1 UNIT

Basic Concept and Automata Theory

CONTENTS

PART-1

Introduction to Theory of Computation : Automata, Alphabets, Symbol, Strings, Formal Languages, Computability and Complexity.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.1. **What do you understand by term alphabets and string in automata theory ? Explain.**

Answer

Alphabet :

1. An alphabet is a finite, non-empty set of a symbol.
2. As a convention we use the Σ symbol to represent an alphabet. It contains the first lower letters of the Latin alphabet (a, b ...) or digit such as 0, 1 to represent symbols.
   For example :
   $$\Sigma = \{a, b, c\}$$
   It is the alphabet composed by the symbols a, b and c.

String :

1. A string (or word) is a sequence of symbols from some alphabets or digits.
2. For example, given the alphabet
   $$\Sigma = \{a, b, c\}$$
   We can build several strings using the symbols a, b and c :
   a
   abc
   aa
   abcabcabcabcabc

Que 1.2. **Explain the term language in reference to automata theory with example.**

Answer

1. A language is set of all strings chosen from some Σ^*, where Σ is particular alphabet.
2. If Σ is an alphabet and $L \subseteq \Sigma^*$, then L is a language over alphabet Σ.
3. A language over Σ need not include strings with all the symbols of Σ. So once we have established that L is a language over Σ, we also know that it is a language over any alphabet which is superset of Σ.

4. The only important constraint that can be on language is all strings should be finite.
5. Although they can have an infinite number of strings, but restricted to strings drawn from one fixed and finite alphabets.

For example :

1. English words are a set of strings over the alphabets that consist of all the letters.
2. The language of all strings consisting of n 0's followed by n 1's for some $n \geq 0$ is
   $\{\varepsilon, 01, 0011, 000111, ...\}$.
3. The set of strings over 0's and 1's with equal number of 0 and 1 is
   $\{\varepsilon, 01, 10, 0011, 1010, 0101, 1100, ...\}$.

Que 1.3. Explain arithmetic expression with example.

Answer

1. An arithmetic expression (AE) is valid combination of input symbols that are read by computer or human being and able to answer a desired result.
2. The recursive definition may be defined for validity of an arithmetic expression. The definition includes

a. Any number (positive, negative or zero) is in AE.

b. If 'a' is in AE then
   i. (a) is also in AE.
   ii. $-a$ is also in AE.

c. If 'a' and 'b' are in AE, then the following will also be in AE.
   i. $a + b$
   ii. $a - b$
   iii. $a * b$
   iv. a/b
   v. $a \uparrow b$ or $a^{**} b$ (Exponentiation)

For example : If Σ is set of alphabets for a language *i.e.*

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /, (,)\}$$

Then $(4 + 3) * 2$ is a valid expression but $4 + 3) * 2$ is not.

Que 1.4. Define the grammar.

Answer

A grammar or phrase-structured grammar is combination of four tuples and can be represented as $G(V, T, P, S)$. Where,

1. V is finite non-empty set of variables/non-terminals. Generally non-terminals are represented by capital letters like $A, B, C, ..., X, Y, Z$.

2. T is finite non-empty set of terminals. Sometimes also represented by Σ or V_T. Generally terminals are represented by $a, b, c, x, y, z, \alpha, \beta, \gamma$ etc.
3. P is finite set whose elements are in the form $\alpha \rightarrow \beta$. Where α and β are strings, made up by combination of V and T *i.e.*, $(V \cup T)$. α has at least one symbol from V. Elements of P are called productions or production rule or rewriting rules.
4. S is special variable/non-terminal known as starting symbol.

PART-2

Deterministic Finite Automaton (DFA) : Definition, Representation, Acceptability of a String and Language.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.5. What do you understand by Deterministic Finite Automata (DFA) and how it is represented ?

OR

Define Deterministic Finite Automata (DFA).

Answer

1. A Finite Automata (FA) is said to be deterministic, if corresponding to an input symbol, there is single resultant state *i.e.*, there is only one transition.
2. A deterministic finite automata is set of five tuples and represented as

$$M = \{Q, \Sigma, \delta, q_0, F\}$$

Where,

Q : A non-empty finite set of states present in the finite control $(q_0, q_1, q_2, ...)$.

Σ : A non-empty finite set of input symbols.

δ : It is a transition function that takes two arguments, a state and an input symbol, it returns a single state. The δ is represented as

$$\delta : Q * \Sigma \rightarrow Q$$

q_0 : It is starting state, one of the state in Q.

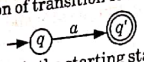
F : It is non-empty set of final states/accepting states from the set belonging to Q.

3. Let 'q' is the state and 'a' be the input symbol passed to the transition function.

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

q' is the output of function, which may be same or new state.

The graphical representation of transition function is as follows :



The state q' is final state and q is the starting state.

