 0

Is this machine information lossless of finite order? If yes, find the order μ .

Answer:

(a) A machine is to be (*information*) *lossless* if the knowledge of the initial state, the output sequence, and the final state is sufficient to determine uniquely the input sequence.

(b) **Step 1:** Check each row of the state table for an appearance of two identical next-state entries associated with the same output symbol.

Here (AC) is a compatible pair, since both A and C are the output 1-successors of A . Similarly, the pairs (AD) and (BC) are compatible pairs.

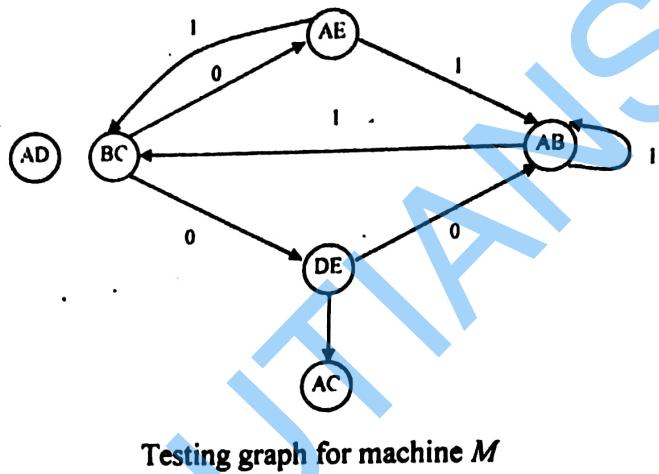
Step 2: If no identical entries appear, the next step is to construct the output successor table.

These pairs are used as row headings for the lower part of the testing table. Pairs (AE) and (DE) , which are implied by (BC) , are now made row headings, and so on. Note that contrary to the testing procedure for finite output memory, the testing table for information losslessness does not necessarily include all distinct pairs of states, but only the compatible ones.

POPULAR PUBLICATIONS

Testing table of machine M

PS	$z = 0$	$z = 1$
A	-	(AC)
B	E	B
C	(AD)	-
D	(BC)	-
<hr/>		
E	A	B
AC	-	-
AD	-	-
BC	(AE)(DE)	-
<hr/>		
AE	-	(AB)(BC)
DE	(AB)(AC)	-
AB	-	(AB)(BC)



Testing graph for machine M

The testing graph G of machine M is derived from the lower half of the testing table. The testing table does not contain any compatible pair consisting of repeated entries. So the machine is information lossless.

7. Consider the following machine:

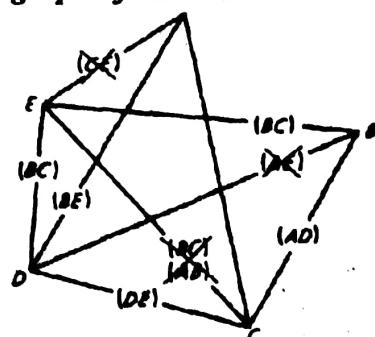
[WBUT 2010]

Present State	Next State			
	I_1	I_2	I_3	I_4
A	-	-	E, 1	-
B	C, 0	A, 1	B, 0	-
C	C, 0	D, 1	-	A, 0
D	-	E, 1	B, -	-
E	B, 0	-	C, -	B, 0

- i) Draw the merger graph.
- ii) Draw the merger table.
- iii) Draw the compatibility graph.
- iv) Find the minimal closed covering with justification.

Answer:

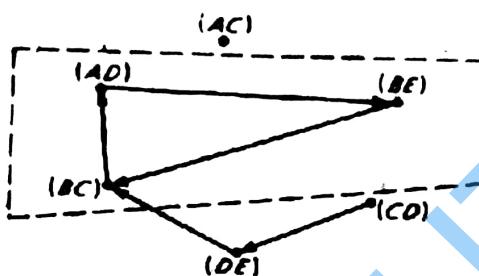
Step 1: First draw the merger graph of machine.



The set of maximal compatibles derives from the merger graph contains four members and is given by $\{(ACD), (BC), (BE), (DE)\}$

Step 2: The compatible pairs and their implied pairs are usually obtained by the merger graph, and since a set of states is a compatible if and only if every pair of states in that set is compatible, then for a given , the set of compatible pairs defines uniquely the entire set of compatibles .

Here from the above merger graph, an arc leads from vertex (AD) to vertex (BE) because the compatibility of (BE) is implied by that of (AD) . No arcs emanate from (AC) since no other compatible is implied by it, and so on.



Step 3: Select closed subgraph in the compatible graph.

8. Convert the following Mealy machine to equivalent Moore machine.

Present State	I/P=0		I/P=1	
	Next State	O/P	Next State	O/P
q_0	q_1	1	q_2	1
q_1	q_3	0	q_0	1
q_2	q_4	0	q_3	1
q_3	q_1	0	q_4	0
q_4	q_2	1	q_4	0

[WBUT 2014, 2017]

Answer:

State	Output
q_0	1
q_1	0, 1
q_2	0, 1
q_3	0
q_4	0, 1

State Redesignated

State	Output	Redesignated
q_0	1	q_0
q_1	0, 1	q_{10}, q_{11}
q_2	0, 1	q_{20}, q_{21}
q_3	0	q_3
q_4	0, 1	q_{40}, q_{41}

Present State	Moore Machine Table		Output
	$x = 0$	$x = 1$	
q_0	q_{11}	q_{21}	1
q_{10}	q_3	q_0	0
q_{11}	q_1	q_0	1
q_{20}	q_{40}	q_3	0
q_{21}	q_{40}	q_3	1
q_3	q_{10}	q_{40}	0
q_{40}	q_{21}	q_{40}	0
q_{41}	q_{21}	q_{40}	1

9. Consider the following state table

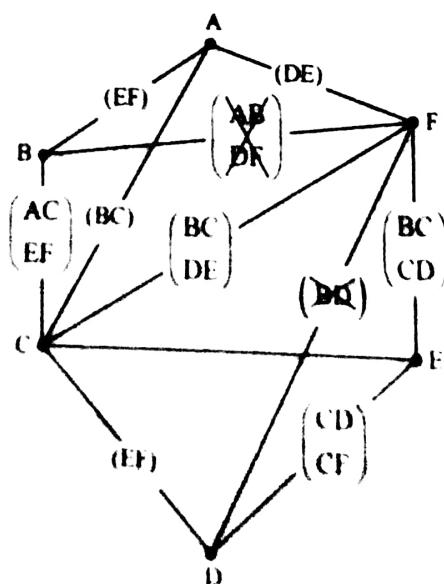
Present state	Next state, output	
	I1	I2
A	E, 0	B, 0
B	F, 0	A, 0
C	E, -	C, 0
D	F, 1	D, 0
E	C, 1	C, 0
F	D, -	B, 0

- a) Draw the merger graph.
- b) Draw the merger table.
- c) Draw the compatibility graph.
- d) Find the minimal machine which covers that machine.

Answer:

a)

[WBUT 2015]



b), c) & d) Refer to Question No. 2(b) of Long Answer Type Questions.

10. Construct a minimum state automata equivalent to a given automata M which transition table is given below:

[WBUT 2018]

State	input	
	a	b
$\rightarrow q_0$	q_0	q_3
q_1	q_2	q_5
q_2	q_3	q_4
q_3	q_0	q_5
q_4	q_0	q_6
q_5	q_1	q_4
q_6	q_1	q_3

Answer:

State	input	
	a	b
$\rightarrow q_0$	q_0	q_3
q_1	q_2	q_5
q_2	q_3	q_4
q_3	q_0	q_5
q_4	q_0	q_6
q_5	q_1	q_4
q_6	q_1	q_3

\therefore Partition algorithm is applied —

$$P_0 = (q_0 \ q_1 \ q_2 \ q_3 \ q_4 \ q_5 \ q_6)$$

$$P_1 = P_0$$

As, here not any state has same outputs and also no same state transition.
So, the given m/c is itself the minimized one.

11. Construct the merger table, merger graph, compatibility graph and minimal machine for the following Machine:

[WBUT 2018]

BS	NS, z	
	I ₁	I ₂
A	E, 0	B, 0
B	F, 0	A, 0
C	E, -	C, 0

POPULAR PUBLICATIONS

D	F, 1	D, 0
E	C, 1	C, 0
F	D, -	B, 0

Answer:

Refer to Question No. 9(a), (b), (c) & (d) of Long Answer Type Questions.

12. Construct testing table, testing graph for the following machine and test whether it has finite memory or Not, if yes then find the order. [WBUT 2018]

PS	NS, z	
	x = 0	x = 1
A	B, 0	D, 0
B	C, 0	C, 0
C	D, 0	A, 0
D	D, 0	A, 1

Answer:

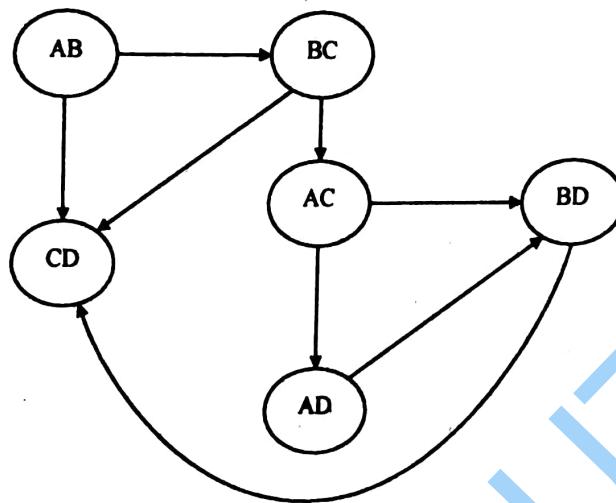
PS	NS, z	
	x = 0	x = 1
A	B, 0	D, 0
B	C, 0	C, 0
C	D, 0	A, 0
D	D, 0	A, 1

Testing Table:

PS	0/0	0/1	1/0	1/1
A	B	-	D	-
B	C	-	C	-
C	D	-	A	-
D	D	-	-	A
(AB)	(BC)	-	(CD)	-
(AC)	(BD)	-	(AD)	-
(AD)	(BD)	-	-	-
(BC)	(CD)	-	(AC)	-
(BD)	(CD)	-	-	-
(CD)	(DD)	-	-	-

(discarded)

Testing Graph:



∴ Here, in this testing graph, ∃ no loop so, we can say it is a finite memory machine.

∴ Order (μ) = Maximum length path (ℓ) + 1

∴ Here,

$$(AB \rightarrow BC \rightarrow AC \rightarrow AD \rightarrow BD \rightarrow CD)$$



max. path length is 6

$$\therefore \mu = 6 + 1 = 7 \quad (\text{Ans.})$$

STRUCTURE OF SEQUENTIAL MACHINES

Short Answer Type Questions

1. Is the following machine information lossless? If yes, find the order of losslessness. [WBUT 2011]

	NS, z	
	X = 0	X = 1
A	A, 0	B, 0
B	C, 0	D, 0
C	D, 1	C, 1
D	B, 1	A, 1

Answer:

Test of information losslessness

PS	NS, t	
	x = 0	x = 1
A	A, 0	B, 0
B	C, 0	D, 0
C	D, 1	C, 1
D	B, 1	A, 1

The first step of this testing procedure is to check each row of the transition table, whether there is two identical next state entries associated with the same output symbol. From the above table it is clear that, there is no identical next state entries. Now we have to construct the output successor table.

Testing table for Machine-M

PS	z = 0	z = 1
A	(AB)	—
B	(CD)	—
C	—	(CD)
D	—	(AB)
AB	(AB) (CD)	—
CD	—	(AB) (CD)
AC	—	—
AD	—	—
BC	—	—
BD	—	—

The lower part of the testing table is constructed from the compatible pair of states of the upper part of the table. In the lower part of the table all the implied pairs have been taken as a row heading.

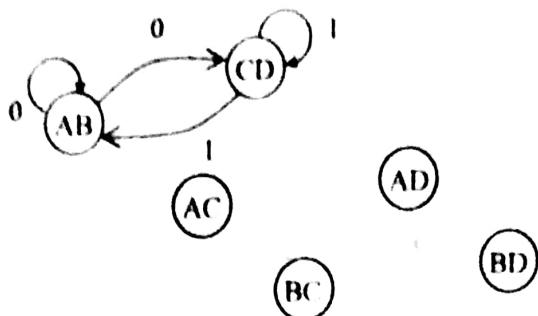


Fig. Testing graph for machine M

From the lower part of the testing table it is clear that, there is no compatible pair with repeated entries. This is necessary and sufficient for a machine to be lossless.
So clearly the given machine is lossless.

In the above testing graph the length of the largest path is $l = 2$.

Hence the order of losslessness, $\mu = l + 2$

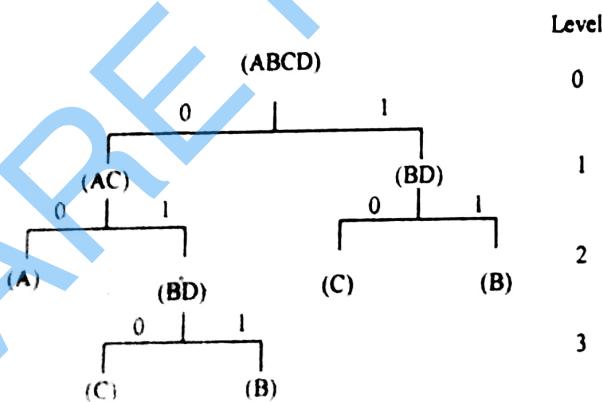
2. Test whether the following machine is definite or not

[WBUT 2011]

- i) by using synchronization tree
- ii) by using repeated derivation of contracted table
- iii) if the machine is definite,
what is the order of definiteness? Justify.

Present State	Next State	
	$a = 0$	$a = 1$
A	A	B
B	C	B
C	A	D
D	C	B

Answer:



It is clear that the tree is terminated by rule 2. So machine is definite machine.

Here $k = 3$

Therefore order of the definiteness is 3.

Long Answer Type Questions

1. Consider the machine given below:

[WBUT 2010]

Present State	Next State		
	X = 0	X = 1	Z
A	D	G	0
B	C	E	0
C	H	F	0
D	F	F	0
E	B	B	0
F	G	D	0
G	A	B	0
H	E	C	1

Drive the closed partitions. Construct a π -lattice for it.

Answer:

First we determine π_{SIS_j} for every pair of states:

1. A,B \rightarrow A,B;C,D;E,G; \rightarrow A,B;C,D;E,G;F,H; $\rightarrow \pi_a$
2. A,C \rightarrow A,C;D,H;G,F; \rightarrow A,C;D,H;G,F;F,E;C,F;A,G;B,D; \rightarrow A,C,E,F,G;B,D,H; \rightarrow A,C,E,F,G;B,D,H;D,H,B,G,A;G,F,B,D; \rightarrow A,B,C,D,E,F,G,H; $\rightarrow \pi(I)$
3. A,D \rightarrow A,D;D,F;G,F; \rightarrow A,D;G,F;B,D,G,F; \rightarrow A,B,D,G,F;A,C,D,F,G;B,D,G,E,F; \rightarrow A,B,C,D,E,F,G;A,B,C,D,F,G,I; \rightarrow A,B,C,D,E,F,G,H; $\rightarrow \pi(I)$
4. A,E \rightarrow A,E;B,D;B,G; \rightarrow A,E;B,D,G;A,C,F;B,E,F; \rightarrow A,B,C,D,E,F,G,H $\rightarrow \pi(I)$
5. A,F \rightarrow A,F;D,G; \rightarrow A,F;D,G;B,F; \rightarrow A,B,F;D,G;C,D,G;D,E,G; \rightarrow A,B,F;C,D,G,E;A,B,F,H; \rightarrow A,B,F,H;C,D,E,G; $\rightarrow \pi_c$
6. A,G \rightarrow A,G;A,D;B,G; \rightarrow A,B,D,G;A,C,D,F;B,E,F,G; \rightarrow A,B,C,D,E,F,G;D,C,H,F,B,G,A,E; \rightarrow A,B,C,D,E,F,G,H; $\rightarrow \pi(I)$
7. A,H \rightarrow A,H;D,E;C,G; \rightarrow A,H;D,E;C,G;B,F; $\rightarrow \pi_b$
8. B,C \rightarrow B,C;C,H;E,F; \rightarrow B,C;C,H;E,F;E,H;C,F;B,D;B,G; \rightarrow B,C,D,E,F,G,H;C,H,F,B,G,A,E; \rightarrow A,B,C,D,E,F,G,H; $\rightarrow \pi(I)$
9. B,D \rightarrow B,D;C,F;E,F; \rightarrow B,D;C,E,F;B,G,H;B,D,F; \rightarrow B,C,D,E,F,G,H;C,H,F,B,G,A,E; \rightarrow A,B,C,D,E,F,G,H; $\rightarrow \pi(I)$
10. B,E \rightarrow B,E;B,C; \rightarrow B,E;B,C;C,H;E,F; \rightarrow B,C,E,F,H;B,C,E,G,H;E,F,D,B,C; \rightarrow B,C,D,E,F,G,H; C,H,F,B,G,A,E; \rightarrow A,B,C,D,E,F,G,H; $\rightarrow \pi(I)$
11. B,F \rightarrow B,F;C,G;D,E; \rightarrow B,F;C,G;D,E;A,H; $\rightarrow \pi_b$
12. B,G \rightarrow B,G;A,C;B,E; \rightarrow B,G;A,C;B,E;D,H;F,G;B,C; \rightarrow A,B,C,E,F,G;D,H;A,B,C,D,G,H; \rightarrow A,B,C,D,E,F,G,H; $\rightarrow \pi(I)$
13. B,H \rightarrow B,H;C,E; \rightarrow B,H;C,E;B,F; \rightarrow B,F,H;C,E;C,E,G;C,D,E; \rightarrow B,F,H;C,D,E,G; \rightarrow B,F,H;C,D,E,G;A,B,F,H; \rightarrow A,B,F,H;C,D,E,G; $\rightarrow \pi_c$
14. C,D \rightarrow C,D;H,F; \rightarrow C,D;H,F;G,E; \rightarrow C,D;H,F;G,E;A,B; $\rightarrow \pi_a$
15. C,E \rightarrow C,E;B,H;B,F; \rightarrow C,E;B,F,H;C,E,G;C,D,E; \rightarrow B,F,H;C,D,E,G; \rightarrow B,F,H;C,D,E,G;A,B,F,H; \rightarrow A,B,F,H;C,D,E,G; $\rightarrow \pi_c$
16. C,F \rightarrow C,F;G,H;D,F; \rightarrow C,D,F;G,H;F,G,H;F,D; \rightarrow C,D,F,G,H;A,E,F,G,H;B,C,D,F; \rightarrow A,B,C,D,E,F,G,H; $\rightarrow \pi(I)$

17. $C,G \rightarrow C,G; A,H; B,F; \rightarrow C,G; A,H; B,F; D,E; \rightarrow \pi_a$
18. $C,H \rightarrow C,H; C,F; E,H; \rightarrow C,E,F,H; B,E,G,H; B,C,D,F; \rightarrow B,C,D,E,F,G,H; C,H,F,B,G; A,E; \rightarrow A,B,C,D,E,F,G,H; \rightarrow \pi(I)$
19. $D,E \rightarrow D,E; B,F; \rightarrow D,E; B,F; C,G; \rightarrow D,E; B,F; C,G; A,H; \rightarrow \pi_b$
20. $D,F \rightarrow D,F; F,G; F,D; \rightarrow D,F,G; A,F,G; B,D,F; \rightarrow A,B,D,F,G; C,D,A,F,G; G,E,F,B,D; \rightarrow A,B,C,D,E,F,G; D,C,H,F,B,G,A; \rightarrow A,B,C,D,E,F,G,H; \rightarrow \pi(I)$
21. $D,G \rightarrow D,G; A,F; B,F; \rightarrow D,G; A,B,F; \rightarrow D,G; A,B,F; C,D,G; D,E,G; \rightarrow C,D,G,E; A,B,F; \rightarrow C,D,G,E; A,B,F,H; \rightarrow \pi_c$
22. $D,H \rightarrow D,H; E,F; C,F; \rightarrow D,H; C,E,F; B,G,H; B,D,F; \rightarrow B,C,D,E,F,G,H; C,H,F,B,G,A; E; \rightarrow A,B,C,D,E,F,G,H; \rightarrow \pi(I)$
23. $E,F \rightarrow E,F; B,D; B,G; \rightarrow E,F; B,D,G; A,C,F; B,E,F; \rightarrow A,B,C,D,E,F,G; D,C,H,F,B,G,A; \rightarrow A,B,C,D,E,F,G,H; \rightarrow \pi(I)$
24. $E,G \rightarrow E,G; A,B; \rightarrow E,G; A,B; C,D; \rightarrow E,G; A,B; C,D; F,H; \rightarrow \pi_a$
25. $E,H \rightarrow E,H; B,E; B,C; \rightarrow E,H; B,C,E; B,C,H; B,E,F; \rightarrow B,C,E,F,H; B,C,E,G,H; \rightarrow B,C,E,F,G,H; C,H,B,G,A,E; E,F,B,D,B,C; \rightarrow A,B,C,D,E,F,G,H; \rightarrow \pi(I)$
26. $F,G \rightarrow F,G; A,G; B,D; \rightarrow A,F,G; B,D; A,D,G; B,D,G; C,F; E,F; \rightarrow A,B,C,D,E,F,G; D,C,H,F,B,G,A; \rightarrow A,B,C,D,E,F,G,H; \rightarrow \pi(I)$
27. $F,H \rightarrow F,H; C,D; E,G; \rightarrow F,H; C,D; E,G; A,B; \rightarrow \pi_a$
28. $G,H \rightarrow G,H; A,E; B,C; \rightarrow G,H; A,E; B,C; B,D; B,G; C,H; E,F; \rightarrow A,B,C,D,E,F,G,H; \rightarrow \pi(I)$

$$\pi_a = \{\overline{A, B}; \overline{C, D}; \overline{E, G}; \overline{F, H}\}$$

$$\pi_b = \{\overline{A, H}; \overline{B, F}; \overline{C, G}; \overline{D, E}\}$$

$$\pi_c = \{\overline{A, B, F, H}; \overline{C, D, E, G}\}$$

Parallel Decomposition of Machine:

$$\pi_a + \pi_b = \pi_c = \{A, B, F, H; C, D, E, G\}$$

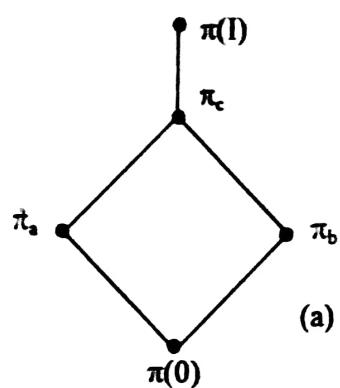
PS	NS		z_1
	x=0	x=1	
A,B \rightarrow a	b	c	0
C,D \rightarrow b	d	d	0
E,G \rightarrow c	a	a	0
F,H \rightarrow d	c	b	1

Machine M_a

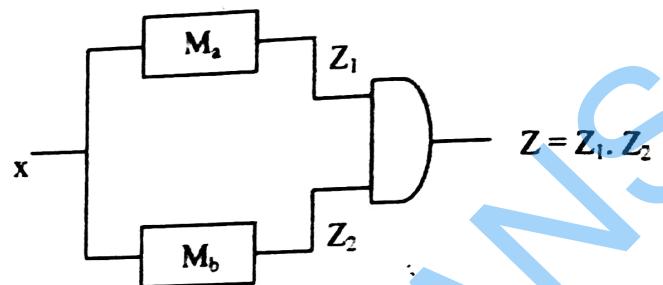
PS	NS		z_1
	x=0	x=1	
A,H \rightarrow α	δ	χ	1
B,F \rightarrow β	χ	δ	0
C,G \rightarrow γ	α	β	0
D,E \rightarrow δ	β	β	0

POPULAR PUBLICATIONS

Machine M_b



(a) π -Lattice



Schematic diagram of machine
 M_b with parallel components

FINITE AUTOMATION

Multiple Choice Type Questions

1. Palindromes cannot be recognized by any FSM because [WBUT 2007, 2012]

- a) an FSM cannot remember arbitrary, large amount of information
- b) an FSM cannot deterministically fix the mid-point
- c) even or the mid-point is known, an FSM cannot find whether the second half of the string matches the first half
- d) all of these

Answer: (c)

2. If S is the number of states in NDFA then equivalent DFA can have maximum of [WBUT 2008]

- a) S states
- b) S-1 state
- c) 2^S states

d) $2^S - 1$ states

Answer: (c)

3. DFA has [WBUT 2008]

- a) single final state
- b) more than one initial states
- c) unique path (for a set of inputs) to the final state
- d) all of these

Answer: (c)

4. What is the highest type number to the grammar given by the following production rules [WBUT 2008]

- $S \rightarrow Aa, A \rightarrow c|Ba, B \rightarrow abc$
- a) zero
 - b) one
 - c) two
 - d) three

Answer: (d)

5. Given an arbitrary NDFA with n states, the maximum number of states in an equivalent minimized DFA is at least [WBUT 2008]

- a) n^0
- b) 2^n
- c) $n!$
- d) None of these

Answer: (d)

6. DFA converted from an NFA with n states can have maximum [WBUT 2009]

- a) n states
- b) $n!$ states
- c) 2^n states

d) C_2 states

Answer: (c)

7. The production grammar $\{S \rightarrow aSbb, S \rightarrow abb\}$ is [WBUT 2010]

- a) type-3 grammar
- b) type-2 grammar
- c) type-1 grammar
- d) type-0 grammar

Answer: (a)

8. The reduced grammar of $S \rightarrow AB|a, A \rightarrow a$ is [WBUT 2010]

- a) $S \rightarrow a$
- b) $S \rightarrow a|A$
- c) $S \rightarrow a$
 $A \rightarrow a$

d) $S \rightarrow aa$

Answer: (c)

POPULAR PUBLICATIONS

9. Which of the following grammars generates strings with any number of 1's?
a) $S \rightarrow 1A, A \rightarrow \epsilon$
b) $S \rightarrow 1S, S \rightarrow \epsilon$
c) $S \rightarrow S1, S \rightarrow \epsilon$
d) (b) & (c)

[WBUT 2014]

Answer: (d)

10. DFA has a transition function

- a) $Q \times \Sigma \text{ to } Q$ b) $Q \times \Sigma \text{ to } 2^Q$ c) both (a) and (b) d) none of these

[WBUT 2014]

Answer: (a)

11. If Q is the number of states in the NFA, the equivalent DFA can have maximum number of states

- a) Q b) $Q - 1$ c) $2Q - 1$

d) 2^Q

[WBUT 2014]

Answer: (d)

12. Can a DFA simulate NFA?

- a) no b) yes

- c) sometimes

d) depends on DFA

Answer: (b)

13. Let N be an NFA with n states and let M be the minimized DFA with m states recognizing the same language. Which of the following is NECESSARILY true?

- a) $m \leq 2^n$
b) $n \leq m$
c) M has one accept state
d) $m = 2^n$

[WBUT 2014, 2015]

Answer: (a)

14. The string 1101 does not belong to the set represented by

[WBUT 2014, 2015, 2016]

- a) $110^*(0+1)$
b) $1(0+1)^*101$
c) $(10)^*(01)^*(00+11)^*$
d) $(00+(11)^*01)^*$

Answer: (c)

15. $A = (a+b)^*a$ and $B = b(a+b)^*$ then $A \cap B$ will be

[WBUT 2014]

- a) $(a+b)^*ab$ b) $ab(a+b)^*$ c) $a(a+b)^*b$

d) $b(a+b)^*$

Answer: (d)

16. The string 1101 does not belong to the set represented by

[WBUT 2014]

- a) $110^*(0+1)$
b) $1(0+1)^*101$
c) $(10)^*(01)^*(00+11)^*$

- d) $[00+(11)^*0]^*$

Answer: (d)

17. An automata is a/an device.

[WBUT 2014]

- a) acceptor only
b) acceptor / rejector
c) rejector only
d) generating

Answer: (b)

18. k -level equivalence is possible between two finite automata

[WBUT 2016]

- a) if all equivalence above k -level exist between them
- b) if all level equivalence up to $k-1$ level already exist
- c) if both contain at least k number of states
- d) if both contain exactly k number of states

Answer: (b)

Short Answer Type Questions

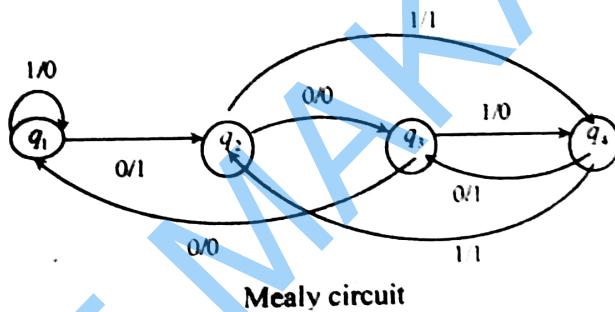
1. Convert the Mealy Machine (given below) to a Moore Machine.

[WBUT 2008]

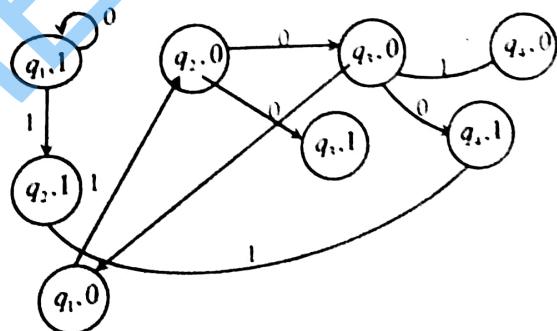
Present State	Next State	i/p=0	Next State	i/p=1
	State	Output	State	Output
Q_1	Q_2	1	Q_1	0
Q_2	Q_3	0	Q_4	1
Q_3	Q_1	0	Q_4	0
Q_4	Q_3	1	Q_2	1

Answer:

The Mealy machine constructed from transition diagram is



The equivalent Moore circuit is given by



POPULAR PUBLICATIONS

2. Construct a Mealy machine which is equivalent to the Moore machine given below: [WBUT 2009]

PS	NS		z
	x = 0	x = 1	
q_0	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_3	1

Answer:

	NS, z	
	x = 0	x = 1
q_0	$q_1, 0$	$q_2, 1$
q_1	$q_3, 1$	$q_2, 1$
q_2	$q_2, 1$	$q_1, 0$
q_3	$q_0, 1$	$q_3, 1$

3. State the difference between DFA and NFA.

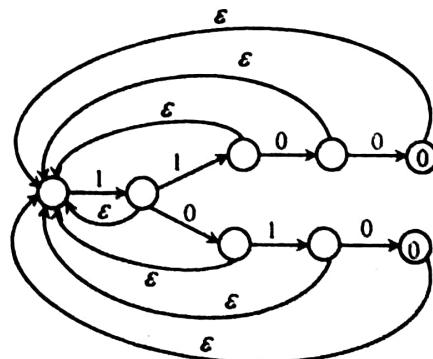
[WBUT 2009, 2010]

Answer:

In a DFA, there can be no ϵ -transitions. Also, for a given state q and a symbol $a \in \Sigma$, there can be at most one entry. An NFA does not have any of the above restrictions.

4. Design an NFA which accepts set of all binary strings containing 1100 or 1010 as substrings. [WBUT 2009, 2010, 2016]

Answer:



5. Draw the transition diagram of a finite state automaton that accepts all strings over {0, 1}

a) having odd number of 0's

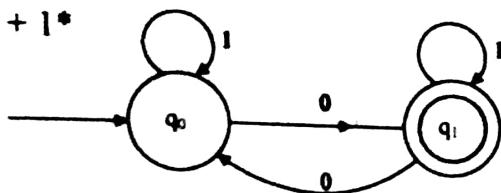
b) having even number of 0's and even number of 1's.

[WBUT 2010]

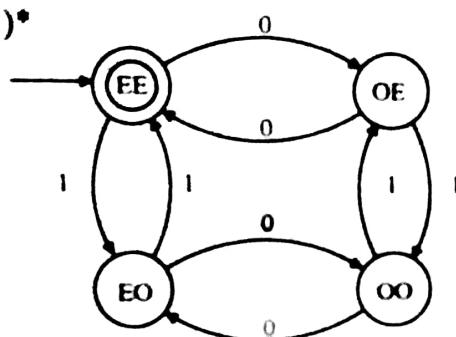
FORMAL LANGUAGE AND AUTOMATA THEORY

Answer:

a) $(1^*01^*)^* + 1^*$



b) $(0^*1^*0^*!)^*$



6. State the difference between DFA and NFA.

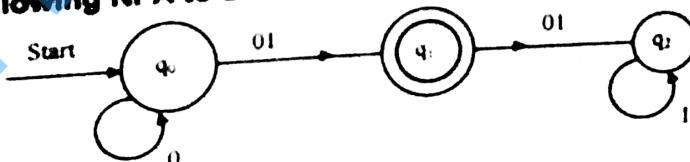
[WBUT 2010, 2018]

Answer:

1. The transition function for nfa ie delta is multi valued where as for dfa it is single valued.
2. Checking membership is easy with dfa where as it is difficult for nfa
3. Construction of nfa is very easy where as for dfa it is difficult
4. Space required for dfa is more where for nfa it is less
5. Backtracking is allowed in dfa, but it is not possible in every case in nfa.
6. For every input and output we can construct dfa machine, but it is not possible to construct an nfa machine for every input and output.
7. There is only 1 final state in nfa but there can be more than 1 final state in dfa.

[WBUT 2010]

7. Convert the following NFA to DFA.



Answer:

State/ ϵ	0	1
$\rightarrow q_0$	q_0, q_1	q_1
q_1	q_2	q_2
q_2	-	q_2

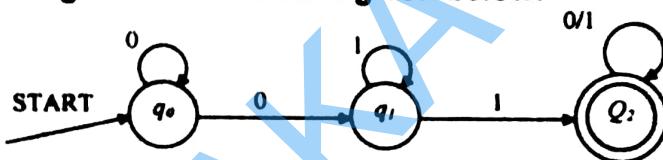
NFA

DFA

Q	Q_0	Q_1
$\rightarrow [q_0]$	$[q_0, q_1]$	$[q_1]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_1, q_2]$
$[q_1, q_2]$	$[q_2]$	$[q_2]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_1, q_2]$
$[q_2]$	ϕ	$[q_2]$
$([q_1])$	$[q_2]$	$[q_2]$
ϕ	ϕ	ϕ

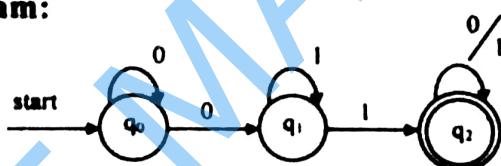
8. Construct a DFA diagram from the NFA given below:

[WBUT 2011, 2012]



Answer:

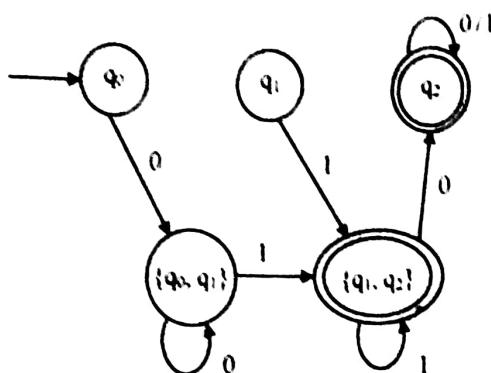
The NFA diagram:



According to the above diagram, the transition table will be formed as:

Present state	i/p	
	0	1
q_0	$\{q_0, q_1\}$	$\{\phi\}$
q_1	$\{\phi\}$	$\{q_1, q_2\}$
q_2	q_2	q_2
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	q_2	$\{q_1, q_2\}$

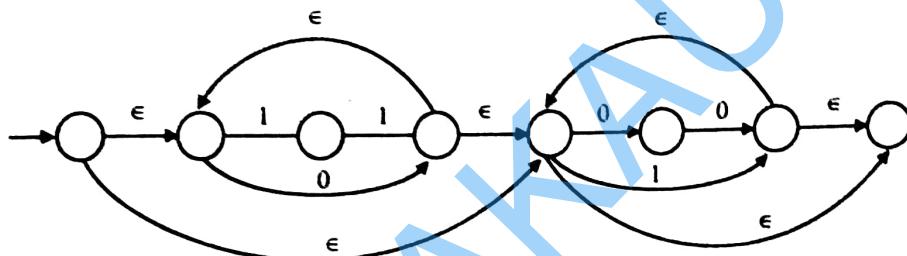
Now, according to the above table the DFA diagram will be,



The DFA Diagram

9. Construct a NFA with ϵ or λ transition for $r = (11+0)^*(00+1)^*$. [WBUT 2011, 2012]

Answer:



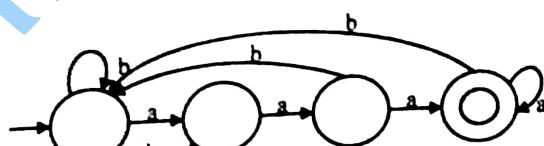
10. a) Give DFA which reads strings from $\{a, b\}$ and with aaa.

b) Construct a DFA equivalent to $M = \{\{q_0, q\}, \{0, 1\}, \delta_{q_0}, \{q_0\}\}$, δ is given by the state table. [WBUT 2012]

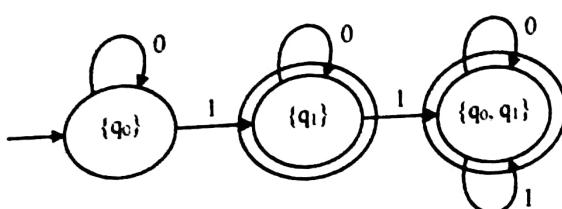
State/	0	1
q0	q0	q1
q1	q1	q0, q1

Answer:

a)



b)



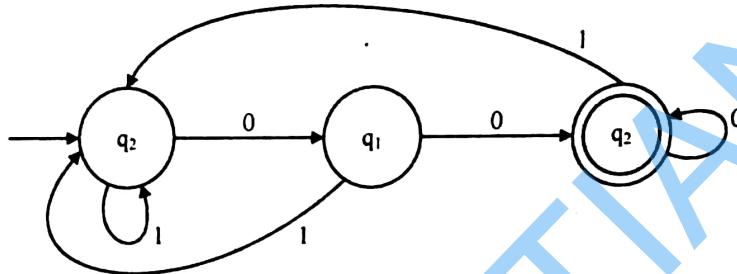
Assuming q_1 to be the final state.

POPULAR PUBLICATIONS

11. Design a Finite automata that accepts set of strings that every string ends with 00 over alphabet {0, 1}. [WBUT 2013]

Answer:

FA that accepts set of strings that every string ends with 00 over alphabet {0, 1}



12. Define NFA. Construct equivalent DFA from the given NFA.

Present State	Next State	
	0	1
$\rightarrow q_0$	q_0, q_1	q_2
q_1	q_2	q_1
q_2	q_1	q_2

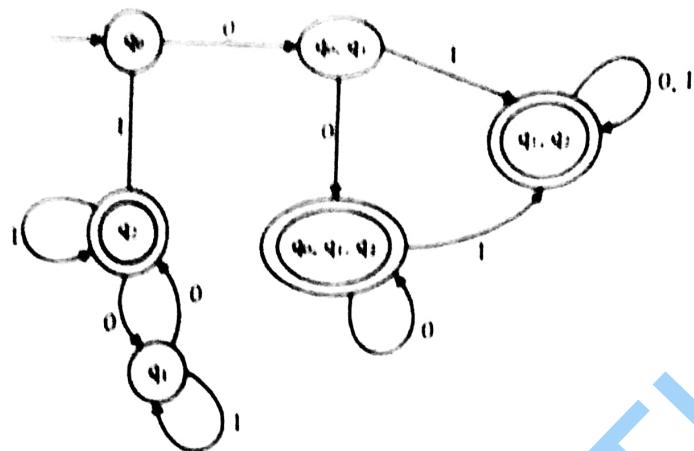
[WBUT 2014, 2015, 2017]

Answer:

In automata theory, a non-deterministic finite automaton (NFA), or non-deterministic finite state machine, is a finite state machine that (1) does not require input symbols for state transitions and (2) is capable of transitioning to zero or two or more states for a given start state and input symbol. This distinguishes it from a deterministic finite automaton (DFA), in which all transitions are uniquely determined and in which an input symbol is required for all state transitions. Although NFAs and DFAs have distinct definitions, all NFAs can be translated to equivalent DFAs using the subset construction algorithm, i.e., constructed DFAs and their corresponding NFAs recognize the same formal language. Like DFAs, NFAs only recognize regular languages.

State table

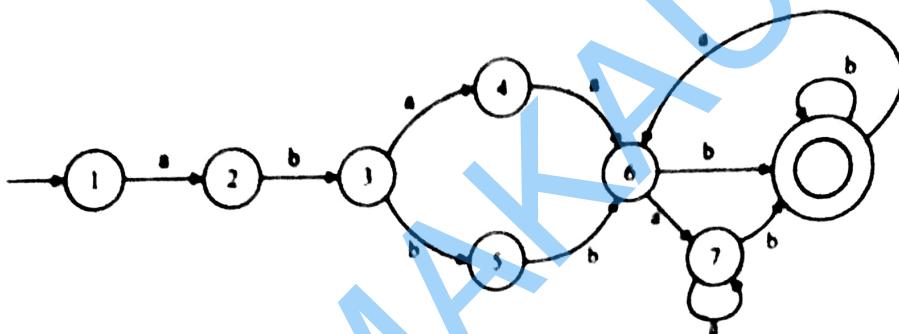
	0	1
q_0	(q_0, q_1)	(q_2)
(q_0, q_1)	(q_0, q_1, q_2)	(q_1, q_2)
(q_0, q_1, q_2)	(q_0, q_1, q_2)	(q_1, q_2)
(q_1, q_2)	(q_1, q_2)	(q_1, q_2)
q_2	q_1	q_2



13. Construct Finite Automata equivalent to the Regular Expression.
 $L = ab(aa+bb)(a+b)^*b.$

[WBUT 2014]

Answer:



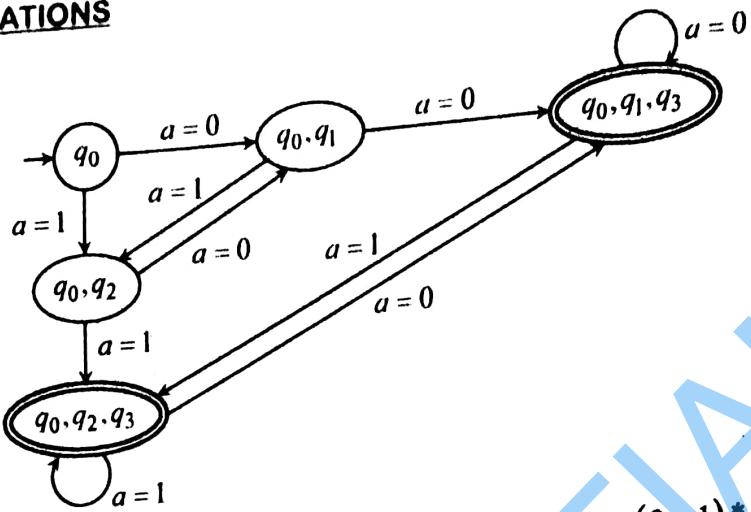
14. Convert the following NFA into an equivalent DFA.

[WBUT 2016]

PS	NS	
	$a = 0$	$a = 1$
$\rightarrow q_0$	q_0, q_1	q_0, q_2
q_1	q_1	-
q_2	-	q_1
q_3	q_1	q_1

Answer:

PS	NS	
	$a = 0$	$a = 1$
$\rightarrow q_0$	(q_0, q_1)	(q_0, q_2)
(q_0, q_1)	(q_0, q_1, q_3)	(q_0, q_2)
(q_0, q_2)	(q_0, q_1)	(q_0, q_2, q_3)
q_0, q_1, q_3	(q_0, q_1, q_3)	(q_0, q_2, q_3)
q_0, q_2, q_3	(q_0, q_1, q_3)	(q_0, q_2, q_3)



15. Define DFA. Derive the DFA for the regular language $0(0+1)^*1$ for the symbols $\Sigma = \{0, 1\}$. [WBUT 2016]

Answer:

1st part:

A deterministic finite automaton (DFA) is a 5-tuple. $(Q, \Sigma, \delta, q_0, F)$, where. Q is a finite set called the states, Σ is a finite set called the alphabet, $\delta : Q \times \Sigma \rightarrow Q$ is the transition function, $q_0 \in Q$ is the start state, and $F \subseteq Q$ is the set of accept states.

2nd part:



16. Construct a minimum state automaton from the transition table given below:

PS	$x = 0$	$x = 1$
q_0	q_1	q_2
q_1	q_2	q_3
q_2	q_2	q_4
q_3	q_3	q_3
q_4	q_4	q_4
q_5	q_5	q_4

q_3, q_4, q_5 are final states.

[WBUT 2016]

Answer:

In the finite automata, the states are $\{q_0, q_1, q_2, q_3, q_4, q_5\}$. Name this set as S_0 .

$$S_0 : \{q_0, q_1, q_2, q_3, q_4, q_5\}$$

All of the states are 0 equivalents.

In the finite automata, there are two types of states: final state and non-final states. So, divide the set of states into two parts, Q_1 and Q_2 .

$$Q_1 = \{q_0, q_1, q_2\} \quad Q_2 = \{q_3, q_4, q_5\}$$

$$S_1 : \{\{q_0, q_1, q_2\}, \{q_3, q_4, q_5\}\}$$

The states belonging to the same subset are 1-equivalent because they are in the same set for string length 1. The states belonging to different subsets are 1-distinguishable. For input 0 and 1, q_0 goes to q_1 and q_2 , respectively. Both of the states belong to the same subset. For q_1 and q_2 with input 0 and 1, the next states are q_2, q_3 , and q_2, q_4 , respectively. For both of the states, for input 0, the next state belongs to one subset, and for input 1, the next state belongs to another subset. So, q_0 can be distinguished from q_1 and q_2 .

The next states with input 0 and 1 for states q_3, q_4 , and q_5 , belong to the same subset. So, they cannot be divided.

$$S_2 : \{\{q_0\}, \{q_1, q_2\}, \{q_3, q_4, q_5\}\}$$

q_0 is the single state in the subset. So, it cannot be divided.

For states q_1 and q_2 with input 0 and 1, for both of the cases, one state belongs to one subset and another state belongs to another subset. So, they cannot be divided.

The next states with input 0 and 1 for states q_3, q_4 , and q_5 , belong to the same subset. So, they cannot be divided. So in the next step,

$$S_3 : \{\{q_0\}, \{q_1, q_2\}, \{q_3, q_4, q_5\}\}$$

S_2 and S_3 are equivalent.

As step $(n-1)$ and step n are the same, there is no need of further advancement.

In the minimized automata, the number of states is 3.

The minimized finite automata is presented in tabular format as follows:

State	Next State .	
	I/P=0	I/P=1
$\{q_0\}$	$\{q_1\}$	$\{q_2\}$
$\{q_1, q_2\}$	$\{q_2\}$	$\{q_3, q_4\}$
$\{q_3, q_4, q_5\}$	$\{q_3, q_4, q_5\}$	$\{q_3, q_4, q_5\}$

But $\{q_1\}$, $\{q_2\}$, and $\{q_3, q_4\}$ do not exist under the column of present state. They are not states of the minimized finite automata, but they are subset of the states. In the next state columns, by replacing the subsets by proper state, the modified table becomes

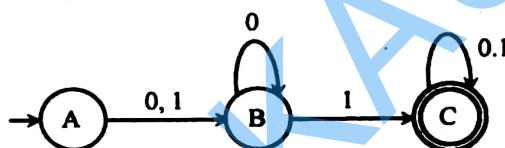
State	Next State	
	I/P=0	I/P=1
$\{q_0\}$	$\{q_1, q_2\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_3, q_4, q_5\}$
$\{q_3, q_4, q_5\}$	$\{q_3, q_4, q_5\}$	$\{q_3, q_4, q_5\}$

POPULAR PUBLICATIONS

As q_0 is the beginning state of the original finite automata, $\{q_0\}$ will be the beginning state of minimized finite automata. As q_3, q_4 , and q_5 are the final states of the original finite automata, the set of the states containing any of the states as element is final state. Here, all the states are contained in a single set $\{q_3, q_4, q_5\}$ and, therefore, it is the final state. By replacing $\{q_0\}$ as A, $\{q_1, q_2\}$ as B, and $\{q_3, q_4, q_5\}$ as C, the modified minimized finite automata becomes,

State	Next State	
	I/P=0	I/P=1
$\rightarrow A$		
B	B	B
(C)	C	C

The transitional diagram of the minimized finite automata is given in following figure.



17. What is Homomorphism? Find the homomorphic image of $L = \{aa, aba\}$ where h is defined as $h(a) = ab$ and $h(b) = bbc$. [WBUT 2018]

Answer:

1st Part:

Homomorphism:

A homomorphism is a map between two algebraic structures of the same type, that preserves the operations of the structures.

$\therefore f: A \rightarrow B$ where two sets A, B having same structure.

Then, say '*' is a binary operation.

$$\Rightarrow f(x * y) = f(x) * f(y)$$

for every pair of $(x, y) \in A$

2nd Part:

$$\therefore L = \{aa, aba\}$$

where $h(a) = ab, h(b) = bbc$

$$\therefore h(a \ a) = h(a) \ h(a) = ab \ ab$$

$$h(a \ b \ a) = h(a) \ h(b) \ h(a) = ab \ bbc \ ab$$

\therefore Homomorphic image of L is —

$$\{ab \ ab, ab \ bbc \ ab\}$$

Long Answer Type Questions

- 1. a) Design a 2-input 2-output Mealy machine, which takes as input a binary stream and generates an output of 1 only when a sequence of the pattern 01011 is found in the input stream. Design should be clearly justified. [WBUT 2007, 2010]**
- Answer:**

The required Mealy machine is given below:

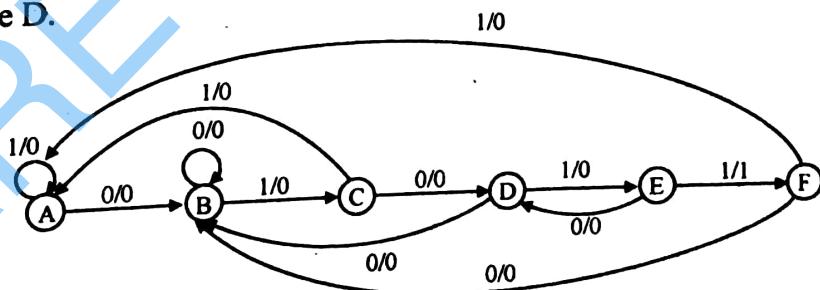
PS	NS, Z	
	X = 0	X = 1
A	B, 0	A, 0
B	B, 0	C, 0
C	D, 0	A, 0
D	B, 0	E, 0
E	D, 0	F, 1
F	B, 0	A, 0

Justification of Design

The forward moving (i.e., A → B → C → D → E → F) transitions can be trivially justified. Also justified are 1 transitions from states A, C and F to state A because this indicates the sequence has been broken completely and scanning must begin again from state A.

We note that the sequence starts with a 0 and upon receiving the first 0, the state is B. Hence, if the sequence breaks in any state B, D or F upon receiving a 0, we should count this 0 as the first possible symbol of the input and begin from state B.

Getting a 0 in the state E is a special case. State E is reached upon receiving the subsequence 0101. If a 0 is received, the sequence is 01010, the last three symbols of which (i.e. 010) is the first three symbols of the sequence to be recognized. But the state which is reached upon receiving subsequence 010 is D. Hence the 0 transition from state E should be state D.



State	Basic String
A	C
B	0
C	01
D	010
E	0101
F	01011

Pending	Basic String Possible	State recognizing largest suffix of basic string possible
A,1	1	A
B,0	00	B
C,1	011	A
D,0	0100	B
E,0	01010	D
F,0	010110	B
F,1	010111	A

b) From the Mealy machine above find the equivalent Moore machine. [WBUT 2007]

Answer:

The Mealy machine designed in part a) produces output 1 only when there is a transition into state F.

Hence the equivalent Moore machine is:

PS	NS		Output
	X = 0	X = 1	
A	B	A	0
B	B	C	0
C	D	A	0
D	B	E	0
E	D	F	0
F	B	A	1

c) Check whether the Mealy machine you obtained is a minimal one or not. Give proper justification to your answer. [WBUT 2007]

Answer:

The designed Mealy machine is NOT minimal. If we begin with the initial partition $P_0 = (ABCDEF)$, we will get $P_1 = (ABCDF)(E)$ because only in state E we get output 1. Thereafter, states D, C and B form independent sets but states A and F are indistinguishable.

The minimal machine is therefore:

PS	NS, Z	
	X = 0	X = 1
A	B, 0	A, 0
B	B, 0	C, 0
C	D, 0	A, 0
D	B, 0	E, 0
E	D, 0	A, 1

2. a) Find a reduced grammar equivalent to the grammar

$$S \rightarrow aAa, A \rightarrow bBB, B \rightarrow ab, C \rightarrow aB.$$

[WBUT 2009]

Answer:

Clearly C is a useless symbol. Eliminating C, we get,

$$S \rightarrow aAa, A \rightarrow bBB, B \rightarrow ab$$

Now we replace all occurrences of B on rhs of production using $B \rightarrow ab$, getting:

$$S \rightarrow aAa, A \rightarrow b ab ab$$

Now we similarly substitute A, getting

$$S \rightarrow a ba ba ba$$

which is the equivalent reduced grammar.

b) Explain the concept of 2-way finite automata.

[WBUT 2009]

Answer:

In a one-way (deterministic) finite automata, after every state transition, the input “head” moves to the ‘next’ input, i.e., to the ‘right’. In a two way finite automata, the “head” can move either to the “right” or to the “left”.

More formally:

A 2DFA is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$ where Σ is a finite set of input symbols, Q is a finite set of states, $q_0 \in Q$ is a “start state” and $F \subseteq Q$ is a set of accepting states. δ is a map from $Q \times \Sigma$ to $Q \times \{L, R\}$. If $\delta(q, a) = (p, L)$, then in state q , scanning symbol a , the 2DFA enters state p and moves its head “left”, if $\delta(q, a) = (p, R)$, the 2DFA enters state p and moves “right”.

A 2DFA accepts a string if it moves “right” beyond the last symbol, at the same time entering an accepting state.

It can be proved that if L is a language accepted by a 2DFA then L is a regular set.

3. a) What do you mean by Distinguishable and Indistinguishable state?

[WBUT 2011, 2012, 2016]

Answer:

Two distinct states, S_i , and S_j , of a machine M are *distinguishable* if and only if there exists at least one finite input sequence which, when applied to M , causes different output sequences, depending on whether S_i or S_j is the initial state.

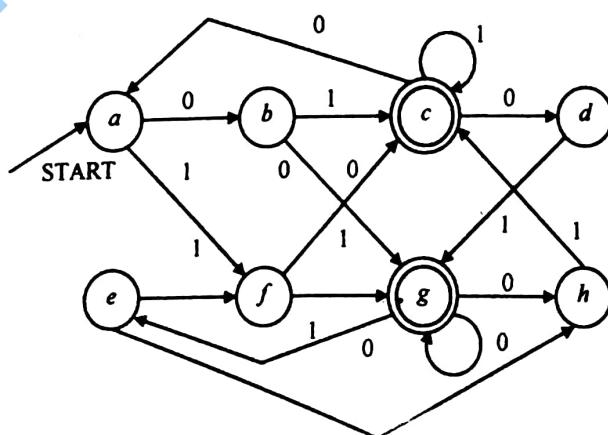
The sequence which distinguishes these states is called a distinguishing sequence of the pair (S_i, S_j) .

If no distinguishing sequence exists for the pair (S_i, S_j) , then the states are said to be *indistinguishable*.

The principle of state minimization of a finite state machine is based on the principle of identifying equivalent classes for states which are indistinguishable in the original machine and then assigning one state per class in the reduced machine.

b) Use Myhill Nerode Theorem to minimize the following finite automata:

[WBUT 2011, 2012]



Answer:

g	x						
f		x					
e		x					
d		x					
c	x		x		x		
b		x				x	
a						x	
	h	g	f	e	d	c	b

Next we can see that $\delta(a, 0) = b$, and $\delta(d, 0) = g$ and (b, g) is marked, so, (a, b) gets marked too. Similarly we can mark,

g	x						
f		x					
e		x	x				
d		x		x			
c	x		x		x		
b	x	x		x		x	
a			x		x	x	x
	a	g	f	e	d	c	b

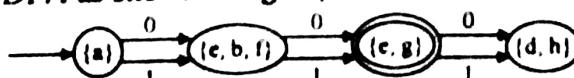
In the next round we can find that $\delta(d, 1) = g$ and $\delta(g, 1) = e$ where, (g, e) is marked. So, (d, f) will be marked. Similarly we can mark

g	x						
f		x					
e		x	x				
d	x	x	x	x			
c	x		x	x	x		
b	x	x		x		x	
a	x	x	x	x	x	x	x
	a	g	f	e	d	c	b

Here $\{a\}$ is completely distinguishable state.

Hence the indistinguishable states are $(e, g), (f, h), (e, h), (b, f), (b, d)$. This can be refined to get indistinguishable sets of states as: $\{e, b, f\}, \{e, g\}, \{d, h\}$

We get the minimized DFA as shown in figure,



4. a) Construct a DFA from the NFA given below:

State/ Σ	I/P	
	0	1
$\rightarrow Q_0$	Q_0, Q_3	Q_0, Q_4
Q_3	Q_f
Q_4	Q_f
Q_f (Final State)	Q_f	Q_f

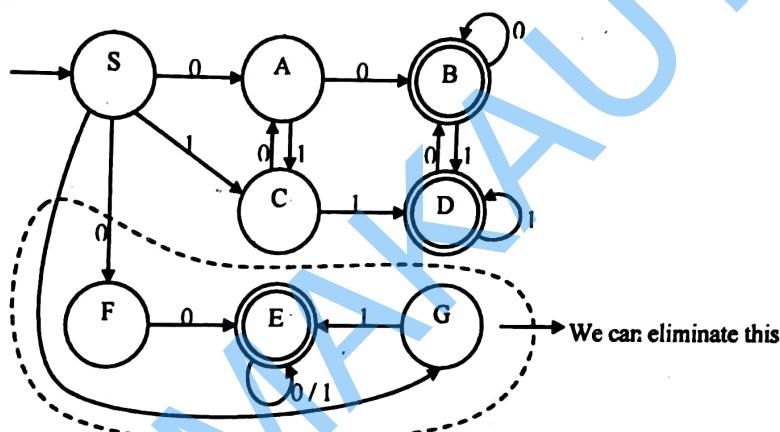
b) Construct a NFA for the regular expression $(0+1)^* 1(0+1)$.

[WBUT 2013]

Answer:

a) **DFA from the NFA:** Here Null transitions are replaced by New (N) states

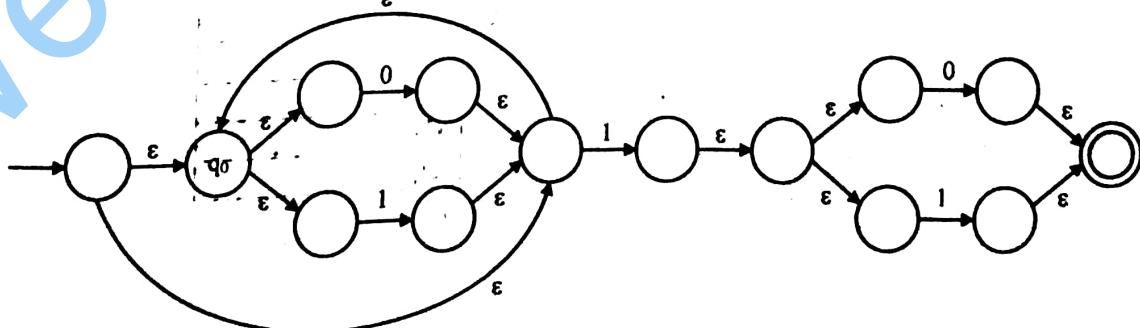
States	Input	
	0	1
Q_0	$\{Q_0, Q_3\}$	$\{Q_0, Q_4\}$
$\{Q_3\}$	Q_f	N
Q_4	N	Q_f
$\{Q_f\}$	Q_f	$\{Q_f\}$
$\{Q_0, Q_4\}$	$\{Q_0, Q_3\}$	$\{Q_0, Q_4, Q_f\}$
$\{Q_0, Q_3\}$	$\{Q_0, Q_3, Q_f\}$	$\{Q_0, Q_4\}$
$\{Q_0, Q_3, Q_f\}$	$\{Q_0, Q_3, Q_f\}$	$\{Q_0, Q_4, Q_f\}$
$\{Q_0, Q_4, Q_f\}$	$\{Q_0, Q_3, Q_f\}$	$\{Q_0, Q_4, Q_f\}$



The following are the states that have been renamed.

- $[Q_0] = S$
- $[Q_0, Q_3] = A$
- $[Q_0, Q_3, Q_f] = B$
- $[Q_0, Q_4] = C$
- $[Q_0, Q_4, Q_f] = D$
- $[Q_3] = F$
- $[Q_4] = G$
- $[Q_f] = E$

b)



POPULAR PUBLICATIONS

5. Minimize the following machine by applying Myhill-Nerode theorem.

PS	NS	
	X = a	X = b
→ A	B	E
B	C	D
C	H	I
D	I	H
E	F	G
F	H	I
G	H	I
H	H	H
I	I	I

[WBUT 2013]

Answer:

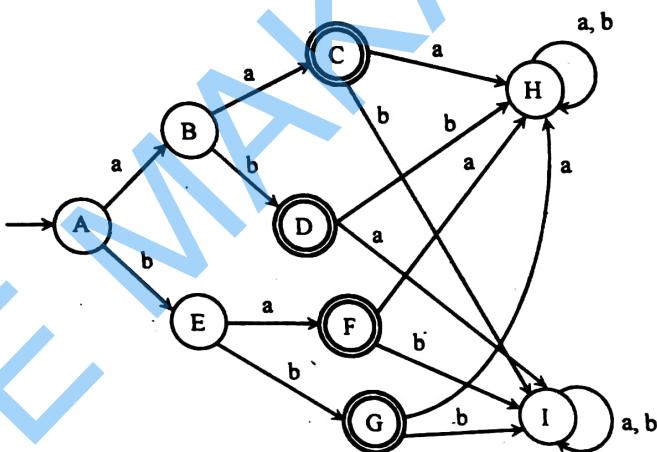


Fig: 1

There is no unreachable state in Fig. 1 so we can directly apply Myhill-Nerode theorem as follows:

$$Q = \{A, B, C, D, E, F, G, H, I\}$$

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

Initial state = A

$$F = \{C, D, F, G\}$$

$$\{Q - F\} = \{A, B, E, H, I\}$$

We use an ordered $F \times F = \{(C, D), (C, F), (C, G), (D, F), (C, G), (F, G)\}$

and

$$(Q - F) \times (Q - F) = \{(A, B), (A, E), (A, H), (A, I), (B, E), (B, H), (B, I), (E, H), (E, I), (H, I)\}$$

Transition table will be following:

δ / Σ	<i>a</i>	<i>b</i>
$\rightarrow A$	<i>B</i>	<i>E</i>
<i>B</i>	<i>C</i>	<i>D</i>
$*C$	<i>H</i>	<i>I</i>
$*D$	<i>I</i>	<i>H</i>
<i>E</i>	<i>F</i>	<i>G</i>
$*F$	<i>H</i>	<i>I</i>
$*G$	<i>H</i>	<i>I</i>
<i>H</i>	<i>H</i>	<i>H</i>
<i>I</i>	<i>I</i>	<i>I</i>

Step 1: Now build the matrix labeling the "p" rows A,B,C ... and labeling the "q" column A,B,... and put the dashes on the diagonal and below the diagonal

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>	<i>H</i>	<i>I</i>
<i>A</i>	-								
<i>B</i>	-	-							
<i>C</i>	-	-	-						
<i>D</i>	-	-	-	-					
<i>E</i>	-	-	-	-	-				
<i>F</i>	-	-	-	-	-	-			
<i>G</i>	-	-	-	-	-	-	-		
<i>H</i>	-	-	-	-	-	-	-	-	
<i>I</i>	-	-	-	-	-	-	-	-	-

Fig: 2

Step 2: Now mark X at (p, q) in upper triangle such that p in F (Final states) and q in $(Q - F)$ or (non-final states) as subscript of p is lower than the subscript of q or p in $(Q - F)$ and q in F and p is lower than q.

$$F = \{C, D, F, G\}$$

$$Q - F = \{A, B, E, H, I\}$$

$F \times (Q - F)$ or $(Q - F) \times F$ such that p is lower than q.

$$\begin{aligned} &= \{(C, E), (C, H), (C, I), (D, E), (D, H), (D, I), (F, H), (F, I), (G, I), (G, H), \\ &\quad (A, C), (A, D), (A, F), (A, G), (B, C), (B, D), (B, F), (B, G), (E, F), (E, G)\} \end{aligned}$$

POPULAR PUBLICATIONS

Now mark \times in the matrix given in Fig: 3.

	A	B	C	D	E	F	G	H	I
A	-		\times	\times		\times	\times		
B	-	-	\times	\times		\times	\times		
C	-	-	-		\times			\times	\times
D	-	-	-	-	\times			\times	\times
E	-	-	-	-	-	\times	\times		
F	-	-	-	-	-	-		\times	\times
G	-	-	-	-	-	-	\times	\times	\times
H	-	-	-	-	-	-	-		
I	-	-	-	-	-	-	-	-	-

Fig: 3

Step 3: Mark ' \times ' and ' 0 ' in the matrix:

First we are considering $Q - F$

$Q - F = \{A, B, E, H, I\}$, so ordered pair of

$$(Q - F) \times (Q - F) = \{(A, B), (A, E), (A, H), (A, I), (B, E), (B, H), (B, I), (E, H), (E, I), (H, I)\}$$

* Now we will select (p, q) from $(Q - F) \times (Q - F)$ and find (r, s) as

$r = \delta(p, a), s = \delta(q, a)$, if (r, s) is X or \times then (p, q) will be x . If (r, s) is neither X nor \times then (p, q) will be '0'.

→ for $(A, B), r = \delta(A, a) = B, s = \delta(B, a) = C$.

It can be check from matrix in Fig. 3 that (B, C) is \times so (A, B) will be x .

→ for $(A, E), r = \delta(A, a) = B, s = \delta(E, a) = F$

(B, F) is also \times so (A, E) will be x .

→ for $(A, H), r = \delta(A, a) = B, s = \delta(H, a) = H$

(B, H) is not \times but it is x so (A, H) will be x .

→ for $(A, I), r = \delta(A, a) = B, s = \delta(I, a) = I$

(B, I) is not \times but it is x so (A, I) will be x .

→ for $(B, H), r = \delta(B, a) = C, s = \delta(H, a) = H$

(C, H) is \times so (B, H) will be x .

→ for $(B, I), r = \delta(B, a) = C, s = \delta(I, a) = I$

(C, I) is \times so (B, I) will be x .

→ for $(E, H), r = \delta(E, a) = F, s = \delta(H, a) = H$

(F, H) is \times so (E, H) will be x .

\rightarrow for $(E, I), r = \delta(E, a) = F, s = \delta(I, a) = I$

(F, I) is \times so (E, I) will be x .

\rightarrow for $(H, I), r = \delta(H, a) = H, s = \delta(I, a) = I$

(H, I) is neither \times nor x so (H, I) will be "0".

Now consider $F = \{C, D, F, G\}$

Ordered pair of $F \times F = \{(C, D), (C, F), (C, G), (D, F), (D, G), (F, G)\}$

\rightarrow for $(C, D), r = \delta(C, a) = H, s = \delta(D, a) = I$

(H, I) is neither \times nor x so (C, D) will be "0".

\rightarrow for $(C, F), r = \delta(C, a) = H, s = \delta(F, a) = H$

(H, H) is dash so (C, F) will be "0".

\rightarrow for $(C, G), r = \delta(C, a) = H, s = \delta(G, a) = H$

(H, H) is dash so (C, G) is "0".

\rightarrow for $(D, F), r = \delta(D, a) = I, s = \delta(F, a) = H$

(I, H) is dash so check (H, I) , it is also neither \times nor x so (D, F) is "0" entry.

\rightarrow for $(D, G), r = \delta(D, a) = I, s = \delta(G, a) = H$

(D, G) is also "0" entry.

\rightarrow for $(F, G), r = \delta(F, a) = H, s = \delta(G, a) = H$

(H, H) is dash so (F, G) will be "0" entry.

Rest of the entries in upper triangle will be "0". Now fill the "x" and "0" in matrix given in Fig. 3.

	A	B	C	D	E	F	G	H	I
A	-	x	x	x	x	x	x	x	x
B	-	-	x	x	0	x	x	x	x
C	-	-	-	0	x	0	0	x	x
D	-	-	-	-	x	0	0	x	x
E	-	-	-	-	-	x	x	x	x
F	-	-	-	-	-	-	0	x	x
G	-	-	-	-	-	-	-	x	x
H	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	-

Fig: 4

POPULAR PUBLICATIONS

The "0" at (B, E) means (B, E) is a state in minimum machine.

So if resulting minimum machine is $M' = (Q', \Sigma', \delta', A', F')$

$$Q' = (\{A\}, \{B, E\}, \{C, D, F, G\}, \{H, I\})$$

$$F' = (\{C, D, F, G\})$$

$$A' = A$$

Now consider Fig. 1 (given DFA) and draw the transition table for minimum machine as

$$\delta'(\{A\}, a) = \{B, E\}$$

Since $\delta(A, a) = B$ in given DFA, B is in $\{B, E\}$ state of minimized machinery on the same way, the transition table for M' is

δ' / Σ	a	b
$\rightarrow \{A\}$	$\{B, E\}$	$\{B, E\}$
$\{B, E\}$	$\{C, D, F, G\}$	$\{C, D, F, G\}$
$*\{C, D, F, G\}$	$\{H, I\}$	$\{H, I\}$
$\{H, I\}$	$\{H, I\}$	$\{H, I\}$

So minimized machine is

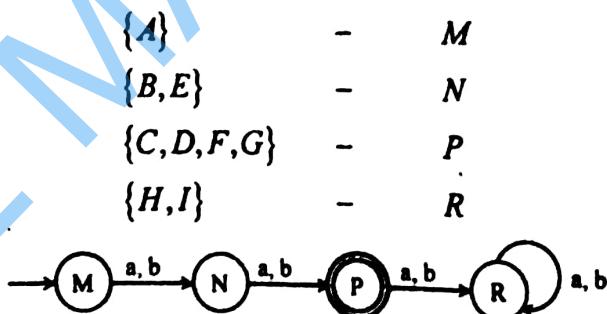


Fig: 5 Minimized DFA

6. Find equivalent partitions and minimize the following Finite State Machine.

PS	NS, z	
	x=0	x=1
A	B, 0	H, 1
B	C, 0	G, 1
C	B, 0	F, 1
D	F, 1	C, 1
E	B, 1	C, 1
F	B, 1	B, 1
G	C, 1	B, 1
H	D, 1	A, 1

[WBUT 2014]

FORMAL LANGUAGE AND AUTOMATA THEORY

Minimize the following incompletely specified machine.

PS	NS, z		
	I ₁	I ₂	I ₃
A	A, 1	D, —	C, —
B	A, —	D, —	E, —
C	E, 0	A, 1	—, —
D	E, —	A, 1	—, —
E	E, 0	—, —	C, —

[WBUT 2014, 2017]

Answer:

1st Part:

Equivalence partitions are

$$P_0 = (ABCDEFGH)$$

$$P_1 = ((ABC)(DEFGH))$$

$$P_2 = ((ABC)((DH)(EFG)))$$

$$P_3 = (((A)(BC))((D)(H))(EFG)))$$

$$P_4 = (((A)(BC))((D)(H))(EFG)))$$

Minimization: We know that equivalent partition is unique.

So, $P_3 = (((A)(BC))((D)(H))(EFG)))$ is the unique combination. Here, every single set represents one state of the minimized machine.

Let us rename these partitions for simplification.

Rename (A) as S_1 , (BC) as S_2 , (D) as S_3 , (H) as S_4 and (EFG) as $S_5(A)$ with input 0 goes to (B) and so there will be a transaction from S_1 to S_2 with input 0. (A) with input 1 goes to (H) and so there will be a transaction from S_1 to S_4 with input 1. (BC) with input 0 goes to (BC) for input 0. There will be a transaction from S_2 to S_2 for input 0. (BC) with input 1 goes to (FG). There will be a transaction from S_2 to S_5 for input 1.

By this process, the whole table of the minimized machine is constructed.

The minimized machine becomes

Present State	Next State, z	
	X = 0	X = 1
$S_1(A)$	$S_2, 0$	$S_4, 1$
$S_2(BC)$	$S_2, 0$	$S_5, 1$
$S_3(D)$	$S_5, 1$	$S_2, 1$
$S_4(H)$	$S_3, 1$	$S_1, 1$
$S_5(EFG)$	$S_2, 1$	$S_2, 1$

2nd Part:

$$P_0 = (ABCDE)$$

$$P_1 = (AB)(CDE)$$

This is the final partition.

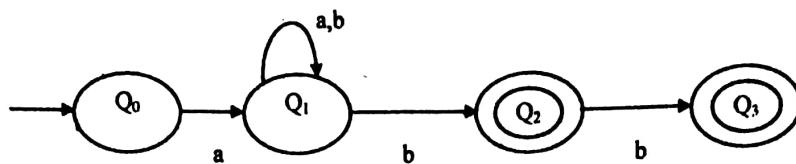
Hence minimized machine is:

PS	NS, Z		
	I ₁	I ₂	I ₃
$S_1 = (AB)$	$S_1, 1$	$S_2, —$	$S_2, —$
$S_2 = (CDE)$	$S_2, 0$	$S_1, 1$	$S_2, —$

POPULAR PUBLICATIONS

7. Consider the N.F.A given by the following diagram

[WBUT 2015]

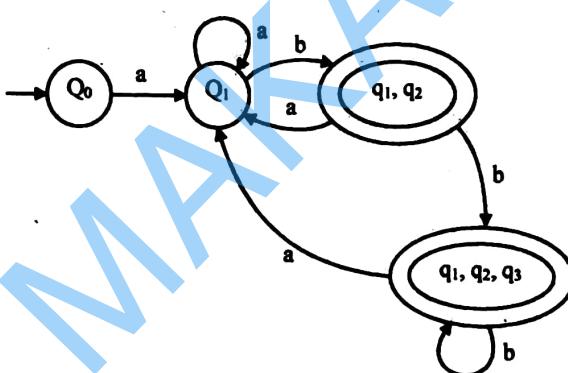


Find the equivalent D.F.A without e-transition.

Answer:

State	a	b
Q_0	$\{Q_1\}$	\emptyset
Q_1	$\{Q_1\}$	$\{Q_1, Q_2\}$
$\{Q_1, Q_2\}$	$\{Q_1\}$	$\{Q_1, Q_2, Q_3\}$
$\{Q_1, Q_2, Q_3\}$	$\{Q_1\}$	$\{Q_1, Q_2, Q_3\}$

So the equivalent D.F.A. is



8. State and prove Arden's theorem in regular expression.

[WBUT 2017]

OR,

State and prove Erden's Theorem.

[WBUT 2018]

Answer:

In order to find out a regular expression of a Finite Automaton, we use Arden's Theorem along with the properties of regular expressions.

Statement –

Let P and Q be two regular expressions.

If P does not contain null string, then $R = Q + RP$ has a unique solution that is $R = QP^*$

Proof –

$$R = Q + (Q + RP)P \quad [\text{After putting the value } R = Q + RP]$$

$$= Q + QP + RPP$$

When we put the value of R recursively again and again, we get the following equation –

$$R = Q + QP + QP^2 + QP^3 \dots$$

$$R = Q(\epsilon + P + P^2 + P^3 + \dots)$$

$$R = QP^* \quad [\text{As } P^* \text{ represents } (\epsilon + P + P^2 + P^3 + \dots)]$$

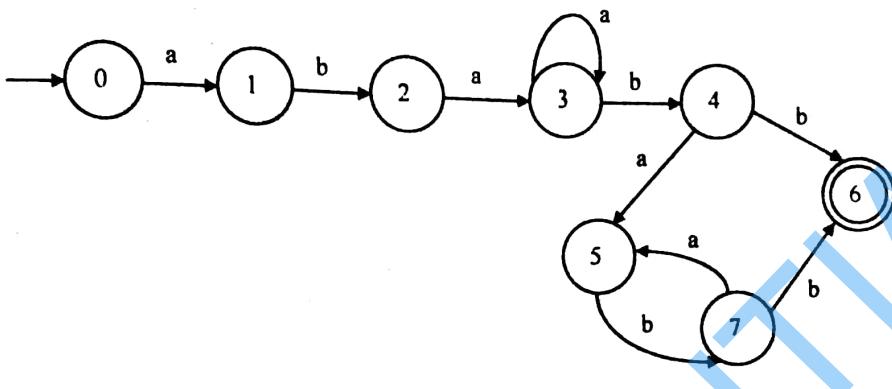
Hence, proved.

9. Construct Finite Automata equivalent to the Regular Expression

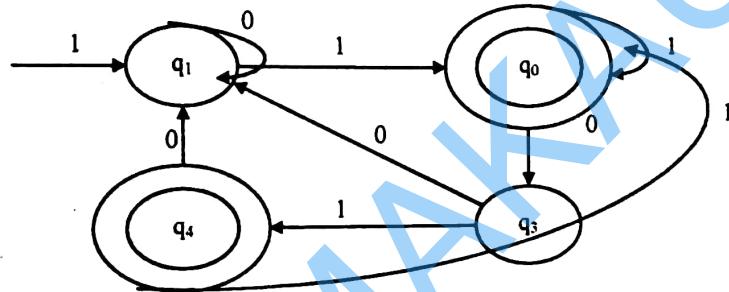
$$L = ab(a+b)(ab)^*b.$$

[WBUT 2017]

Answer:



10. Find the regular expression corresponding to the following figure: [WBUT 2018]



Answer:

Using Arden's theorem —

$$R = Q + RP, \quad R = QP^*$$

∴ Here,

$$q_1 = \epsilon + q_1 0 + q_4 0 + q_3 0$$

$$q_0 = q_0 1 + q_4 1 + q_1 1$$

$$q_3 = q_0 0$$

$$q_4 = q_3 1$$

$$\therefore q_1 = \epsilon + q_4 0 + q_3 0 + q_1$$

$$\therefore q_0 = q_0 1 + q_4 1 + q_1 1$$

$$\Rightarrow q_0 = q_0 1 + q_3 11 + q_1 1$$

$$\Rightarrow q_0 = q_0 1 + q_0 011 + q_1 1$$

$$\Rightarrow q_0 = q_1 1 + q_0 (1 + 011)$$

$$\Rightarrow q_0 = q_1 1 (1 + 011)^*$$

$$\therefore q_1 = \epsilon + q_4 0 + q_3 0 + q_1$$

$$\Rightarrow q_1 = \epsilon + q_3 10 + q_3 0 + q_1$$

$$\Rightarrow q_1 = \epsilon + q_0 010 + q_0 00 + q_1$$

POPULAR PUBLICATIONS

$$\begin{aligned}
 \Rightarrow q_1 &= \epsilon + q_1 1(1+011)^* 010 + q_1 1(1+011)^* 00 + q_1 \\
 \Rightarrow q_1 &= \epsilon + q_1 (1 \cdot (1+011)^* 010 + 1(1+011)^* 00 + \epsilon) \\
 \Rightarrow q_1 &= \epsilon \cdot (1(1+011)^* 010 + 1(1+011)^* 00 + \epsilon)^* \\
 \Rightarrow q_1 &= (1 + (1+011)^* 010 + 1(1+011)^* 00)^* \\
 \Rightarrow q_4 &= q_3 1 = q_0 01 = q_1 1(1+011)^* 01 \\
 &\quad = (1(1+011)^* 010 + 1(1+011)^* 00)^* 1(1+011)^* 01
 \end{aligned}$$

\therefore Final regular expression:

$$(q_4 + q_0) = ((1 + (1+011)^* 010 + 1(1+011)^* 00)^* 1(1+011)^*) (\epsilon + 01) \quad (\text{Ans.})$$

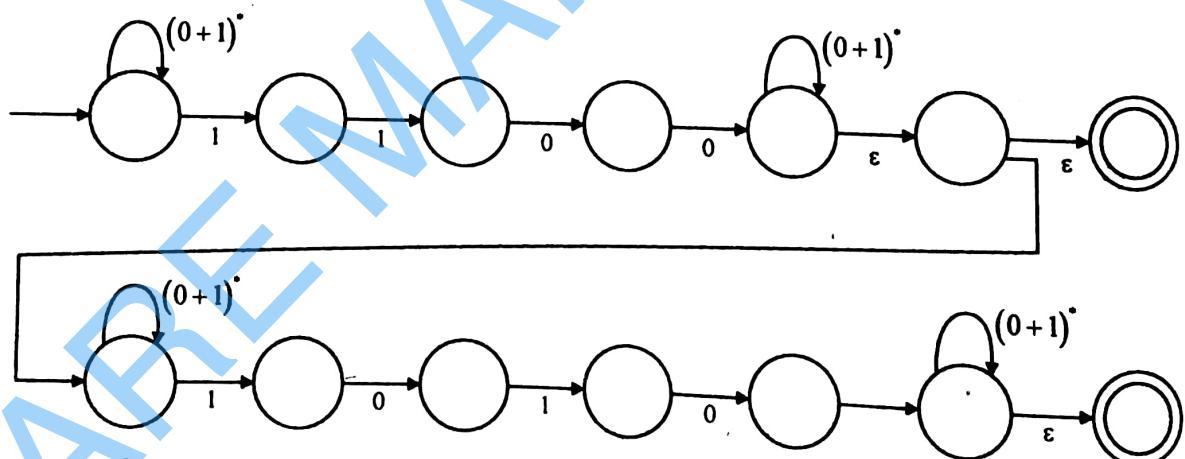
11. Design an NFA which accepts set of all binary strings containing 1100 or 1010 as substrings. [WBUT 2018]

Answer:

\therefore A NFA over $\Sigma = \{0, 1\}^*$ with 1100 or 1010 as substring.

\therefore Regular expression:

$$(0+1)^* 1100(0+1)^* + (0+1)^* 1010(0+1)^*$$



M/C: $(0+1)^* 1100(0+1)^* + (0+1)^* 1010(0+1)^*$

CLOSURE PROPERTIES OF REGULAR SETS

Multiple Choice Type Questions

1. Pumping Lemma is generally used for proving whether
 a) a given grammar is regular
 b) a given grammar is not regular
 c) two given grammars are equivalent or not
 d) none of these

Answer: (b)

[WBUT 2008, 2016]

2. A solution to the equation $R = Q + RP^*$ is
 a) $R^2 = QP^*$ b) $Q = RP^*$ c) $P = RQ^*$ d) $R = PQ^*$

Answer: (a)

[WBUT 2007, 2013, 2017]

3. Which of the following sets is regular?
 a) $\{a^i : i = n^2, n \geq 1\}$
 b) $\{a^n : p \text{ is a prime}\}$
 c) $\{ww : w \text{ is in } \{a, b\}^*\}$
 d) $\{a^{2^n} : n \geq 1\}$

Answer: (d)

[WBUT 2007, 2012]

4. The value of $L(\phi^*)$ is
 a) Σ b) $\{\epsilon\}$ c) $\{\}$ d) none of these

Answer: (c)

[WBUT 2007]

5. The regular expression representing the set of all strings over $\{x, y\}$ ending with xx beginning with y is
 a) $xx(x+y)^*y$ b) $yy(x+y)^*x$ c) $y(x+y)^*xx$ d) $y(xy)^*xx$

Answer: (d)

[WBUT 2007, 2012]

6. Regular expression $(a|b)(a|b)$ denotes the set
 a) $\{a, b, ab, aa\}$ b) $\{a, b, ba, bb\}$ c) $\{a, b\}$ d) $\{aa, ab, ba, bb\}$

Answer: (d)

7. Which of the following regular expressions over $\{0, 1\}$ denotes the set of all strings not containing 100 as a sub-string?
 a) $0^*(1^*0)^*$ b) 0^*1010^* c) $0^*1^*01^*$ d) $0^*(10+1)^*$

Answer: (d)

[WBUT 2008]

POPULAR PUBLICATIONS

8. Which of the following is regular?

- a) Strings of 0's, whose length is a perfect square
- b) Strings of all palindromes made up of 0's & 1's
- c) Strings of 0's, whose length is a prime number
- d) Strings of odd number of zeroes

Answer: (d)

[WBUT 2008]

9. $a^*(a+b)^*$ is equivalent to

- a) $a^* + b^*$
- b) $(ab)^*$
- c) $a^* b^*$

[WBUT 2009, 2013]

d) None of these

Answer: (d)

10. Consider the following regular expression: $R = (ab + abb)^* bbab$

Which of the following is not in the set denoted by R?

- a) ababab
- b) ababbabbbab
- c) abbbbab

[WBUT 2010]

d) abbabbbbab

Answer: (a)

11. Which of the following is correct?

- a) Language can be derived from the FA
- b) Regular expressions can be derived from the FA
- c) FA can be derived from the language
- d) Both (a) & (b)

[WBUT 2010, 2016]

Answer: (d)

12. The following production rules of a regular grammar generates a language L

$$S \rightarrow aS / bS / a / b$$

The regular expression for L is

- a) $a + b$
- b) $(a+b)$
- c) $(a+b)(a+b)^*$
- d) $(aa+bb)a^*b^*$

[WBUT 2011]

Answer: (b)

13. The basic limitation of FSM is that

- a) it can't remember arbitrary large amount of information
- b) it sometimes recognize grammar that is not regular
- c) it sometimes fails to recognize grammar that is regular
- d) all of these

[WBUT 2012, 2016]

Answer: (a)

14. What is the RE for the language set strings with at least one 1, one 2 and one 3?

- a) $1+2+3$
- b) $11^* 22^* 33^*$

- c) $1^* 2^* 3^*$

[WBUT 2012]

Answer: (b)

- d) both (a) and (b)

15. The logic of pumping lemma is a good example of

- a) The pigeon-hole principle
- c) Recursion

- b) the divide and conquer technique
- d) Iteration

Answer: (a)

FORMAL LANGUAGE AND AUTOMATA THEORY

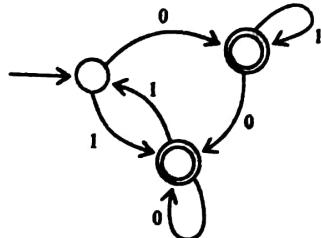
16. Which of the following strings can be obtained by the language $L = \{a^i b^{2i} | i \geq 1\}$

- a) aaabbbbbb
- b) aabbb
- c) abbabbba
- d) aaaabbbabb

Answer: (a)

17. Which string is not accepted by the following FSA?

[WBUT 2013]



- a) 00111
- b) 00110
- c) 01010
- d) 11010

Answer: (a)

18. Regular sets are closed under

[WBUT 2014, 2015]

- a) Union
- b) Concatenation
- c) Kleene Closure
- d) All of the above

Answer: (d)

19. Regular grammar is

[WBUT 2017]

- a) context free grammar
- b) context sensitive grammar
- c) non-context grammar
- d) none of these

Answer: (a)

20. Regular expression is accepted by

[WBUT 2017]

- a) Finite automata
- b) Mealy machine
- c) Pushdown automata
- d) all of these

Answer: (d)

21. Pumping Lemma for Regular Expression is used to prove that

[WBUT 2018]

- a) Certain sets are Regular
- b) Certain sets are not Regular
- c) Certain Regular Grammar produce Regular Expression
- d) Certain Regular Grammar does not produce Regular Expression

Answer: (b)

Short Answer Type Questions

1. What is a regular expression? Give a complete procedure to convert a regular expression into its equivalent NFA. Using that procedure convert $01[((10)^*+111)^*+0]^*$ into its equivalent NFA.
OR,

[WBUT 2003]

What is regular expression?

[WBUT 2013]

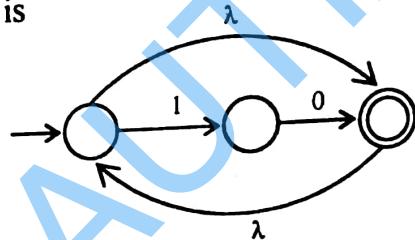
POPULAR PUBLICATIONS

Answer:

Let

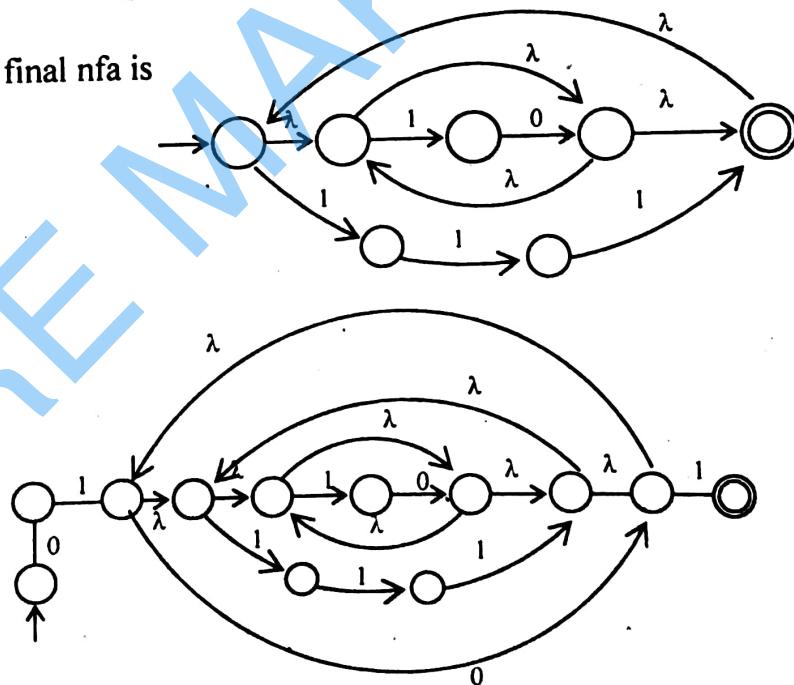
- i) $r_1 = (10)^*$
- ii) $r_2 = (r_1 + 11)^*$
- iii) $r_3 = (r_2 + 0)^*$
- iv) $r_4 = r_3 1$
- v) $r_5 = 01r_4$

Corresponding nfa of regular expression r_1 is



Corresponding nfa of regular expression r_2 is

In this way the final nfa is



2. State Myhill-Nerode theorem.

[WBUT 2005, 2007, 2008, 2010]

OR,

State Myhill-Nerode theorem with the definition of equivalent relation and Invariance.

[WBUT 2013]

Answer:

Statement of the Theorem

The Myhill-Nerode Theorem says the following three statements are equivalent:

FORMAL LANGUAGE AND AUTOMATA THEORY

1. The set L , a subset of Σ^* , is accepted by a DFA. (We know this means L is a regular language.)
2. L is the union of some of the equivalence classes of a right invariant (with respect to concatenation) equivalence relation of finite index.
3. Let equivalence relation RL be defined by: $x RL y$ if and only if for all $z \in \Sigma^*$, $xz \in L$ exactly when $yz \in L$. Then RL is of finite index.

3. Prove that the language $L = \{0^n 1^n \mid n > 0\}$ is not regular. [WBUT 2006, 2007, 2012]

Answer:

Suppose L is regular. There exists a finite state automaton M which accepts L . Suppose M has k states. Let $w = a^k b^k$. Then $|w| > k$. By the pumping Lemma $w = xyz$ where y is not empty and $w_2 = xy^2z$ is also accepted by M . If y consists of only a 's or only b 's the w_2 will not have the same number of a 's as b 's. If y contains both a 's and b 's, then w_2 will have a 's followed b 's. In either case w_2 does not belong to L , which is contradiction. Thus L is not regular.

4. Prove that following identity: $r(s + t) = rs + rt$.

[WBUT 2007]

Answer:

Suppose R , S and T denote the languages for r , s , and t , respectively.

Now, $L(r(s + t)) = S \cup T$.

Hence $L(r(s + t)) = R(S \cup T) = RS \cup RT$ [from Set theory] = $L(rs + rt)$

Hence $L(r(s + t)) = L(rs + rt)$. QED.

5. Construct a regular grammar G generating the regular set represented by

$$P = a^* b(a+b)^*$$

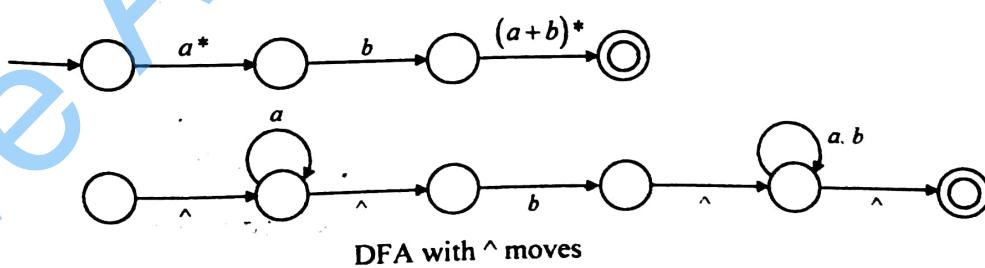
[WBUT 2007, 2010, 2016]

Answer:

Construct a regular grammar G generally the regular set represented by

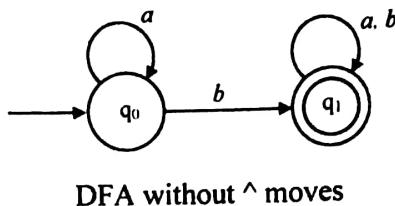
$$P = a + b(a+b)^*$$

We construct the DFA corresponding to P using the construction is given the following figure.



POPULAR PUBLICATIONS

After eliminating \wedge -moves, we get the DFA straight away,



Let $G = \{(A_0, A_1), (a, b), P, A_0\}$ where P is given by

$$\begin{array}{ll} A_0 \rightarrow aA_0 & A_1 \rightarrow aA_1 \\ A_0 \rightarrow bA_1 & A_1 \rightarrow bA_1 \\ A_0 \rightarrow b & A_1 \rightarrow a \\ & A_1 \rightarrow b \end{array}$$

G is required regular grammar.

6. Let R be an equivalence relation in $\{0\}^*$ with the following equivalence classes:

$$[\]_R = \{0\}^0$$

$$[0]_R = \{0\}^1$$

$$[00]_R = \{0\}^2 \cup \{0\}^3 \cup \{0\}^4 \dots \dots \dots$$

- Show that R is a right invariant.

[WBUT 2007, 2014, 2015]

Answer:

If R is a right-invariant the for all x, y, z $xRy \Rightarrow xzRyz$.

Clearly, $[\]_R$ has the null-string, $[0]_R$ has only the string 0 and $[00]_R$ has strings 00, 000, 0000, etc.

Suppose x and y are in $[\]_R$. Then, $x = y = \text{null}$. Hence $xz = z$ and $yz = z$ and zRz is trivially true.

Suppose x and y are in $[0]_R$. Then $x = y = 0$. If z is from $[\]_R$, then $z = \text{null}$, i.e., $xz = x$ and $yz = y$. Hence $xzRyz$. If z is from $[0]_R$, then $z = 0$. Thus, $xz = 00$ and $yz = 00$. Hence $xz = yz$ implying that $xzRyz$. Finally, if z is from $[00]_R$, z has at least two 0-s. Therefore xz as well as yz have at least three zeroes and hence both belong to $[00]_R$, i.e., $xz R yz$.

Suppose x and y are in $[00]_R$. Then both x and y have at least two 0-s. Hence irrespective of what z is, xz as well as yz will have at least two 0-s and thus both belong to $[00]_R$. Thus $xzRyz$.

So we see that for all cases $xRy \Rightarrow xzRyz$. Thus R is right invariant.

7. The set $L = \{a^i b^j c^k \mid \text{where } i, j, k \text{ are integer and } i, j, k \geq 1\}$. Is L regular? Justify your answer.

Answer:

Suppose L is regular

Let n be given by pumping lemma.

[WBUT 2008]

Let $w = \xleftarrow{i} \xrightarrow{i} \xleftarrow{j} \xrightarrow{k}$
 aaaa.....abb.....bb....bbb.....bcc....cc.....c
 $\underbrace{\hspace{1cm}}_x \quad \underbrace{\hspace{1cm}}_y \quad \underbrace{\hspace{1cm}}_z$

Be broken up as shown $|y| > 0$

Hence $xy^r z \in L$, $r \geq 0$ by pumping lemma.

$$\therefore w_1 = x z = \xleftarrow{k} 00 \dots 0 \xleftarrow{p < n} 11 \dots 1 \xleftarrow{n} 22 \dots 2$$

Let $r = 0$ has less 1's than 2's, thus leading to a contradiction.

8. Show that $L = \{a^p \mid p \text{ is prime}\}$ is not regular.

[WBUT 2009]

Answer:

$$L = \{a^p : P \text{ is Prime}\}$$

Let L be regular. Let "adversary" choose n .

$$\text{Let } z = a^p, \text{ where } P > n. \text{ Clearly } |z| \geq n.$$

Let adversary choose a decomposition uvw satisfying the conditions of pumping lemma.

$$\text{Let } |u| = p, |v| = q \text{ and } |w| = r$$

So we have:

$$q > 0$$

$$p + q < n$$

$$p + q + r = P$$

By pumping lemma, $uv^i w$ is in L . In particular, let us choose $i = q^p$ and let $y = uv^{q^p} w$.

By pumping lemma, y should be in L .

$$\text{Thus } |y| = p + q^* q^p + r$$

By Fermat's theorem, $q^p = s * P + q$ for some integer s .

Hence:

$$|y| = p + s * P + q + r = s * P + (p + q + r) = (s + 1) * P$$

Since $(s + 1) * P$ is not a prime, y is not in L , leading to a contradiction. So L is not regular.

[WBUT 2010]

9. a) State the pumping lemma for regular language.

b) Using pumping lemma prove that the set $L = \{0^i 1^i \mid i \geq 1\}$ is not regular.

Answer:

a) Formal statement of the Pumping Lemma

Let L be a infinite Regular Language. Then there exists some positive integer n such that

any $z \in L$ with $|z| \geq n$

Can be decomposed as

$$z = uvw$$

with $|uv| \leq n$,

POPULAR PUBLICATIONS

and $|v| \geq 1$,

such that $z_i = uv^i w$,

is also in L for all $i=0,1,2,\dots$

The two commonest ways to use the Pumping Lemma to prove a language is NOT regular are:

- show that there is no possible n for the (there exists n), this is usually accomplished by showing a contradiction such as $(n+1)(n+1) < n^*n+n$
- show there is no way to partition z into u, v and w such that $uv^i w$ is in L

b) $L = \{a^n b^n \mid n > 0\}$

Let L be regular. Let "adversary" choose n.

We choose $z = a^n b^n$. Obviously, $|z| \geq n$

Now, let adversary choose decomposition $z = uvw$ such that $|uv| \leq n$ and $|v| > 0$.

We note that whatever be the decomposition, the string uv consists only of 0-s and v has at least one 0. Therefore, for $i > 1$, uv^i has more 0-s than uv and hence the string $uv^i w$ has more 0-s at the beginning than there are 1-s at the end, allowing us to conclude that $uv^i w$ is not in L. Hence L is not regular. QED.

Note: Similar logic can be applied to prove the languages

$L = \{a^n b^k \mid n > 0, k > n\}$ non-regular.

10. What are Kleene Closure and Positive Closure? Give example for both.

[WBUT 2011, 2012]

OR,

What is Kleene's star? Give example. What is positive closure? Give example.

[WBUT 2016]

Answer:

For a Regular expression R, Kleene closure, denoted by R^* , is defined to be the expression obtained by concatenating zero or more

R -s. Thus, $R^* = \epsilon | R | RR | RRR | \dots$, where the vertical bar means disjunction. A Positive closure, denoted by R^+ , is defined to be the expression on the other hand is obtained by concatenating one or more R -s. Thus, $R^+ = \epsilon | RR | RRR | \dots$

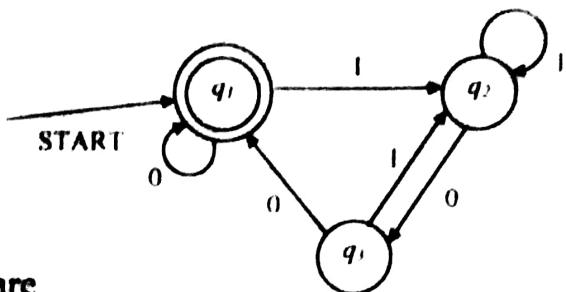
Example:

Suppose $R = a | bb$ Then,

$$R^* = \epsilon | a | bb | aa | abb | bba | bbbb |$$

$$R^+ = a | bb | aa | abb | bba | bbbb |$$

11. Give the Regular Expression for the DFA using Arden Theorem.
 [WBUT 2011, 2012, 2016]



Answer:

The state equations are

$$q_1 = \epsilon + q_1 0 + q_2 0 \quad \dots(1)$$

$$q_2 = q_1 1 + q_2 1 + q_3 0 \quad \dots(2)$$

$$q_3 = q_2 0 \quad \dots(3)$$

Substituting (3) in (2) we get:

$$q_2 = q_1 1 + q_2 (1 + 01) \quad \dots(4)$$

We know that if $R = Q + RP$, then $R = QP^*$

Hence, from (4),

$$q_2 = q_1 1 (1 + 01)^* \quad \dots(5)$$

Substituting (3) and (5) in (1)

We get:

$$q_1 = \epsilon + q_1 0 + q_2 00$$

$$= \epsilon + q_1 (0 + 1(1 + 01)^* 00)$$

$$\text{or } q_1 = \epsilon (0 + 1(1 + 01)^* 00)^*$$

$$= (0 + 1(1 + 01)^* 00)^*$$

Which is the RE for the given transition diagram.

12. Using Pumping Lemma show that $L = \{a^n b^n : n \geq 0\}$ is not regular.

[WBUT 2011, 2012]

Answer:

Let L be regular. Let "adversary" choose n .

We choose $z = a^n b^n$. Obviously, $|z| \geq n$

Now, let adversary choose decomposition $z = uvw$ such that $|uv| \leq n$ and $|v| > 0$.

We note that whatever be the decomposition, the string uv consists only of 0-s and v has at least one 0. Therefore, for $i > 1$, uv^i has more 0-s than uv and hence the string $uv^i w$ has more 0-s at the beginning than there are 1-s at the end, allowing us to conclude that $uv^i w$ is not in L . Hence L is not regular. QED.

Note: Similar logic can be applied to prove the languages

$$L = \{a^n b^k : n > 0, k > n\} \text{ non-regular.}$$

POPULAR PUBLICATIONS

Suppose L is regular. There exists a finite state automation M which accepts L . Suppose M has k states. Let $w = a^k b^k$. Then $|w| > k$. By the pumping Lemma $w = xyz$ where y is not empty and $w_2 = xy^2z$ is also accepted by M . If y consists of only a 's or only b 's the w_2 will not have the same number of a 's as b 's. If y contains both a 's and b 's, then w_2 will have a 's followed b 's. In either case w_2 does not belong to L , which is contradiction. Thus L is not regular.

13. What will be regular expression over the alphabet {a, b}, for the language $L = \{a^n b^m : n \geq 4, m \leq 3\}$? [WBUT 2013]

Answer:

The regular expression over the alphabet {a, b}, for the language

$L = \{a^n b^m : n \geq 4, m \leq 3\}$ is as follows : Separate into cases $m = 0, 1, 2, 3$. Generate 4 or more a 's, followed by the requisite number of b 's.

So, the regular expression for the following language is $aaaa^*(\lambda + b + bb + bbb)$.

14. Construct the language for the grammar $G = (\{S\}, \{a, b\}, S, P)$, with P given by

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

[WBUT 2018]

Answer:

$G = (\{S\}, \{a, b\}, S, P)$ with P given by

$$S \rightarrow aSb, S \rightarrow \lambda$$

$$\therefore S \rightarrow a \underline{S} b \rightarrow a > b \rightarrow ab$$

$$\therefore S \rightarrow a \underline{S} b \rightarrow aa \underline{S} bb \rightarrow aa > bb \rightarrow aabb$$

$$\therefore S \rightarrow a \underline{S} b \rightarrow aa \underline{S} bb \rightarrow aaa \underline{S} bbb \rightarrow aaabbb$$

So, the Language is \rightarrow "Equal no. of a 's followed by equal no. of b 's with Null".

$$\therefore L(G) = \{a^n b^n ; n \geq 0\} \quad (\text{Ans.})$$

Long Answer Type Questions

1. a) Construct a minimum state automaton equivalent to a given automation M whose transition table is given below: [WBUT 2007]

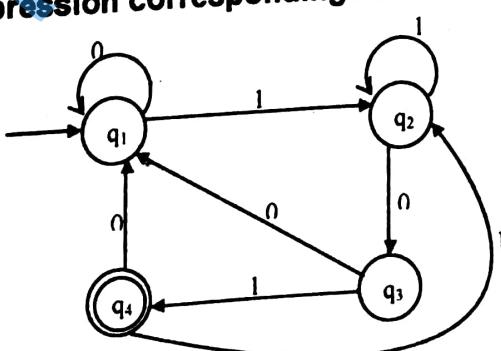
State	Input	
	a	b
$\rightarrow q_0$	q_0	q_3
q_1	q_2	q_5
q_2	q_3	q_4
q_3	q_0	q_5
q_4	q_0	q_6
q_5	q_1	q_4
q_6	q_1	q_3

Answer:

q_1	x					
q_2	x	x,				
q_3	x	x	x			
q_4	x	x	x	x		
q_5	x	x	x	x	x	
q_6	x	x	x	x	x	x
q_0	q_1	q_2	q_3	q_4	q_5	

Since all states are distinguishable, the DFA is already minimized.

b) Find the regular expression corresponding to the following figure: [WBUT 2007]



Answer:

$$0^* 1^+ (0^+ 1^+)^* 0^* 1$$

[WBUT 2009, 2010, 2018]

2. a) What is regular language?

Answer:

A regular language is a formal language defined over a finite alphabet such that every string in the language can be:

POPULAR PUBLICATIONS

- a) Accepted by some NFA/DFA
- b) Described by a regular expression
- c) Generated by a Regular (i.e. Type-3) Grammar.

b) Find regular expressions over $\Sigma = \{a, b\}$ for the languages defined as follows:

- i) $L1 = \{a^m b^m : m > 0\}$
- ii) $L2 = \{a^{2n} b^{2m+1} \mid n \geq 0, m \geq 0\}$
- iii) $L3 = \{b^m ab^n : m > 0, n > 0\}$

[WBUT 2009, 2010, 2013]

Answer:

i) Not possible since the language is not regular.

ii) $(aa)^* b(bb)^*$

iii) $bb^* abb^*$

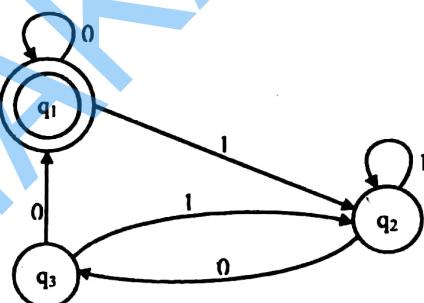
c) Find the regular expression for following transition graph.

[WBUT 2009, 2010]

OR,

Construct the regular expression corresponding to the state diagram given below:

[WBUT 2013]



Answer:

The state equations are

$$q_1 = \epsilon + q_1 0 + q_3 0 \quad \dots(1)$$

$$q_2 = q_1 1 + q_2 1 + q_3 1 \quad \dots(2)$$

$$q_3 = q_2 0 \quad \dots(3)$$

Substituting (3) in (2) we get:

$$q_2 = q_1 1 + q_2 (1 + 01) \quad \dots(4)$$

We know that if $R = Q + RP$, then $R = QP^*$

Hence, from (4),

$$q_2 = q_1 1 (1 + 01)^* \quad \dots(5)$$

Substituting (3) and (5) in (1)

We get:

$$q_1 = \epsilon + q_1 0 + q_2 00$$

$$= \epsilon + q_1 (0 + 1(1 + 01)^* 00)$$

FORMAL LANGUAGE AND AUTOMATA THEORY

$$\text{or } q_1 = \epsilon(0+1(1+01)^*00)^*$$

$$= (0+1(1+01)^*00)^*$$

Which is the RE for the given transition diagram.

- 3. Construct the minimum state automata equivalent to given automata M defined below:** [WBUT 2010]

Σ states	a	b
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_6
$*q_2$	q_2	q_0
q_4	q_5	q_7
q_5	q_6	q_2
q_6	q_4	q_6
q_7	q_2	q_6

(* q_2 indicates that q_2 is the final state)

Answer:

q_1	x					
q_2	x	x				
q_4		x	x			
q_5	x	x	x	x		
q_6	x	x	x	x	x	
q_7	x		x	x	x	
	q_0	q_1	q_2	q_4	q_5	q_6

$$\begin{aligned} & \{q_0, q_4\}, \{q_1, q_7\}, \{q_2\}, \{q_5\}, \{q_6\} \\ & \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ & \rightarrow A \quad B \quad C \quad D \quad E \end{aligned}$$

	a	b
$\rightarrow A$	D	B
B	C	E
C^*	C	A
D	E	C
E	A	E

- 4. Using Pumping lemma prove that $L = \{a^i \mid i \geq 1\}$ is not regular. Construct Finite Automata equivalent to the Regular Expression. $L = ab(aa+bb)(a+b)^*b$.** [WBUT 2014]

POPULAR PUBLICATIONS

Answer:

Suppose L is regular and we get a contradiction. Let n be the number of states in FA accepting L .

Let $w = a^{n^3}$. Then $|w| = n^3 > n$ by pumping lemma we can write

$$w = xyz \text{ with } |xy| \leq n \text{ and } |y| > 0$$

Consider $xy^3z \cdot |xy^3z| = |x| + 3|y| + |z| > |x| + |y| + |z|$

as $|y| > 0$. This means $n^3 = |xyz| = |x| + |y| + |z| < |xy^3z|$

as $|xy| \leq n, |y| \leq n$. Therefore

$$|xy^3z| = |x| + 3|y| + |z| \leq n^3 + n$$

$$\text{i.e., } n^3 < |xy^3z| \leq n^3 + n \leq n^3 + 3n^2 + 3n + 1$$

Hence, $|xy^3z|$ strictly lies between n^3 and $(n+1)^3$ but it is not equal to any one of them.

Thus $|xy^3z|$ is not perfect case and so $xy^3z \notin L$. But by pumping lemma $xy^3z \in L$. This is a contradiction.

5. Using Pumping lemma prove that $L = \{a^n b^n \mid n \geq 1\}$ is not regular. [WBUT 2017]

Answer:

At first, we assume that L is regular and n is the number of states.

Let $w = a^i b^i$. Thus $|w| = 2i \geq i$.

By pumping lemma, let $w = xyz$, where $|xy| \leq i$.

Let $x = a^p, y = a^q$, and $z = a^r b^i$, where $p + q + r = n, p \neq 0, q \neq 0, r \neq 0$. Thus $|y| \neq 0$.

Let $k = 2$. Then $xy^2z = a^p a^{2q} a^r b^i$.

Number of $a = (p + 2q + r) = (p + q + r) + q = n + q$

Hence, $xy^2z = a^{i+q} b^i$. Since $q \neq 0$, xy^2z is not of the form $a^i b^i$.

Thus, xy^2z is not in L . Hence L is not regular.

CONTEXT FREE GRAMMARS

Multiple Choice Type Questions

1. A grammar in CNF may contain productions like

a) $A \rightarrow B$ b) $\cancel{A} \rightarrow BC$

c) $A \rightarrow aB$

d) $A \rightarrow aBCD$

[WBUT 2007]

Answer: (b)

2. $L = \{a^n b^n c^n, \text{ where } n \in \mathbb{N}\}$ is

a) regular

b) context free but not regular

c) context sensitive but not context free

d) none of these

[WBUT 2009]

Answer: (c)

3. Which of the following productions is in CNF?

a) $S \rightarrow aA$

b) $SA \rightarrow AS$

c) $\cancel{S} \rightarrow AB$

[WBUT 2009]

d) All of these

Answer: (c)

4. Which is more suitable for an Ambiguous Grammar?

a) All ambiguities can be removed

b) Ambiguity can be removed by setting priority

c) Only inherent ambiguity can be removed

d) There is no suitable rule for removing ambiguity

[WBUT 2009]

Answer: (c)

5. The class of CFG is not closed under

a) concatenation

b) intersection

c) union

d) repeated concatenation

[WBUT 2011]

Answer: (b)

6. Consider the CFG

$$X \rightarrow XY$$

$$X \rightarrow zX / bX / a$$

$$Y \rightarrow Ya / Yb / b$$

Any string of terminals, which can be generated by the CFG

b) ends with a

a) has at least one b

d) has at least 2 a's

c) has no consecutive a's or b's

[WBUT 2011]

Answer: (d)

7. A grammar that produces more than one parse tree for some sentence is said to be

a) contiguous

b) ambiguous

c) unambiguous

[WBUT 2011]

d) regular

Answer: (b)

POPULAR PUBLICATIONS

8. A CFG, $S \rightarrow aS/bS/a/b$, is equivalent to

[WBUT 2011]

- a) $(a+b)^*$ b) $(a+b)(a+b)^*$ c) $(a+b)^*(a+b)$

d) all of these

Answer: (b)

9. $(P+Q)^* = ?$

[WBUT 2012]

- a) $(P^* + Q^*)$ b) $P^* + Q^*$ c) $(P^*Q^*)^*$

d) both (a) and (c)

Answer: (d)

10. Which of the following production is in CNF?

[WBUT 2013]

- a) $S \rightarrow aA$ b) $SA \rightarrow AS$ c) $S \rightarrow AB$

d) All of these

Answer: (c)

11. Consider the following language: $L = \{a^n b^n c^n d^n \mid n \geq 1\}$

[WBUT 2013]

L is

- a) CFL but not regular
c) Regular

b) CSL but not CFL

d) Type 0 language but not type 1

Answer: (b)

12. Consider the grammar $S \rightarrow aSAb/\epsilon$

$A \rightarrow bA/\epsilon$

The grammar generates the strings of the form $a^i b^j$ for some $i, j \geq 0$. What are the conditions of the values of i and j?

[WBUT 2014, 2015]

- a) $i = j$ b) $j \leq 2i$ c) $j \geq 2i$

d) $i \leq j$

Answer: (d)

13. Which of the following common in both CNF & GNF? [WBUT 2014, 2015, 2017]

- a) (NT) \rightarrow (Single T) (String of NT)
c) (NT) \rightarrow (String of NT)

b) (NT) \rightarrow (Single of exactly two NT)

d) (NT) \rightarrow (Single T)

Answer: (d)

14. Consider the languages:

$$L_1 = \{WW^R \mid W \in \{0,1\}^*\}$$

$$L_2 = \{W \# W^R \mid W \in \{0,1\}^*\}, \text{ where } \# \text{ is a special symbol}$$

$$L_3 = \{WW \mid W \in \{0,1\}^*\}$$

Which one of the following is true?

[WBUT 2014, 2015, 2017]

- a) L_1 is a deterministic CFL
b) L_2 is a deterministic CFL
c) L_3 is a CFL, but not a deterministic CFL
d) L_3 is a deterministic CFL

Answer: (b)

15. A context free grammar is not closed under

- a) Product
- ~~c) Complementation~~
- b) Union
- d) Kleen star

[WBUT 2016]

Answer: (c)

16. Context sensitive grammar can be recognized by

- a) deterministic push down machine
- b) non-deterministic push down machine
- c) FSM
- ~~d) linearly bounded memory machine~~

[WBUT 2016]

Answer: (d)

17. Which of the following is most general phase structured grammar?

- a) Regular
- ~~b) Context sensitive~~
- c) PDA
- d) none of these

[WBUT 2016]

Answer: (b)

18. CFL is _____ language.

- a) Type 0
- b) Type 1
- ~~c) Type 2~~
- d) Type 3

[WBUT 2018]

Answer: (c)

Short Answer Type Questions

1. Convert the following context free grammar into an equivalent grammar in CNF:

$$S \rightarrow aAbB$$

$$A \rightarrow abAB/aAA/a$$

$$B \rightarrow bBaA/aBB/b$$

[WBUT 2006, 2007, 2010]

Answer:

Let us assume that a and b may be represented as C and D respectively.

So, the products are as follows:-

$$S \rightarrow CABD$$

$$A \rightarrow CDAB|CAA|a$$

$$B \rightarrow DBCA|CBB|b$$

$$C \rightarrow a$$

$$D \rightarrow b$$

Let us assume that AA be E, BB be F, CA be G and DB be H. So the productions are:-

$$S \rightarrow GH$$

$$A \rightarrow GH$$

$$B \rightarrow GH$$

$$C \rightarrow a$$

$$D \rightarrow b$$

$$E \rightarrow AA$$

$$F \rightarrow BB$$

$$G \rightarrow CA$$

POPULAR PUBLICATIONS

H→BD

2. a) What do you mean by a sub-tree of a derivation tree?

[WBUT 2008]

Answer:

A subtree of a derivation tree is a *partial derivation tree* such that, for any node of the subtree, either all of its children are also in the subtree, or none of them are.

b) Consider G whose productions are $S \rightarrow aAS/a$, $A \rightarrow SbA/SS/ba$. Show that $S \rightarrow aabbaa$ by constructing a derivation tree, by right most derivation, whose yield is aabbaa.

[WBUT 2008]

Answer:

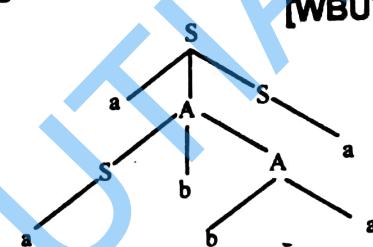
$S \rightarrow aAS$

$\rightarrow aSbAS$

$\rightarrow aabAS$

$\rightarrow aabbaS$

$\rightarrow aabbaa$ (proved).



3. Reduce the following grammars to GNF:

[WBUT 2008]

$S \rightarrow A0$, $A \rightarrow 0B$, $B \rightarrow 0A$, $B \rightarrow 1$

Answer:

As it is already in GNF form, so there is no requirement for converting it into GNF.

4. Write the CFG for the language

[WBUT 2008, 2012]

$L = \{0^i 1^j 2^k \mid i=j \text{ or } j=k\}$.

Answer:

$S \rightarrow AB|B$

$A \rightarrow 0A|0$

$B \rightarrow 1B2|12$

5. Let G be a grammar

[WBUT 2009, 2010]

$s \rightarrow 0B \mid 1A$, $A \rightarrow 0 \mid 0S \mid 1AA$, $B \rightarrow 1 \mid 1S \mid 0BB$.

For the string 00110101, find

- i) leftmost derivation
- ii) rightmost derivation
- iii) derivation tree

OR,

Convert the following Context-free grammar into an equivalent grammar in CNF

$S \rightarrow 1A \mid 0B$

$A \rightarrow 1AA \mid 0S \mid 0$

$B \rightarrow 0BB \mid 1S \mid 1$

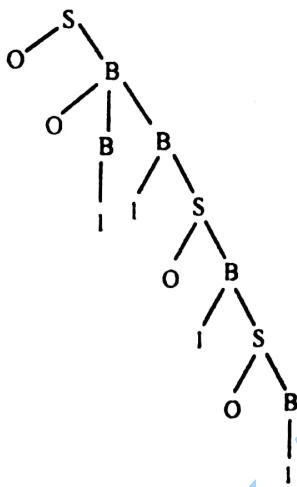
[WBUT 2011]

Answer:

- i) $S \Rightarrow 0\underline{B} \Rightarrow 00\underline{BB} \Rightarrow 001\underline{B} \Rightarrow 0011\underline{S} \Rightarrow 00110\underline{B} \Rightarrow 001101\underline{S}$
 $\Rightarrow 0011010\underline{B} \Rightarrow 00110101$

ii) $S \Rightarrow 0\underline{B} \Rightarrow 00\underline{BB} \Rightarrow 00B\underline{1S} \Rightarrow 00B10\underline{B} \Rightarrow 00B101\underline{S}$
 $\Rightarrow 00B1010\underline{B} \Rightarrow 00B10101 \Rightarrow 00110101$

iii)



[WBUT 2011]

6. Let G be the grammar

$$S \rightarrow aB / ba, A \rightarrow a / aS / bAA, B \rightarrow b / bS / aBB$$

For the string *aaabbabbba*, find

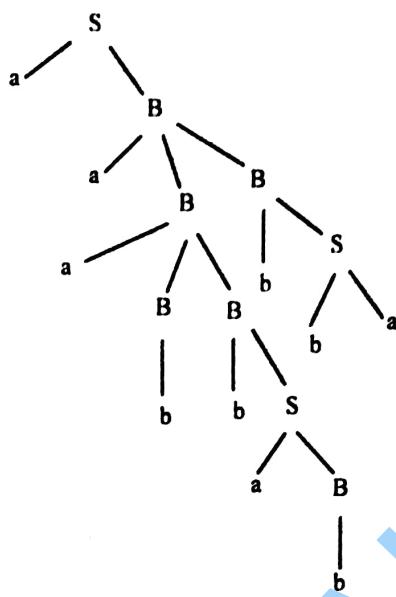
- a) leftmost derivation
- b) rightmost derivation
- c) parse tree.

Answer:

a) $S \Rightarrow a\underline{B} \Rightarrow aa\underline{BB} \Rightarrow aaa\underline{BBB} \Rightarrow aaab\underline{BB} \Rightarrow aaabb\underline{SB}$
 $\Rightarrow aaabba\underline{BB} \Rightarrow aaabbab\underline{B} \Rightarrow aaabbabb\underline{S} \Rightarrow aaabbabbba$

b) $S \Rightarrow a\underline{B} \Rightarrow aa\underline{B}\underline{B} \Rightarrow aaBb\underline{S} \Rightarrow aaBbb\underline{a} \Rightarrow aaaB\underline{B}\underline{bba}$
 $\Rightarrow aaaBb\underline{S}\underline{bba} \Rightarrow aaaBba\underline{B}\underline{bba} \Rightarrow aaaBbab\underline{bba} \Rightarrow aaabbabbba$

c)



7. What is Greibach Normal Form (GNF) for Context Free grammar?

Convert the following grammar into GNF

[WBUT 2011, 2012, 2016]

$S \rightarrow ABb/a$

$A \rightarrow aaA/B$

$B \rightarrow bAb$

Answer:

A Context Free Grammar is said to be in Greibach Normal Form if each rule in the grammar has one of the following forms:

1. $A \rightarrow aA_1A_2 \dots A_n$

2. $A \rightarrow a$

where, $A, A_i \in V_N, i = 1, 2, \dots, n$ and $a \in \Sigma$

If ϵ belongs to the language of the grammar, the rule $A \rightarrow \epsilon$ (where S is the start symbol) is also permitted.

Greibach's Theorem: Every Context Free Language can be generated by a Context Free Grammar which is in Greibach Normal Form GNF.

The above theorem implies that given a grammar G (which generates the language $L(G)$), the grammar can be converted to an equivalent grammar G' which is in GNF such that $L(G') = L(G)$.

For Step-0, we need to add a rule $Y \rightarrow b$. So we start the main process of conversion of the grammar to GNF with the rules:

$S \rightarrow ABy/a$

$A \rightarrow aaA/B$

$$B \rightarrow bBY$$

$$Y \rightarrow b$$

There is no immediate left recursion in the grammar.

We select the order S, A, B, Y . Clearly there is no need for nonterminal substitution.

So, the final grammar in GNF is:

$$Y \rightarrow b$$

$$B \rightarrow bBY$$

$$A \rightarrow aaA \mid bBY$$

$$S \rightarrow aaBY \mid bBY \mid BY \mid a$$

8. Find a GNF grammar equivalent to the following CFG:

[WBUT 2012]

$$A_1 \rightarrow A_2 A_3$$

$$A_2 \rightarrow A_3 A_1 \mid b$$

$$A_3 \rightarrow A_1 A_2 \mid a$$

Answer:

Given:

$$A_1 \rightarrow A_2 A_3$$

$$A_2 \rightarrow A_3 A_1 \mid b$$

$$A_3 \rightarrow A_1 A_2 \mid a$$

The grammar is already in CNF.

Round-1

~~$$A_1 \rightarrow A_2 A_3$$~~

~~$$A_2 \rightarrow A_3 A_1 \mid aA_1 \mid b$$~~

~~$$A_3 \rightarrow A_1 A_2 \mid a$$~~

Round-2

$$A_1 \rightarrow A_2 A_3 A_1 A_3 \mid aA_1 A_3 \mid bA_3$$

But this requires removal of left recursion

$$A_1 \rightarrow aA_1 A_3 Z \mid bA_3 Z \mid aA_1 A_3 \mid bA_3$$

$$Z \rightarrow A_1 A_2 A_1 A_3 Z \mid A_1 A_2 A_1 A_3$$

$$A_2 \rightarrow A_1 A_2 A_1 \mid aA_1 \mid b$$

$$A_3 \rightarrow A_1 A_2 \mid a$$

POPULAR PUBLICATIONS

Round-3

$$\begin{aligned}
 A_1 &\rightarrow aA_1A_3Z \mid bA_3Z \mid aA_1A_3 \mid bA_3 \\
 Z &\rightarrow aA_1A_3ZA_2A_1A_3Z \mid \\
 &\quad bA_3ZA_2A_1A_3Z \mid \\
 &\quad aA_1A_3A_2A_1A_3Z \mid \\
 &\quad bA_3A_2A_1A_3Z \mid \\
 &\quad aA_1A_3ZA_1A_2A_1A_3Z \mid \\
 &\quad bA_3ZA_2A_1A_3Z \mid \\
 &\quad aA_1A_3A_2A_1A_3Z \mid \\
 &\quad bA_3A_2A_1A_3Z \mid \\
 A_2 &\rightarrow aA_1A_3ZA_2A_1 \mid bA_3ZA_2A_1 \mid \\
 &\quad aA_1A_3A_2A_1 \mid bA_3A_2A_1 \mid aA_1 \mid b \\
 A_3 &\rightarrow aA_1A_3ZA_2 \mid bA_3ZA \mid aA_1A_3A_2 \mid bA_3A_2 \mid a
 \end{aligned}$$

The grammar is now in GNF.

9. Find the Context Free Grammar for the following language $L = \{a^n b^{2n} c^m : n, m \geq 0\}$.

[WBUT 2013]

Answer:

Let us assume $CFG \Rightarrow G = (V_n, V_t, P, S)$

$V_n = \{S, A, B\}$, non terminals

$V_t = \{a, b, c\}$, terminals

P (production rule) is defined as follows:

$$\begin{aligned}
 S &\rightarrow AB \\
 A &\rightarrow aAbb/\epsilon \\
 B &\rightarrow cB/\epsilon
 \end{aligned}$$

10. Construct grammar of the following.

- a) Construct the grammar for the language $a1bmcn$, where $1+m=n$ [WBUT 2014]
 b) all even integers up to 998. [WBUT 2014, 2017]

Answer:

- a) $S \rightarrow aAc ; A \rightarrow bBc | aAc | \epsilon$; $B \rightarrow bBc | \epsilon$
 b) $S \rightarrow AB | AAB | 0 | 2 | 4 | 6 | 8 ; A \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ; B \rightarrow 0 | 2 | 4 | 6 | 8$

11. Convert the following grammar into GNF.

$S \rightarrow AA/a$

$A \rightarrow SS/b$

[WBUT 2014, 2017]

Answer:

$$S \rightarrow AA|a$$

$$A \rightarrow SS \mid b$$

$$A \rightarrow AAS \mid aS \mid b$$

$$A \rightarrow aSZ \mid bZ \mid aS \mid b$$

$$Z \rightarrow ASZ \mid \text{epsilon}$$

So the GNF

$$S \rightarrow aSZA \mid bZA \mid aSA \mid bA \mid a$$

$$A \rightarrow aSZ \mid bZ \mid aS \mid b$$

$$Z \rightarrow aSZSZ \mid bZSZ \mid aSSZ \mid bSZ$$

12. a) What is parse tree?

[WBUT 2015]

b) Consider the CFG

$$S \rightarrow aaB$$

$$A \rightarrow bBb \mid \epsilon$$

$$B \rightarrow Aa$$

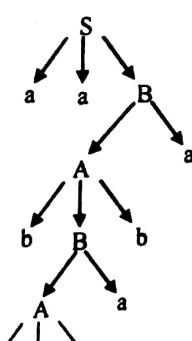
Find the Parse tree for the string aabbababa.

Answer:

a) A parse tree is an entity which represents the structure of the derivation of a terminal string from some non-terminal (not necessarily the start symbol). The definition is as in the book. Key features to define are the *root* $\in V$ and *yield* $\in \Sigma^*$ of each tree.

- For each $\sigma \in \Sigma$, there is a tree with root σ and no children; its yield is σ
- For each rule $A \rightarrow \epsilon$, there is a tree with root A and one child ϵ ; its yield is ϵ
- If t_1, t_2, \dots, t_n are parse trees with roots r_1, r_2, \dots, r_n and respective yields y_1, y_2, \dots, y_n , and $A \rightarrow r_1r_2\dots r_n$ is a production, then there is a parse tree with root A whose children are t_1, t_2, \dots, t_n . Its root is A and its yield is $y_1y_2\dots y_n$

b)



POPULAR PUBLICATIONS

[WBUT 2016]

13. Let G be the grammar $S \rightarrow aB \mid ba$, $B \rightarrow b \mid bS \mid aBB$.

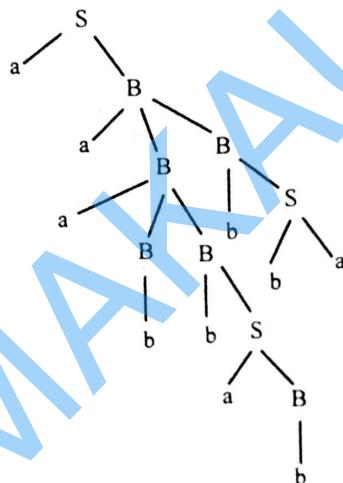
For the string aaabbabbba find:

- i) left most derivation
- ii) rightmost derivation
- iii) parse tree

Answer:

i) $S \rightarrow aB \rightarrow aaBB \rightarrow aaaBBB \rightarrow aaabbB \rightarrow aaabbaBB \rightarrow aaabbabB \rightarrow aaabbabbS \rightarrow aaabbabbba$

ii) $S \Rightarrow aB \Rightarrow aaBB \Rightarrow aaBbS \Rightarrow aaBbba \Rightarrow aaaBbbba \Rightarrow aaaBbSbba \Rightarrow aaaBbaBbba \Rightarrow aaaBbabbbba \Rightarrow aaabbabbba$



14. What do you mean by unit production? Remove unit productions from the grammar

$S \rightarrow AB$, $A \rightarrow a$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow b$.

[WBUT 2017]

Answer:

1st Part:

A unit production is a production $A \rightarrow B$ where both A and B are non-terminals. Unit productions are redundant and hence should be removed.

Repeat the following steps while there is a unit production

1. Select a unit production $A \rightarrow B$, such that there exist a production $B \rightarrow \alpha$, where α is a terminal
2. For every non-unit production, $B \rightarrow \alpha$ repeat the following step
 1. Add production $A \rightarrow \alpha$ to the grammar
 3. Eliminate $A \rightarrow B$ from the grammar

2nd Part:

There are 2 unit productions in the grammar –

$B \rightarrow C$

$C \rightarrow D$

For production $C \rightarrow D$ there is $D \rightarrow b$, so we add $C \rightarrow b$ and delete $C \rightarrow D$ from the grammar. Now we have $B \rightarrow C$, so we add a production $B \rightarrow b$ and remove $B \rightarrow C$ from the grammar.

Now we get the final grammar free of unit production as –

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$C \rightarrow b$$

Now we can see that C is an unreachable symbol and so to get a completely reduced grammar, we remove C from the CFG. Thus the final CFG is –

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

15. Construct grammar of the following:

For the language $a^n b^n$, where $n \geq 0$.

[WBUT 2017]

Answer:

$$L(G) = \{ab, aabb, aaabbb, \dots\}$$

$$G = \{N, T, P, S_i\}$$

$$N = \{S, A, B\}$$

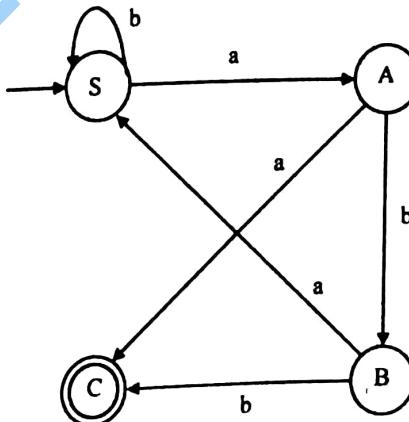
$$T = \{a, b\}$$

S_i = starting symbol = S

$$P = S \rightarrow aA, A \rightarrow aA \mid bB, B \rightarrow bB \mid \epsilon$$

16. Define Left Linear and Right linear grammar. Construct grammar for the following FA:

[WBUT 2017]



Answer:

1st Part:

Left Linear Grammar: In a grammar, if all productions are in the form –

$A \rightarrow B\alpha$ or $A \rightarrow \alpha$, where $A, B \in V_N$ and $\alpha \in \Sigma^*$, then the grammar is called Left Linear Grammar.

Example: $A \rightarrow Aa \mid Bb \mid b$

POPULAR PUBLICATIONS

Right Linear Grammar: In a grammar, if all productions are in the form –

$A \rightarrow \alpha B$ or $A \rightarrow \alpha$, where $A, B \in V_N$ and $\alpha \in \Sigma^*$, then the grammar is called Right Linear Grammar. Example: $A \rightarrow aA|bB|b$

2nd Part:

The grammar G can be formally written as a 4-tuple (N, T, S_t, P) where –

$N = \{S, A, B, C\}$; N is a set of variables or non-terminal symbols

$T = \{a, b\}$; T is a set of terminal symbols

S_t = Starting Symbol (S)

P = Production rules for terminals and non-terminals $- S \rightarrow aA|bS$, $A \rightarrow aC|bB$,

$B \rightarrow aS|bC$, $C \rightarrow \epsilon$.

[WBUT 2018]

17. Remove the UNIT productions from the following grammar:

$$S \rightarrow Aa|B$$

$$B \rightarrow A|bb$$

$$A \rightarrow a|bc|B$$

Answer:

Remove unit production,

$$S \rightarrow Aa|B$$

$$B \rightarrow A|bb$$

$$A \rightarrow a|bc|B$$

⇒ Here, three unit productions are present

$$S \rightarrow B, A \rightarrow B, B \rightarrow A$$

∴

$$S \rightarrow Aa|A|bb$$

$$\boxed{B \rightarrow a|bc|B|bb}$$

$$A \rightarrow a|bc|A|bb$$

We can remove the void production $B \rightarrow B$

So,

$$B \rightarrow a|bc|bb$$

$$S \rightarrow Aa|a|bc|bb$$

$$\boxed{A \rightarrow a|bc|a|bc|bb}$$

Same productions
can be removed.

∴ Final productions are →

$$B \rightarrow a|bc|bb$$

$$S \rightarrow Aa|a|bc|bb$$

$$A \rightarrow a|bc|bb$$

(Ans.)

Long Answer Type Questions

1. a) Let G be the grammar

$$\begin{aligned} S &\rightarrow aB / bA \\ A &\rightarrow a / aS / bAA \\ B &\rightarrow b / bS / aBB \end{aligned}$$

[WBUT 2006, 2009]

Find left most derivation for the string aaabbabbba.

- a) leftmost derivation
- b) rightmost derivation
- c) parse tree.

Answer:

Let us number the rules

- (1) $S \rightarrow aB$
- (2) $S \rightarrow bA$
- (3) $A \rightarrow a$
- (4) $A \rightarrow aS$
- (5) $A \rightarrow bAA$
- (6) $B \rightarrow b$
- (7) $B \rightarrow bS$
- (8) $B \rightarrow aBB$

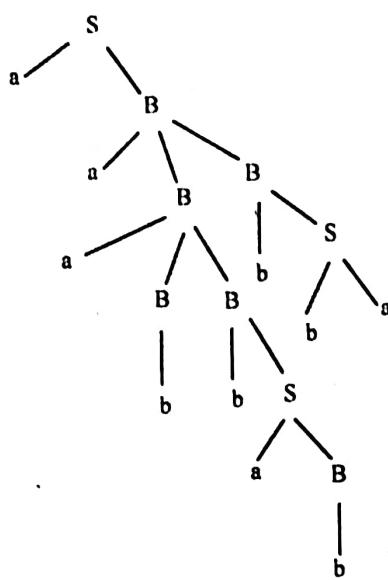
a) The left-most derivation (With \Rightarrow labeled with rule number) is:

$$\begin{aligned} S &\xrightarrow{(1)} a \underline{B} \xrightarrow{(8)} aa \underline{B} B \xrightarrow{(8)} aaa \underline{B} B B \xrightarrow{(6)} aaa b \underline{B} B \\ &\xrightarrow{(7)} aaa b b \underline{S} \xrightarrow{(1)} aaa b b a \underline{B} B \xrightarrow{(6)} aaa bb a b \underline{B} \\ &\xrightarrow{(7)} aaa bb a bb \underline{S} \xrightarrow{(2)} aaa bb a bbb \underline{A} \xrightarrow{(3)} aaa bb a bbb a \end{aligned}$$

$$\begin{aligned} b) S &\Rightarrow a \underline{B} \Rightarrow aa \underline{B} B \Rightarrow aaBb \underline{S} \Rightarrow aaBbba \Rightarrow aaaB \underline{B} bba \\ &\Rightarrow aaaBb \underline{S} bba \Rightarrow aaaBba \underline{B} bba \Rightarrow aaaBbabbbba \Rightarrow aaabbabbba \end{aligned}$$

POPULAR PUBLICATIONS

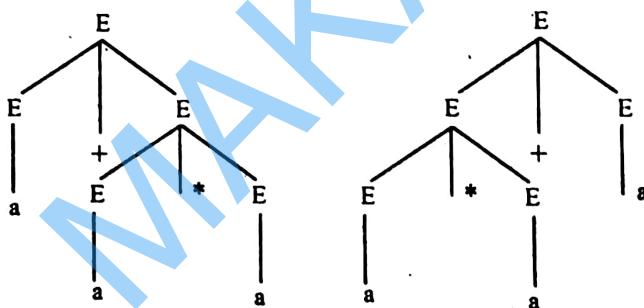
c)



2. a) $E \rightarrow E+E \mid E^*E \mid a$. Prove that the CFG with this production rule is ambiguous.
Remove the ambiguity from this grammar.

[WBUT 2008]

Answer:



b) $S \rightarrow AB; A \rightarrow a, B \rightarrow C/b, C \rightarrow D; D \rightarrow E, E \rightarrow a$.

remove the unit production.

$L = \{a^n b^n \mid n \geq 0\}$. Find a CFG to generate L^2 .

[WBUT 2008]

Answer:

Here unit productions are:

$B \rightarrow C$,

$C \rightarrow D$,

$D \rightarrow E$.

And original non-unit productions are:-

$S \rightarrow AB$,

$A \rightarrow a$,

$B \rightarrow b$,

$E \rightarrow a$.

Next right side of all unit production replace only terminal symbol so the productions are
 $S \rightarrow AB$,

$A \rightarrow a$,

$B \rightarrow b$,

$E \rightarrow a$.

Then the complete productions are:-

$S \rightarrow AB$,

$A \rightarrow a$,

$B \rightarrow a \mid b$,

$E \rightarrow a$.

Suppose $w = a^n b^n \in L$.

Then $awb = a^{n+1} b^{n+1} \in L$

Hence required CFG is:

$S \rightarrow awb \mid \epsilon$.

3. a) Convert grammars to Greibach Normal Form (GNF).

[WBUT 2009]

i) $S \rightarrow aSa \mid aSb \mid \epsilon$

ii) $S \rightarrow aSB \mid aSbS \mid \epsilon$

Answer:

Strictly speaking, none of the given grammars can be converted to GNF since ϵ belongs to the languages of both the grammars. However, we continue by ignoring the $S \rightarrow \epsilon$ productions and adding it to the converted grammar.

i) The converted grammar in GNF is

$$S \rightarrow aSA$$

$$S \rightarrow aSB$$

$$S \rightarrow \epsilon$$

$$A \rightarrow a$$

$$B \rightarrow b$$

ii) The given grammar is incomplete since there are NO B productions given.

b) Find a reduced grammar equivalent to the grammar

[WBUT 2009, 2014, 2017]

$$S \rightarrow aAa, A \rightarrow bBB, B \rightarrow ab, C \rightarrow ab$$

Answer:

Clearly C is a useless symbol. Eliminating C, we get,

$$S \rightarrow aAa, A \rightarrow bBB, B \rightarrow ab$$

Now we replace all occurrences of B on rhs of production using $B \rightarrow ab$, getting:

$$S \rightarrow aAa, A \rightarrow b ab ab$$

Now we similarly substitute A, getting

$$S \rightarrow a ba ba ba$$

which is the equivalent reduced grammar.

POPULAR PUBLICATIONS

4. a) Construct CFG for the following:

[WBUT 2013, 2016]

i) Palindrome for binary numbers

ii) $L = \{a^n b^n c^m d^m \mid m, n > 0\}$

iii) $L = \{a^n b^m \mid n \neq m\}$

b) Convert the following grammar to CNF

$$S \rightarrow aA / B / C / a$$

$$A \rightarrow aB / E$$

$$B \rightarrow aA$$

$$C \rightarrow cCD$$

$$D \rightarrow abd$$

c) Define non-generating and non-reachable symbols with example.

[WBUT 2013]

[WBUT 2013]

Answer:

a) i) Consider $L_{pal} = \{w \in \Sigma^* : w = w^R\}$

For example otto $\in L_{pal}$, madamimadam $\in L_{pal}$.

In finnish language e.g. saippuakauppias $\in L_{pal}$ ("soap- merchant")

Let $\Sigma = \{0,1\}$ and suppose L_{pal} were regular.

Let n be given by the pumping lemma. Then $0^n 1 0^n \in L_{pal}$. In reading 0^n the FA must make a loop. Omit the loop; contradiction.

Let's define L_{pal} inductively:

Basis: $\epsilon, 0$, and 1 are palindromes.

Induction: If w is a palindrome, so are $0w0$ and $1wl$:

Circumscription: Nothing else is a palindrome.

CFG's is a formal mechanism for definitions such as the one for L_{pal}

1. $P \rightarrow \epsilon$
2. $P \rightarrow 0$
3. $P \rightarrow 1$
4. $P \rightarrow 0P0$
5. $P \rightarrow 1P1$

0 and 1 are terminals

P is a variable (or nonterminal, or syntactic category)

P is in this grammar also the start symbol.

1-5 are productions (or rules)

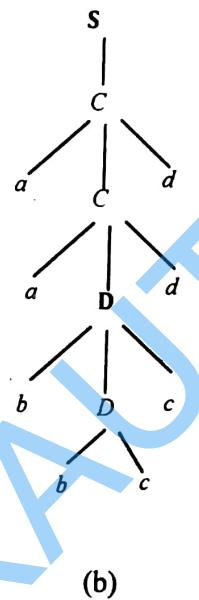
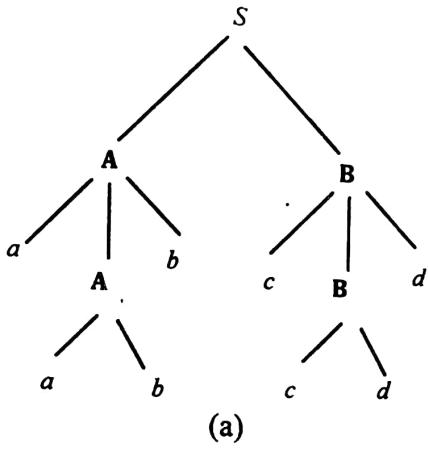
ii) A CFL L is inherently ambiguous if all grammars for L are ambiguous.

Example: Consider, $L = \{a^n b^n c^m d^m : n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n : n \geq 1, m \geq 1\}$.

A grammar for L is

$$\begin{aligned}
 S &\rightarrow AB|C \\
 A &\rightarrow aAb|ab \\
 B &\rightarrow cBd|cd \\
 C &\rightarrow aCd|aDd \\
 D &\rightarrow bDc|bc
 \end{aligned}$$

Let's look at parsing the string aabbccdd.



From this we see that there are two leftmost derivations:

$$\begin{array}{l}
 S \xrightarrow{\text{lm}} AB \xrightarrow{\text{lm}} aAbB \xrightarrow{\text{lm}} aabbB \xrightarrow{\text{lm}} aabbC \xrightarrow{\text{lm}} aabbccdd \\
 S \xrightarrow{\text{lm}} C \xrightarrow{\text{lm}} aCd \xrightarrow{\text{lm}} aaDdd \xrightarrow{\text{lm}} aabDcdd \xrightarrow{\text{lm}} aabbccdd
 \end{array} \quad \text{And}$$

It can be shown that every grammar for L behaves like the one above. The language L is inherently ambiguous.

iii) CFG for the following is given below:

$$L = \{a^n b^m \mid n \neq m\}$$

If $n \neq m$, then two cases are possible

Case - 1

$$n > m$$

$$L_1 = \{a^n b^m : n > m\}$$

If G_1 be the CFG for L_1 language

$$G_1 = (V_n^A, V_t^A, P^A, S^A)$$

$$V_n^A = \{S_1, A, S^A\}$$

$$V_t^A = \{a, b\}$$

Productions P^A are

$$S^A \rightarrow AS_1$$

POPULAR PUBLICATIONS

$$S_1 \rightarrow aS_1b/\epsilon$$

$$A \rightarrow aA/a$$

Case - 2

$$n < m$$

$$L_2 = \{0^n 1^m : n < m\}$$

If G_2 be the CFG for L_2

$$G_B = (V_n^B, V_t^B, P^B, S^B)$$

$$V_n^B = \{S^B, S_2, B\}$$

$$V_t^B = \{a, b\}$$

P^B are :

$$S^B \rightarrow S_2 B$$

$$S_2 \rightarrow aS_2b/\epsilon$$

$$B \rightarrow bB/b$$

∴ By combining $(G_A \& G_B)$ we can write CFG for the above.

$$S \rightarrow S^A / S^B \text{ where } S \text{ is the start symbol.}$$

b) $S \rightarrow aA / B / C / a$

$$A \rightarrow aB / E$$

$$B \rightarrow aA$$

$$C \rightarrow cCD$$

$$D \rightarrow abd$$

$$\text{CNF} \rightarrow S \rightarrow C_1 A / C_2 B / C_3 C / C_4 C_1$$

$$C_1 \rightarrow a$$

$$C_2 \rightarrow \epsilon$$

$$A \rightarrow C_1 B / C_2 E$$

$$B \rightarrow C_1 A$$

$$C \rightarrow C_3 CD$$

$$C_3 \rightarrow c$$

$$C \rightarrow XD$$

$$X \rightarrow C_3 C$$

$$D \rightarrow C_1 C_4 C_5$$

$$C_4 \rightarrow b$$

$$C_5 \rightarrow d$$

$$D \rightarrow Y C_5$$

$$Y \rightarrow C_1 C_4$$

FORMAL LANGUAGE AND AUTOMATA THEORY

c) Definition of non-generating and non-reachable symbols with example.

$$S \rightarrow AB \mid a$$

$$A \rightarrow aA$$

$$B \rightarrow b$$

A variable is non-generating if it does not derive any string of terminals. In the above example A is non-generating symbol.

A variable is non-reachable if the start symbol does not derive a string which contains that variable.

If $X \rightarrow w$, A symbol y is unreachable if w string does not contain y. In the above example B is unreachable after deleting non-generating symbol A from production.

5. Prove that Context Free Languages are not closed under intersection.

[WBUT 2014, 2017]

Answer:

Consider the two languages

$$L_1 = \{ a^n b^n c^m : n \geq 0 \ m \geq 0 \}$$

$$\text{and } L_2 = \{ a^n b^m c^m : n \geq 0 \ m \geq 0 \}.$$

There are several ways one can show that L_1 and L_2 are context-free. For instance, a grammar for L_1 is

$$S \rightarrow S_1 S_2$$

$$S_2 \rightarrow aS_1 b \mid \lambda$$

$$S_2 \rightarrow cS_1 \mid \lambda$$

Alternatively, we note that L_1 is the concatenation of two context – free languages, so it is context-free. But

$$L_1 \cap L_2 = \{ a^n b^n c^n : n \geq 0 \},$$

which is not context-free. Thus, the family of context-free languages is not closed under intersection.

6. a) $E \rightarrow E + E \mid E^* E \mid a$. Prove that the CFG with this production rule is ambiguous. Remove the ambiguity from this grammar.

[WBUT 2018]

Answer:

1st Part:

$$E \rightarrow E + E \mid E^* E \mid a$$

\therefore Say, string is: $a + a^* a$

POPULAR PUBLICATIONS

Derivation Tree 1	Derivation Tree 2

∴ So, more than one derivation tree for the string generated from the grammar.

∴ The given grammar is ambiguous.(Ans.)

2nd Part:

Removing ambiguity:

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow E \mid a \end{aligned}$$

b) $S \rightarrow AB; A \rightarrow a; B \rightarrow C/b, C \rightarrow D; D \rightarrow E, E \rightarrow a$

Remove the unit production.

[WBUT 2018]

Answer:

Refer Question No. 2(b) 1st Part of Long Answer Type Questions.

PUSHDOWN AUTOMATION

Multiple Choice Type Questions

1. Context Free Grammar can be recognized by [WBUT 2010]
 a) finite state automata
 b) 2-way liner bounded automata
 c) push-down automata
 d) both (b) & (c)

Answer: (d)

2. A Push down automation is different from a finite automation because of [WBUT 2011]
 a) a read head
 b) a memory in the form of stack
 c) a set of states
 d) all of these

Answer: (b)

3. PDA is the machine format of [WBUT 2018]
 a) Type 0 Language
 b) Type 1 Language
 c) Type 2 Language
 d) Type 3 Language

Answer: (c)

4. The difference between finite automata and PDA _____. [WBUT 2018]
 a) Reading Head
 b) Input Tape
 c) Finite Control
 d) Stack

Answer: (d)

Short Answer Type Questions

1. Design a PDA which accepts the language.
 $L = \{W \in (a, b)^* \mid W \text{ has equal number of } a \text{ & } b\}$. [WBUT 2006, 2008, 2012, 2018]

OR,

Construct a PDA accepting the set of all strings over $\{a, b\}$ with equal number of a's and b's. [WBUT 2009, 2010]

Answer:

Counting the number of a's and b's, which is easily done with a stack. Here we need not even worry about the order of the a's and b's. We can insert a counter symbol, say 0, into the stack whenever an a is read, then pop one counter symbol from the stack when a b is found. The only difficulty with this is that if there a prefix of w with more b's than a's, we will not find a 0 to use. But this is easy to fix; we can use a negative counter symbol, say 1, for counting the b's that are to be matched against a's later. The complete solution is an nPDA

$M = (\{q_0, q_f\}, \{a, b\}, \{0, 1, z\}, \delta, 0, z, \{q_f\})$ with δ given as

$$\delta(q_0, \lambda, z) = \{(q_f, z)\}, \quad \delta(q_0, a, z) = \{(q_0, 0z)\}, \quad \delta(q_0, b, z) = \{(q_0, 1z)\},$$

POPULAR PUBLICATIONS

$$\delta(q_0, a, 0) = \{(q_0, 00)\}, \quad \delta(q_0, b, 0) = \{(q_0, \lambda)\}, \quad \delta(q_0, a, 1) = \{(q_0, \lambda)\}$$

$$\delta(q_0, b, 1) = \{(q_0, 11)\}$$

In processing the string $baab$, the nPDA makes the moves

$$(q_0, baab, z) \vdash (q_0, aab, 1z) \vdash (q_0, ab, z) \vdash (q_0, b, 0z) \vdash (q_0, \lambda, z) \vdash (q_f, \lambda, z)$$

and hence the string is accepted.

2. Construct an NPDA that accepts the language generated by the productions $S \rightarrow aSa \mid bSb \mid c$. Show an Instantaneous Description of this string $abcba$ for this problem. [WBUT 2007, 2018]

Answer:

$$\begin{aligned} S &\Rightarrow aSa \\ &\Rightarrow abSba \\ &\Rightarrow abcba \end{aligned}$$

3. What are the nonempty transitions in an NPDA?

[WBUT 2009, 2010]

Answer:

Each move an NPDA reads a symbol from the input, changes the contents of the stack (that is, pop a symbol off and/or push several symbols into the stack), and changes the internal state of the npda.

Definition: An npda is $M = \text{def}(Q; \Sigma; \Gamma; \delta; q_0; z; F)$, where

Q is a (finite, nonempty) set of states,

Σ is the (finite, nonempty) input vocabulary,

Γ is the (finite, nonempty) stack alphabet,

$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow \text{finite subsets of } Q \times \Gamma$

\square

is the state transition function,

$q_0 \in Q$ is the initial state,

$z \in \Gamma$ is the initial stack symbol, and

$F \subseteq Q$ is the (finite, nonempty) set of accepting states.

An npda may include λ -moves of the form $\delta(q; \lambda; x)$.

Note that every moves pops off exactly one symbol from the stack.

4. Construct a Push Down Automata equivalent o the following Context Free Grammar $G = (V_n, V_t, P, S)$ with $V_n = \{S, E\}$, $V_t = \{0, 1\}$ and P is define transition follows:

$$\begin{aligned} S &\rightarrow 0S1/A \\ A &\rightarrow 1A0/S/\epsilon \end{aligned}$$

Now check the string 001011 is acceptable by this Push Down Automata or not. [WBUT 2015]

Answer:

1st Part:

The PDA is defined as follows:

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

δ is defined by the following rules:-

$$R1: \delta(q, \Lambda, S) = \{(q, 0S1)\}$$

$$R2: \delta(q, \Lambda, A) = \{(q, 1S0), (q, S)\}$$

$$R3: \delta(q, 0, 0) = \{(q, \Lambda)\}$$

$$R4: \delta(q, 1, 1) = \{(q, \Lambda)\}$$

2nd Part:

$(q, 001011, z) = (q, 01011, 0z) = (q, 1011, z) = (q, 011, 1z) \Rightarrow$ no rule is specified for the last transition. Hence the given string is not accepted by the PDA.

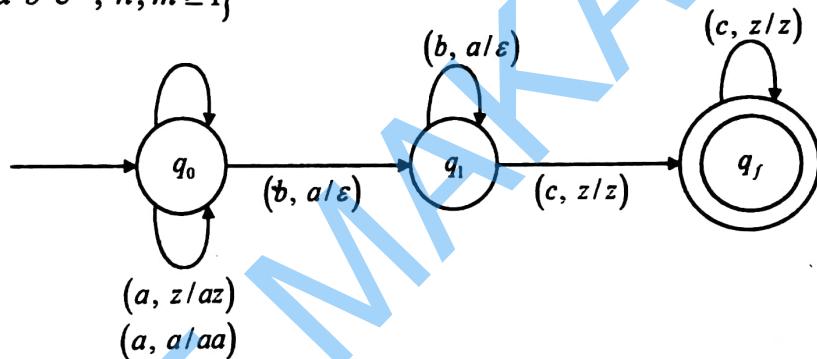
5. Construct a PDA to accept the language $L = \{a^n b^n c^m; n, m \geq 1\}$ by empty stack and by final state.

[WBUT 2018]

Answer:

Construct a PDA to accept the Language

$$L = \{a^n b^n c^m; n, m \geq 1\} \text{ ---}$$



Transition Functions: (Stack)

$$\delta(q_0, a, z) = (q_0, az) \quad \left[\text{push 'a' from the beginning} \right]$$

$$\delta(q_0, a, a) = (q_0, aa) \quad \left[\text{push 'a' from the beginning} \right]$$

$$\delta(q_0, b, a) = (q_1, \epsilon) \quad \left[\text{pop whenever 'b' comes} \right]$$

$$\delta(q_1, b, a) = (q_1, \epsilon) \quad \left[\text{pop whenever 'b' comes} \right]$$

$$\delta(q_1, c, z) = (q_f, z) \quad \left[\text{do nothing for 'c', accepted by the empty stack} \right]$$

$$\delta(q_f, c, z) = (q_f, z) \quad \left[\text{final state} \right]$$

\therefore Here, no. of a 's are equal to

No. of b 's \Rightarrow so, push for ' a ' and pop for ' b ' \Rightarrow approach is used.

Long Answer Type Questions

1. Define Pushdown Automata.

[WBUT 2006, 2009, 2010, 2011, 2012, 2018]

Answer:

A **pushdown automaton** is a finite automaton that can make use of a stack containing data.

Pushdown automata differ from normal finite state machines in two ways:

1. They can use the top of the stack to decide which transition to take.
2. They can manipulate the stack as part of performing a transition.

Pushdown automata choose a transition by indexing a table by input signal, current state, and the top of the stack. Normal finite state machines just look by input signal and current state: they have no stack to work with. Pushdown automata add the stack as a parameter for choice. Given an input signal, current state, and a given symbol at the top of the stack, a transition path is chosen.

Pushdown automata can also manipulate the stack, as part of performing a transition. Normal finite state machines choose a new state, the result of following the transition. The manipulation can be to push a particular symbol to the top of the stack, or to pop off the top of the stack. The automaton can alternatively ignore the stack, and leave it as it is. The choice of manipulation (or no manipulation) is determined by the transition table.

2. Construct PDA for $L = \{ww^R : w \text{ belongs to } (0,1)^*\}$.

[WBUT 2011, 2012]

OR,

Construct a PDA to accept $L = \{WW^R \mid W \text{ belongs to } (a, b)^* \text{ and } W^R \text{ is reverse string of } W\}$ by empty stack and final state.

[WBUT 2013]

Answer:

Hear the main concept is that the symbols are retrieved from a stack in the reverse order of their insertion. When reading the first part of the string we push consecutive symbols on the stack. For the second part, we compare the current input symbol with the top of the stack continuing as long as the two match. Since symbols are retrieved from the stack in reverse of the order in which they were inserted, a complete match will be achieved if and only if the input is of the form ww^R .

An apparent difficulty with this suggestion is that we do not know the middle of the string, that is, where w ends and w^R starts. But the nondeterministic nature of the automaton helps us with this; the nPDA correctly guesses where the middle is and switches states at that point. A solution to the problem is given by

$$\begin{aligned} M &= \{Q, \Sigma, \Gamma, \delta, q_0, z, F\}, \\ Q &= \{q_0, q_1, q_2\}, \\ \Sigma &= \{a, b\}, \\ \Gamma &= \{a, b, z\}, \\ F &= \{q_2\} \end{aligned}$$

The transition function can be visualized as having several parts: a set to push w on the stack.

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

$$\begin{aligned}\delta(q_0, b, a) &= \{(q_0, ba)\}, \\ \delta(q_0, a, b) &= \{(q_0, ab)\}, \\ \delta(q_0, b, b) &= \{(q_0, bb)\}, \\ \delta(q_0, a, z) &= \{(q_0, az)\}, \\ \delta(q_0, b, z) &= \{(q_0, bz)\},\end{aligned}$$

a set to guess the middle of the string, where the nPDA switches from state q_0 to q_1 .

$$\begin{aligned}\delta(q_0, \lambda, a) &= \{(q_1, a)\}, \\ \delta(q_0, \lambda, b) &= \{(q_1, b)\},\end{aligned}$$

a set to match w^R against the contents of the stack,

$$\begin{aligned}\delta(q_1, a, a) &= \{(q_1, \lambda)\}, \\ \delta(q_1, b, b) &= \{(q_1, \lambda)\},\end{aligned}$$

and finally

$$\delta(q_1, \lambda, z) = \{(q_2, z)\},$$

to recognize a successful match.

The sequence of moves in accepting $abba$ is

$$(q_0, abba, z) \vdash (q_0, baa, az) \vdash (q_0, ba, baz) \vdash (q_1, ba, baz) \vdash (q_1, a, az) \vdash (q_1, \lambda, z) \vdash (q_2, z)$$

The nondeterministic alternative for locating the middle of the string is taken at the third move. At that stage, the PDA has the instantaneous descriptions (q_0, ba, baz) and has two choices for its next move. One is to use $\delta(q_0, b, b) = \{(q_0, bb)\}$, and make the move

$$(q_0, baa, az) \vdash (q_0, a, bbaz)$$

The second is the one used above, namely $\delta(q_0, \lambda, b) = \{(q_1, b)\}$, only the latter leads to acceptance of the input.

3. Construct an equivalent PDA for the following CFG.

[WBUT 2013, 2016]

$$S \rightarrow aAB/bBA$$

$$A \rightarrow bS/a$$

$$B \rightarrow aS/b$$

Show an ID for the string **abbaaabbbab** for the PDA generated with stack description.

Answer:

Ith Part: Constructing the PDA

We note that the given grammar is already in Greibach Normal Form. Hence, our PDA has only one state --- q (i.e., $Q = \{q\}$), which is also the start state and the final state ((i.e., $F = \Phi$)).

The input alphabet is clearly $\Sigma = \{a, b\}$ and the stack alphabet consists of only the non-terminals, i.e. $\Gamma = \{S, A, B\}$, with S as the start symbol.

The required PDA is $\langle Q, \Sigma, \Gamma, \delta, q, S, \Phi \rangle$, where δ is:

$$\begin{aligned}(q, a, S) &= \{(q, AB)\} \\ (q, b, S) &= \{(q, BA)\}\end{aligned}$$

POPULAR PUBLICATIONS

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

$$(q, b, A) = \{(q, S)\}$$

$$(q, a, B) = \{(q, S)\}$$

$$(q, b, B) = \{(q, \epsilon)\}$$

2nd Part:

Instantaneous Descriptions (IDs) form string abbaaabbbbab.

$$(q, abbaaabbbbab, S)$$

- (q, bbaaabbbbab, AB)
- (q, baaabbbbab, SB)
- (q, aaabbbbab, BAB)
- (q, aabbbbab, SAB)
- (q, abbbbab, ABAB)
- (q, bbbbab, BAB)
- (q, bbab, AB)
- (q, bab, SB)
- (q, ab, BAB)
- (q, b, SAB)
- (q, ϵ , BAAB)

Since the ID is NOT (q, ϵ, ϵ) and there is no input to scan, the PDA rejects the string.

Note: The above ID corresponds to the below mentioned left-most derivation in the given grammar:

S è aAB è abSB è abbBAB è abbaSAB è abbaaABAB è abbaaaBAB è
abbaaabAB è abbaaabbSB è abbaaabbbBAB è abbaaabbaSAB è
abbaaabbbbabBAAB (No further derivation possible).

4. Define PDA by giving a block diagram.

[WBUT 2014, 2017]

What is the difference between DPDA and NPDA?

[WBUT 2014]

Design a Non Deterministic Pushdown Automata for accepting the string $L = \{\text{Set of all palindromes over } a, b\}$ by Empty stack

[WBUT 2014]

OR,

Design a non-Deterministic Pushdown Automata for accepting the string $L = \{WCW^R \mid W \in \{a, b\}^*\}$ and W^R is the reverse of W by Empty stack. [WBUT 2017]

Construct an equivalent PDA for the following Context Free Grammar.

$$S \rightarrow aA$$

$$A \rightarrow aABC/bB/a$$

$$C \rightarrow c$$

Show an ID for the string aabbabc for the PDA generated.

[WBUT 2014, 2017]

Answer:

1st Part:

A pushdown automata (PDA) is a seven-tuple: $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$,

where

- Q is a finite set of states
- Σ is a finite input alphabet,

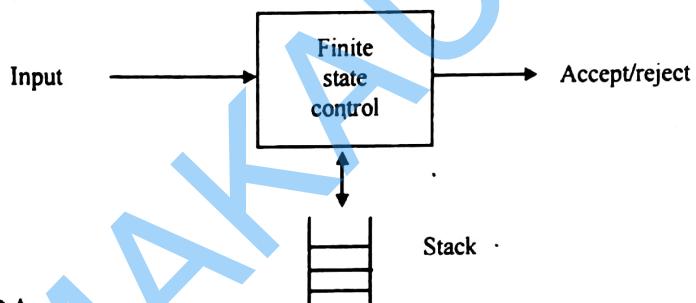
FORMAL LANGUAGE AND AUTOMATA THEORY

- Γ is a finite stack alphabet ,
- $\delta: Q \times \Sigma \cup \{\epsilon\} \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ is the transition function,
- q_0 is the start state,
- $Z_0 \in \Gamma$ is the start symbol for the stack, and
- $F \subseteq Q$ is the set of accepting states.

A PDA is essentially an ϵ -NFA with a stack.

On a transition the PDA

1. Consumes an input symbol.
2. Goes to a new state (or stays in the old).
3. Replaces the top of the stack by any string (does nothing, pops the stack, or pushes a string onto the stack).



2nd Part:

Difference between DPDA and NPDA

1. The transition function δ is atmost single-valued for DPDA, multi-valued for an NPDA. Formally: $|\delta(q_i, a, b)| = 0$ or 1 , for every $q_i \in Q$, $a \in \Sigma \cup \{\epsilon\}$ and $b \in F$.
2. Both NPDA and DPDA may have ϵ -transitions; but a DPDA may have a ϵ -transition only if no other transition is possible.

3rd Part:

Null string may belong to the Language set. The PDA will be designed in such a way that null string can be accepted by the PDA. The transitional functions will be

$$\begin{aligned}
 &\delta(q_0, a, z_0) \rightarrow (q_0, z_1 z_0), \\
 &\delta(q_0, b, z_0) \rightarrow (q_0, z_2 z_0), \\
 &\delta(q_0, a, z_1) \rightarrow (q_0, z_1 z_1), (q_1, \lambda), \\
 &\delta(q_0, b, z_1) \rightarrow (q_0, z_1 z_1), \\
 &\delta(q_0, a, z_2) \rightarrow (q_0, z_1 z_2), \\
 &\delta(q_0, b, z_2) \rightarrow (q_0, z_2 z_2), (q_1, \lambda), \\
 &\delta(q_1, a, z_1) \rightarrow (q_1, \lambda), \\
 &\delta(q_1, b, z_2) \rightarrow (q_1, \lambda), \\
 &\delta(q_0, \lambda, z_0) \rightarrow (q_1, \lambda) \quad // \text{this function is for accepting null string}, \\
 &\delta(q_1, \lambda, z_0) \rightarrow (q_1, \lambda) \quad \text{accepted by empty stack}, \\
 &\delta(q_1, \lambda, z_0) \rightarrow (q_1, z_0) \quad \text{accepted by final state}.
 \end{aligned}$$

POPULAR PUBLICATIONS

4th Part: Question is wrong.

5. Explain how a string be accepted by a PDA?

[WBUT 2017]

Answer:

There are two different ways to define PDA acceptability.

Final State Acceptability

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

For a PDA $(Q, \Sigma, S, \delta, q_0, I, F)$, the language accepted by the set of final states F is –
 $L(PDA) = \{w \mid (q_0, w, I) \xrightarrow{*} (q, \epsilon, x), q \in F\}$
for any input stack string x .

Empty Stack Acceptability

Here a PDA accepts a string when, after reading the entire string, the PDA has emptied its stack.

For a PDA $(Q, \Sigma, S, \delta, q_0, I, F)$, the language accepted by the empty stack is –
 $L(PDA) = \{w \mid (q_0, w, I) \xrightarrow{*} (q, \epsilon, \epsilon), q \in Q\}$

CLOSURE PROPERTIES OF CFLS

Multiple Choice Type Questions

1. The intersection of CFL and regular language [WBUT 2006, 2012]

- a) need not be regular
- b) need not be CF
- c) is always regular
- d) none of these

Answer: (d)

$L \cap R$ is a CFL. So it is correct, but (iv) should be the answer.

2. The class of context free language is not closed under [WBUT 2008, 2009, 2013]

- a) concatenation
- b) union
- c) intersection
- d) repeated concatenation

Answer: (c)

3. The grammar $G = (\{S\}, \{0,1\}, P, S)$ where $P = (S \rightarrow 0S1, S \rightarrow 0S, S \rightarrow S1, S \rightarrow 0)$ is a [WBUT 2008]

- a) recursively enumerable language
- b) regular language
- c) context sensitive language
- d) context free language

Answer: (d)

4. Which of the following statements is wrong? [WBUT 2010]

- a) A turning machine cannot solve halting problem
- b) Set of recursively enumerable languages is closed under union
- c) A finite state machine with 3 stacks is more powerful than infinite state machine with 2 stacks
- d) Context sensitive grammar can be recognized by a linearly bounded memory machine

Answer: (c)

5. The intersection of CFL & RE is always [WBUT 2014, 2015]

- a) CFL
- b) RE

- c) CSL

- d) CFL OR CSL

Answer: (a)

[WBUT 2018]

6. Useless symbols in CFG are

- a) Non-generating symbol and non-reachable symbols
- b) Null alphabets and null string
- c) Non-terminal symbols
- d) All of the above

Answer: (b)

Short Answer Type Questions

Prove that CFLs are not closed under intersection and complement operation. [WBUT 2008, 2010, 2016]

Prove that the intersection of a context-free language and a regular language is a context-free language. [WBUT 2008, 2010]

OR,

[WBUT 2013, 2016]

Explain Ogden's Lemma for CFL.

Answer:

Ogden's Lemma

In the theory of formal languages, **Ogden's lemma** provides an extension of flexibility over the pumping lemma for context-free languages.

Ogden's lemma states that if a language L is context-free, then there exists some number $p > 0$ (where p may or may not be a pumping length) such that for any string w in L , we can mark p or more of the positions in w as "distinguished"; then w can be written as

$$w = uvxyz$$

with strings u , v , x , y and z , such that v and y have at least one distinguished position between them, vxy has at most p distinguished positions, and

$$uv^i xy^i z \text{ is in } L \text{ for every } i \geq 0.$$

Note that this is trivially true if the language is not infinite, since then p just needs to be longer than longest string in the language.

Ogden's lemma can be used to show that certain languages are not context-free, in cases where the pumping lemma for context-free languages is not sufficient. Observe that when every position is marked as distinguished, this lemma is equivalent to the pumping lemma for context-free languages.

Next Part:

Let L_1 be a context-free language and L_2 be a regular language. Then we have to prove that $L_1 \cap L_2$ is context-free.

Proof:

Let $M_1 = (Q, \Sigma, \Gamma, \delta_1, q_0, z, F_1)$ be an npda which accepts L_1 and $M_2 = (P, \Sigma, \delta_2, p_0, F_2)$ be a dfa that accepts L_2 . We construct a pushdown automation $\hat{M} = (\hat{Q}, \Sigma, \Gamma, \hat{\delta}, \hat{q}_0, \hat{z}, \hat{F})$ which simulates the parallel action of M_1 and M_2 ; whenever a symbol is read from the input string, \hat{M} simultaneously executes the moves of M_1 and M_2 . To this end we let

$$\hat{Q} = Q \times P \quad q_0 = (q_0, p_0), \quad \hat{F} = F_1 \times F_2,$$

and define $\hat{\delta}$ such that $((q_k, p_l), x) \in \hat{\delta}((q_i, p_j), a, b)$,

$$\text{iff} \quad (q_k, x) \in \delta_1(q_i, a, b), \quad \text{and} \quad \delta_2(p_j, a) = p_l$$

FORMAL LANGUAGE AND AUTOMATA THEORY

In this, we also require that $\alpha = \lambda$, then $p_j = p_i$. In the other words, the states of \hat{M} are labeled with pairs (q_i, p_j) , representing the respective states in which M_1 and M_2 can be after reading a certain input string. It is a straightforward induction argument to show that $((q_0, p_0), w, z) \vdash^*_{\hat{M}} ((q_r, p_s), x)$, with $q_r \in F_1$ and $p_s \in F_2$ if and only if $(q_0, w, z) \vdash^*_{M_1} (q_r, p_s)$, and $\delta^*(p_0, w) = p_s$.

Therefore, a string is accepted by \hat{M} if and only if it is accepted by M_1 and M_2 , that is, if it is in $L(M_1) \cap L(M_2) = L_1 \cap L_2$.

TURING MACHINE

Multiple Choice Type Questions

1. A shift register is

- a) Mealy M/C
- b) Turing M/C

- c) Moore M/C

[WBUT 2013]

d) All of these

Answer: (d)

2. Difference between Turing Machine & Two way FA is in [WBUT 2014, 2015, 2017]

- a) Input Tape
- b) Read write head

- c) Finite Control

d) All of these

Answer: (d)

3. Which of the following statements is false?

[WBUT 2014, 2015, 2017]

- a) The halting problem of Turing machine is undecidable

- b) Determining whether a context free grammar is ambiguous is undecidable

- c) Given two arbitrary context free grammars G_1 and G_2 . It is undecidable whether $L(G_1) = L(G_2)$

- d) Given two regular grammars G_1 and G_2 . It is undecidable whether $L(G_1) = L(G_2)$

Answer: (d)

4. Which is not a part of the mechanical diagram of 'Turing Machine'? [WBUT 2018]

- a) Input Tape
- b) Read-write head
- c) Finite control

- d) Stack

Answer: (d)

5. Which of the string is accepted by Turing Machine?

[WBUT 2018]

- a) $L = a^n c^m b^n$, where $m, n > 0$

- b) $L = a^n b^n c^i$, $n, i > 0$

- c) $L = a^n b^n c^n$, where $n > 0$

- d) All of these

Answer: (d)

Short Answer Type Questions

1. What do you mean by Halting problem of a Turing machine? [WBUT 2008, 2013]

OR,

Why a Turing machine is called linear bounded Automata?

[WBUT 2008]

OR,

What is halting problem in Turing machine?

[WBUT 2015, 2016]

Answer:

1st Part:

A Turing Machine is a mathematical object to define "Computing". It has a "doubly-infinite" tape for input and a state machine that governs its running. A Tuning Machine can behave like an algorithm, especially for problems with "yes/no" answer. In such a case, the machine starts with the description of a problem on the input tape and comes to

FORMAL LANGUAGE AND AUTOMATA THEORY

a “Yes” halt or a “no” halt depending upon the result. However, it may happen that the machine never comes to any of he halts and go on running.

A related concept of “Universal Turing Machine” is a Turing machine that can take two things as input –

- a coded description of the state machine of another Turing machine ‘M’ and
- the input given to M. The Universal Turing Machine then proceeds to ‘simulate’ M.

The halting problem of Turing Machine is the question whether there can be a Universal Turing Machine that is given to simulate ANY machine ‘M’ with ANY input ‘ ω ’, with answer Yes / No depending upon whether M halts (i.e. M does not run forever) or does not halt, respectively, on input ‘ ω ’.

Turing proved that the halting problem is unsolvable i.e. there can be no such machine as described.

2nd Part:

A linear bounded automata is an non deterministic turing machine satisfying the following two conditions.

1. Its input alphabet includes two special symbols ϵ and $\$$, the left and right end markers respectively.
2. The LBA has no moves left from or right from $\$$, nor may it print another symbol over ϵ or $\$$.

The linear bounded automaton is simply a turning machine which instead of having potentially infinite tape on which to compute, is restricted to the portion of the tape containing the input x plus the two tape squares holding the end -markers.

An LBA will be denoted as

$M = (Q, \Sigma, \Gamma, \delta, q_0, \epsilon, \$, F)$ where $Q, \Sigma, \Gamma, \delta, q_0$ and F are as for a non-deterministic TM; ϵ and $\$$ are symbols in Σ , the left and right end-markers. $L(M)$, the language accepted by M , is $\{w | w \text{ is in } (\Sigma - \{\epsilon, \$\})^* \text{ and } q_0 \xrightarrow[M]{\epsilon} w \xrightarrow[M]{\$} qf \text{ for some } q \in F\}$

2. Construct a Turing machine that accepts all strings over {0, 1} with an even number 0's and even number of 1's. [WBUT 2011]

Answer:

The required Turing Machine is

$$M = \langle \{q_0, q_1, q_2, q_3, q_{SUCC}, q_{FAIL}\}, \Gamma, \Delta, \{0, 1\}, \delta, q_0, \{q_{SUCC}, q_{FAIL}\} \rangle$$

where δ is as given below:

$$\delta(q_0, 0) = (q_1, 0, R)$$

$$\delta(q_0, 1) = (q_3, 0, R)$$

$$\delta(q_0, \Delta) = (q_{SUCC}, \Delta, L)$$

$$\delta(q_1, 0) = (q_0, 0, R)$$

POPULAR PUBLICATIONS

$$\delta(q_1, 1) = (q_2, 0, R)$$

$$\delta(q_1, \Delta) = (qFAIL, \Delta, L)$$

$$\delta(q_2, 0) = (q_3, 0, R)$$

$$\delta(q_2, 1) = (q_2, 0, R)$$

$$\delta(q_2, \Delta) = (qFAIL, \Delta, L)$$

$$\delta(q_3, 0) = (q_2, 0, R)$$

$$\delta(q_3, 1) = (q_0, 0, R)$$

$$\delta(q_3, \Delta) = (qFAIL, \Delta, R)$$

3. Design a Turing Machine that recognizes the language of all string of even length over the alphabet {a, b}. [WBUT 2013]

Answer:

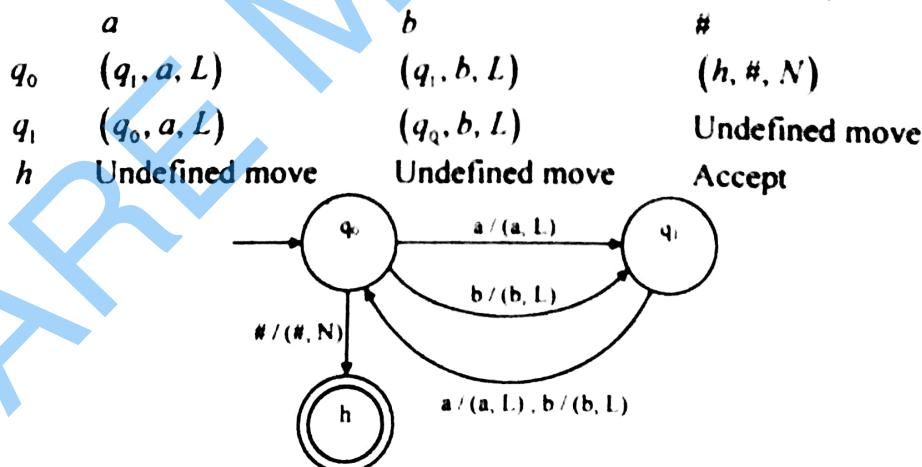
Let Turing Machine $(T) = (Q, \Sigma, \Gamma, \delta, q_0, h)$

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

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

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

Here is h half state when machine halts after accepting the language. q_0 is initial state



Long Answer Type Questions

1. a) Design a TM that accepts $\{0^n 1^n | n \geq 1\}$

[WBUT 2013, 2016]

b) Design a TM which can multiply two positive integers.

[WBUT 2013]

Answer:

a) Let us now follow the working of TM M_1 with input 000111.

FORMAL LANGUAGE AND AUTOMATA THEORY

The initial configuration is **bS000111b**. This means that the TM is in state S and the head is on the first 0 of the input string. This corresponds to M_1 being as given in Fig - 1. The *current state* is mentioned in the box representing the finite control of the machine.

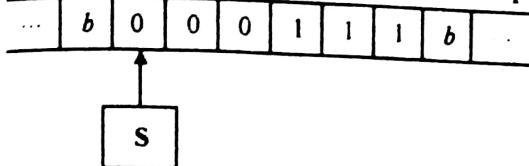


Fig: 1 Machine M_1 in Initial Configuration

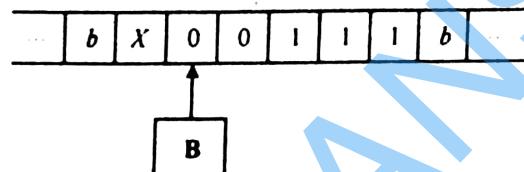


Fig: 2 M_1 After First Move

Since the input is 0, consulting the state table, we find that the move is (B, X, R) , i.e., the next state is B , the 0 under the tape is overwritten by an X and the head moves *right* to place itself on the next (i.e., second) 0 of the input. This gives us the next configuration as

bXB00111b, as shown pictorially in Fig - 2.

Following up with the next moves, we find that the subsequent configurations are:

bX0B0111b followed by **bX00B111b**. The state of the machine in the last of the above two configurations is as seen in Fig - 3.

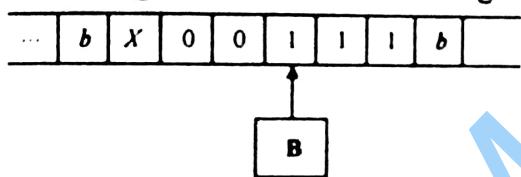


Fig: 3 M_1 When on First 1

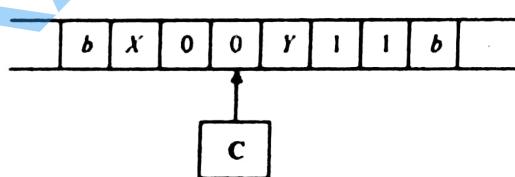


Fig: 4 M_1 After Marking First 1 to Y

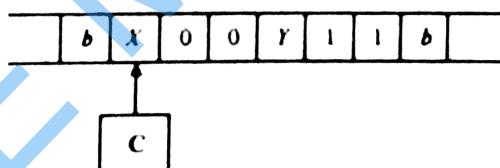


Fig: 5 M_1 After Matching One Pair of 0 and 1

At this point, we find the applicable move is (C, Y, L) which means that the next configuration is **bX0C0Y11b**. Observe in Fig - 4 that the machine has moved left after marking the first 1 to Y.

At this point, the next applicable move, i.e., $(C, 0, L)$, will effectively *skip* over the 0-s till the head is positioned on the first (up to now, the only) X encountered. The configurations leading to that

are **bXC00Y11b** followed by **bCX00Y11b**. The machine position now is as given in Fig - 5.

The next move is (S, X, R) and hence the next configuration is **bXS00Y11b**.

Let us summarize what the machine has done up to this point:

1. It has marked the first 0 of the input string to X
2. Then it has skipped the following 0-s while moving *right* till it encountered the first 1 which was marked to Y and the machine moved *left*

POPULAR PUBLICATIONS

3. Now the machine skipped all 0-s while moving left till it got an X upon which it moved right and reached a condition similar to what was at the beginning of the run as shown in Fig - 6.

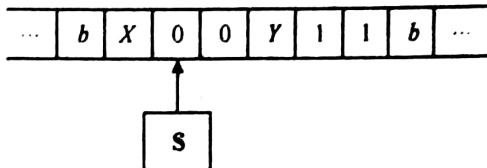


Fig: 6 M_1 Back in State S

Let us now augment the above summarization with the emphasized phrases:

1. It has marked the first 0 of the input string to X. However, if there is no 0 to mark but there is a Y below the head then it moves right over the Y-s till it reaches the end of input whence it halts with SUCCESS
2. Then it has skipped the following 0-s or Y-s while moving right till it encountered the _rst 1 which was marked to Y and the machine moved left
3. Now the machine skipped all 0-s while moving left till it got an X upon which it moved right and reached a condition similar to what was at the beginning of the run. *However, if there is no 0 to skip, it halts with SUCCESS*

If you properly visualize the moves, you will see that our Turing Machine M_1 :

- Checks out one 0 to X and one 1 to Y in each zigzag movement over the input.
- Declares SUCCESS if and only if the number of 0-s marked as X is exactly the same as the number of 1-s marked as Y

Hence our TM recognizes only inputs that are of the form $0^n 1^n$, $n \geq 1$.

b) Design a TM, which can multiply two positive integers

We can use subroutines for design of TM which can multiply two positive integers

$0^m 1 0^n$ [$m \times n$: operators are represented with 1] [If $m = 3, n = 2$, $m \times n = 6$]

So at the final state the TM will halt with 6 zeros and all surrounded by Blank B.

i) If First 0 is encountered by q_0 it is changed to Blank B

B 0 0 1 0 0 1 B B B B

ii) Move right until first 1 is encountered, change the immediate next 0 to X

B 0 0 1 X 0 1 B B B B

iii) Move right and skip all symbol until first blank is encountered, then change the blank to 0 (as if 0 is copied here) – it is called subroutine call

B 0 0 1 X 0 1 0 B B B

iv) Move left until X is encountered, change the next 0 to X

B 0 0 1 X X 1 0 B B B

v) Move right until B is encountered and change B to 0

B 0 0 1 X X 1 0 0 B B

- vi) Move left until X reached and change consecutive X to 0's
B 0 0 1 0 0 1 0 0 B B
- vii) Move to left until B is reached, change the next 0 to B
B B 0 1 0 0 1 0 0 B B
- viii) Move right until 1 reached change the next 0 to X, (subroutine)
B B 0 1 X 0 1 0 0 B B
- ix) Move right until B reached, change that B to 0
B B 0 1 X 0 1 0 0 0 B B
- x) Move left until X reached, change next 0 to X and repeat the procedure as above.
B B 0 1 X X 1 0 0 0 B B
B B 0 1 X X 1 0 0 0 0 B B
B B 0 1 0 0 1 0 0 0 0 B B
B B B 1 0 0 1 0 0 0 0 B B B
B B B 1 X 0 1 0 0 0 0 B B
B B B 1 X 0 1 0 0 0 0 B
B B B 1 X X 1 0 0 0 0 0 B B B
B B B 1 X X 1 0 0 0 0 0 0 B B B
B B B 1 0 0 1 0 0 0 0 0 B B B B
B B B 1 0 0 1 0 0 0 0 0 B

So, now no more 0's encountered, then change all the 1 & 0's on left to B.

Halt $\rightarrow 0/B \quad [B \text{ } B | 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 | B \text{ } \dots]$.

2. Design a Turing Machine which performs addition of two integers. Write short note on Multi Tape and Multi Head Turing Machine. Prove that the problem "A string w halts on a Turing Machine M" is undecidable. [WBUT 2014]

Answer:

1st Part:

The turning machine will have 0"10" on its tape initially. Therefore it will start with the leftmost 0, so on scanning 0's moving right keeping them as it is, till if get 1, if replace this 1 by 0 and then again so on. Scanning 0's and moving right keeping them as it is. When it gets a B (blank), it moves left and replaces the right most 0 by B and halves thereby leaving finally $m+n$ 0's on the tape. Therefore, the moves of the Turing machine are

	0	1	B
q_0	$(d_0, 0, R)$	$(d_1, 0, R)$	—
q_1	$(d_1, 0, R)$	—	q_2, B, L
q_2	q_3, B, R	—	—
q_3	—	—	—

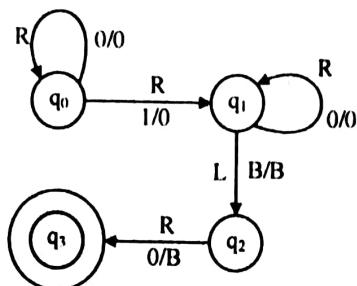
POPULAR PUBLICATIONS

Therefore the Turing machine

$$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_1\})$$

where δ is given above.

The transition diagram corresponding to the above table is



2nd Part:

A Multi-tape Turing machine is like an ordinary Turing machine with several tapes. Each tape has its own head for reading and writing. Initially the input appears on tape 1, and the others start out blank.

A k-tape Turing machine can be described as a 6-

$$M = (Q, \Gamma, s, b, F, \delta)$$

where

Q is a finite set of states

Γ is a finite set of the tape alphabet

$s \in Q$ is the initial state

$b \in \Gamma$ is the blank symbol

$F \subseteq Q$ is the set of final or accepting states

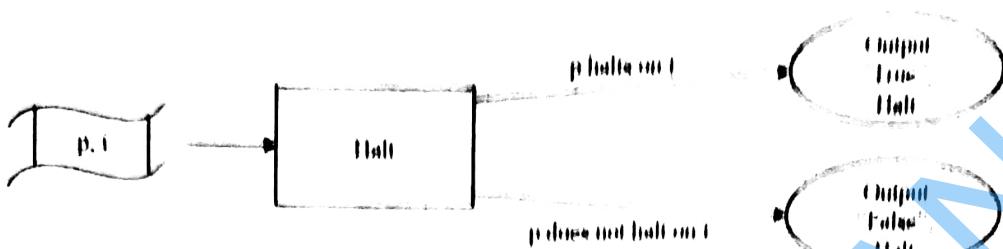
$\delta: Q \times \Gamma^k \rightarrow (\Gamma \times \{L, R, S\})^k$ is a partial function called the transition function, where k is the number of tapes, L is left shift, R is right shift and S is no shift.

A multihead Turing machine can be visualized as a Turing machine with a single tape and a single control unit but with multitape.

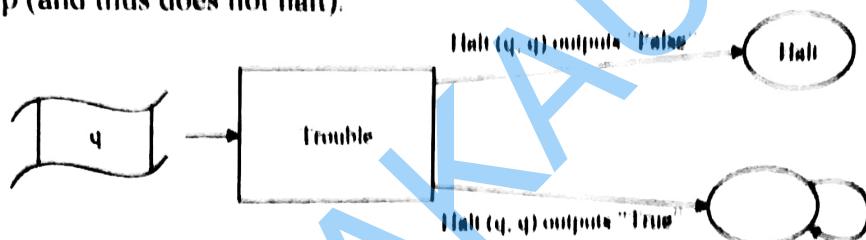
3rd Part:

Here is a sketch of how you can prove that the Halting problem is undecidable.

Assume that a TM capable of solving the halting problem exists. Call this TM halt. It takes, as input, the encoded representation of a TM and an input tape on which the encoded TM will run. Let's call this input pair (p, i) : p for "program" (the encoded TM) and i for "input" (the input tape). If input TM p will halt on input i , then halt halts and outputs "true". If input TM p will not halt on i , then halt halts and outputs "false". So, we can view halt as a function that takes parameters p and i and produces the output "true" or "false" depending on whether or not TM p halts on input i .



Based on halt, we can trivially construct a new TM, which we can call trouble, which will work as follows. It first duplicates the entire input tape q , creating two copies of the original input tape. It then runs halt, using one copy of the original input q as parameter p and the other copy of the original input q as parameter t . If $\text{halt}(q, q)$ outputs "false", then trouble TM halts. If $\text{halt}(q, q)$ outputs "true", then trouble goes into an infinite loop (and thus does not halt).



Any TM can be encoded as an initial tape. So, let's assume that the encoding of trouble as a tape is called t .

Consider what will happen when trouble is executed with t as its input tape. Does trouble halt?

If trouble halts, that means that $\text{halt}(t, t)$ answered "false". However, that means that trouble does not halt when given t as input.

If trouble does not halt, that means that $\text{halt}(t, t)$ answered "true". However, that means that trouble does halt when given t as input.

In either case, halt gave the wrong answer. Therefore, given any TM that claims to solve the halting problem, it is possible to construct a program/input pair for which it will answer incorrectly. So, any claim that a particular TM solves the Halting problem can be proved false, meaning that the Halting Problem is undecidable.

[WBUT 2016]

- a) Define Turing machine.
- b) Explain different types of Turing machine.
- c) Design a Turing machine that accepts the language of all strings which contain "aba" as a substring.

Answer:

- a) A Turing machine is an abstract concept used to describe a type of machine that, given an indefinite amount of space and time, can be adapted to calculate anything, such as the digits of π or even a whole universe.

POPULAR PUBLICATIONS

b) There are many different types of Turing machines that are often used to describe certain kinds of execution. The two that are most often used are deterministic and non-deterministic Turing machines.

Automata are often used to represent Turing machines. This is a deterministic Turing machine which calculates the two's complement of some binary input.

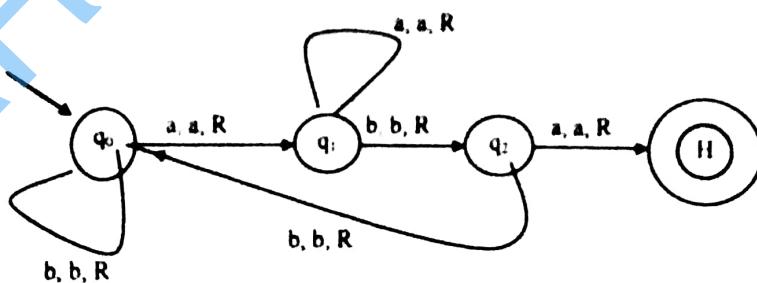
A deterministic Turing machine is one that uses the concept of determinism to calculate a solution to a problem.

Determinism is the concept of being only in one state at a given time and determining what the next state will be from that state. In simpler terms, determinism would be being in state q_0 , and only holding that state until moving onto the next state, q_1 . In determinism we would be able to predict without any doubt that the head would move from state q_0 to state q_1 . A Turing machine does not have to even halt, or stop execution, in order for it to be considered deterministic.

A non-deterministic Turing machine is one that uses the concept of non-determinism to calculate a solution to a problem.

A non-deterministic Turing Machine differs from a deterministic Turing Machine in the sense that a non-deterministic Turing Machine can have several possible states to which it can transition from any given state, q_i . One way to think of it would be to think that, given the possibility of choosing from several subsequent states, the non-deterministic Turing machine guesses the next iteration that will bring it to a 'yes' answer. Put a different way, the non-deterministic Turing machine branches out into holding many states at the same time in a sort of tree fashion until one of the many paths leads it to a 'yes' answer. In perspective a non-deterministic Turing machine may, for instance, be in state q_0 and then hold both state q_{11} as one of the branches and state q_{12} as another branch.

c) Strings in which aba is present as a substring.



4. Design a Turing Machine which accepts the language $L = \{a^n b^n, n \geq 1\}$. Write a short note on Multi-Tape and Multi Head Turing Machine.

[WBUT 2017]

Answer:

1st Part:

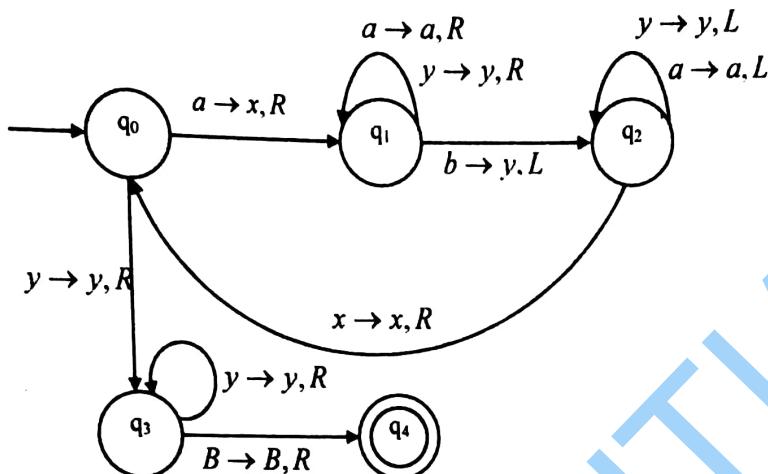


Fig: 1

Let us now follow the working of TM M_1 with input **aabb**.

The initial configuration is **BaabbBB**. This means that the TM is in state q_0 and the head is on the first **a** of the input string. This corresponds to M_1 being as given in Fig: 2(a). The *current state* is mentioned in the box representing the finite control of the machine.

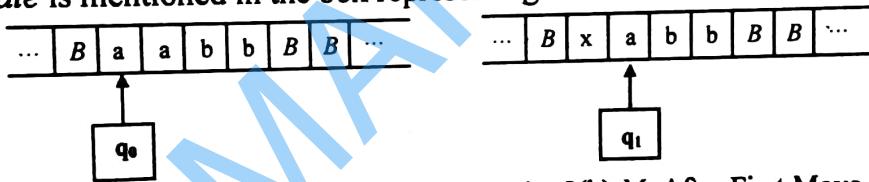


Fig: 2(a) Machine M_1 in Initial Configuration

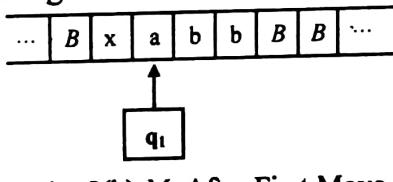


Fig: 2(b) M_1 After First Move

After the first move, the head q_0 reads **a**, removes it with **x** and moves to right, thus changing the state from q_0 to q_1 . This gives us the next configuration as **BxabbBB**. Since there is another **a** in the second state, so we will overwrite the **a** with **a** and move to right ($a \rightarrow a, R$).

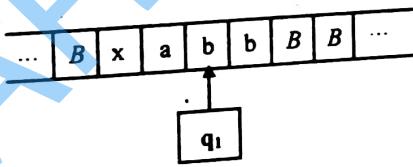


Fig: 2(c) M_1 After Second Move

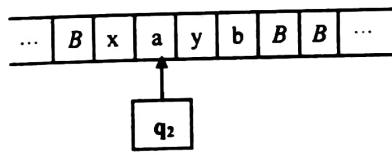


Fig: 2(d)

After the second move the configuration becomes **BxabbBB** with the head pointing to **b**. So it reads **b**, removes it with **y** and moves to left, thus changing the state from q_1 to q_2 . Following the same procedure the turing machine is designed. The next moves are as follows:

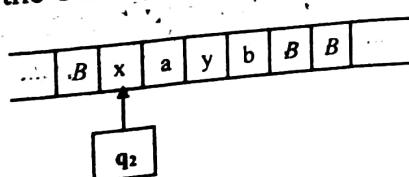


Fig: 2(e)

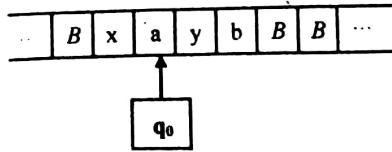


Fig: 2(f)

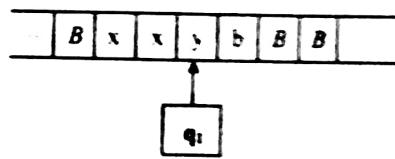


Fig: 2(g)

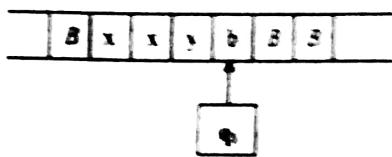


Fig: 2(h)

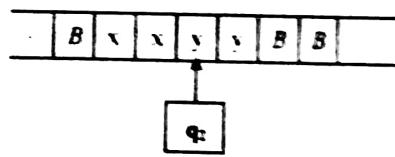


Fig: 2(i)

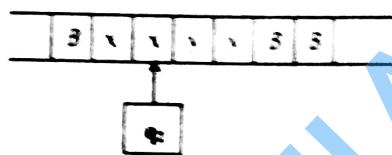


Fig: 2(j)

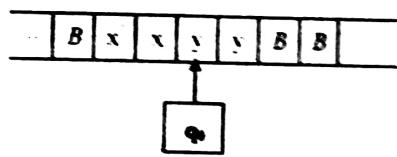


Fig: 2(k)

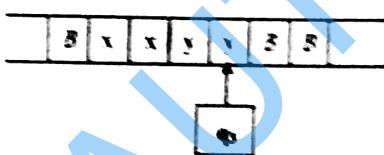


Fig: 2(l)

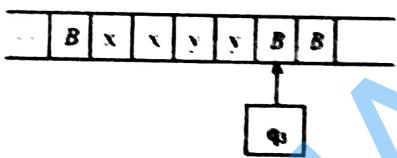


Fig: 2(m)

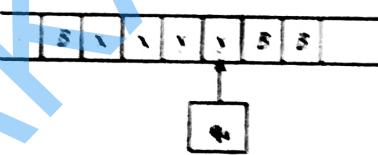


Fig: 2(n)

2nd Part: Refer to Question No. 2 (2nd Part) of Long Answer Type Questions.

5. Define turing machine. Explain Church's hypothesis. What is Universal turing machine? [WBUT 2018]

Answer:

1st Part: Refer to Question No. 3.(a) of Long Answer Type Questions.

2nd Part:

Church's Hypothesis:

∴ This hypothesis has no practical proof.

Weak form:

A Turing Machine can compute anything that can be computed by digital computer

Strong form:

A Turing Machine can perform any possible computation.

i. Church States that —

A function on the natural numbers is computable by a human being following an algorithm, ignoring resource limitations, if and only if it is computed by a Turing Machine.

3rd Part:

Universal Turing Machine

∴ Universal Turing Machine is a Turing Machine for all other turing machines.
∴ For example,

$$A_{TM} = \{\langle M, w \rangle \mid M \text{ is a turing machine and } M \text{ accepts } w\}$$

is Turing Recognizable.

If, M accepts w — Halt & Accept
 M rejects w — Halt & Reject
 M loops on w — no Halt

For Universal Turing Machine

Input: M = description of some TM

w = an input string for M

Action: — simulate M
— Behave just like M
— Behave just like M (accept, reject or loop)