Que 1.6. **Construct a DFA for the language that contains the strings ending with 0.**

Answer

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA

q_0 = initial state = $\{q_0\}$

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

F = final state = $\{q_f\}$

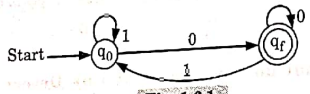


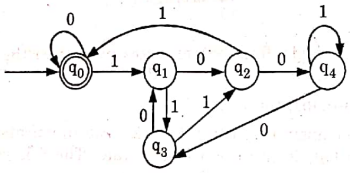
Fig. 1.6.1.

Que 1.7. **Define Deterministic Finite Automata (DFA) and design a DFA that accepts the binary number whose equivalent is divisible by 5.** **AKTU 2017-18, Marks 07**

Answer

DFA : Refer Q. 1.5, Page 1–4A, Unit-1.

Numerical :



Que 1.8. **Draw DFA for following over set $\Sigma = \{0, 1\}$**

i. $L = \{w : |w| \bmod 3 = 0\}$

ii. $L = \{w : |w| \bmod 3 > 1\}$

Answer

i.

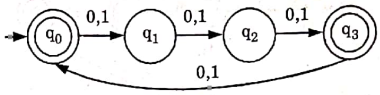


Fig. 1.8.1.

ii.

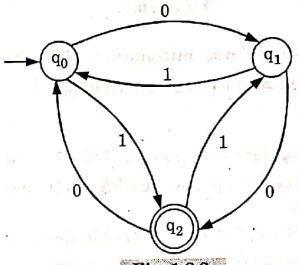


Fig. 1.8.2.

Que 1.9. **Design a FA which accepts set of strings containing exactly four 1's in every string over $\Sigma = \{0, 1\}$.**

AKTU 2014-15, Marks 05

Answer

DFA should accept the strings such as 1111, 0101011, 011011000.....

Let DFA be $M = (Q, \Sigma, \delta, q_0, F)$

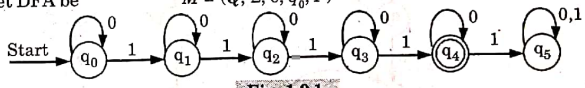


Fig. 1.9.1.

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$F = \{q_4\}$

In this q_5 is dead state.

Que 1.10. **Design a DFA for the language**

$L = \{w$: every run of a's has either two three$\}$.

Answer

The transition diagram for language

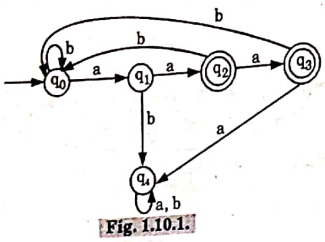


Fig. 1.10.1.

Que 1.11. **Draw the finite automata which accept all the strings containing both 11 and 010 as sub-strings.**

Answer

1. The set of strings for the language is 110101, 1100101, 101011, ...
2. The string will be accepted by the DFA if both the sub-sequence 11 and 010 are present.
3. Both 11 and 010 can follow each other in the language.
4. So, the DFA corresponding to the given language can be represented by the following transition diagram :

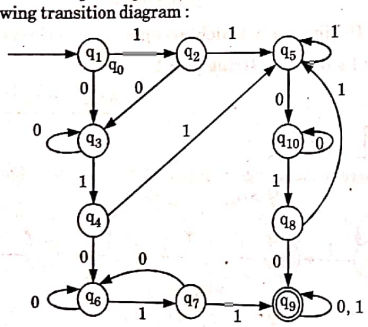


Fig. 1.11.1.

Que 1.12. **Design a finite automata which accepts the complement of the language accepted by the following automata :**

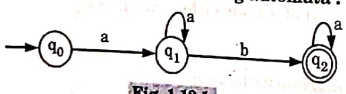


Fig. 1.12.1.

Answer

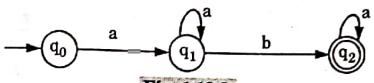


Fig. 1.12.2.

1. We know that if L is regular than complement of L that is $\overline{L}$ is also regular. That is
   $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts language L; then DFA that accepts language $\overline{L}$ is
   $$\overline{M} = (Q_1, \Sigma, \delta, q_0, Q - F)$$
2. DFA which accepts the complement of language L is given by changing every non-final state to final state and every final state to non-final.
3. Thus, the required DFA is

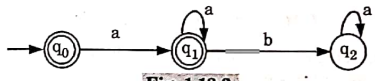


Fig. 1.12.3.

Que 1.13. **Draw DFA of following over {0, 1} :**

i. All strings with even number of 0's and even number of 1's.

ii. All strings of length at most 5.

Answer

i. DFA should accept the strings such as 11, 00, 1010, 0101, 1111, 101101, ... etc. The transition diagram is given by

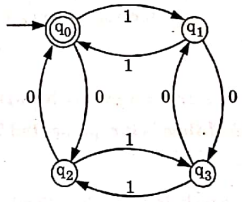


Fig. 1.13.1. Transition diagram.

ii. DFA should accept the string of length 5 such as 10110, 00000, 11111, 01101, ... etc. The transition diagram is given by

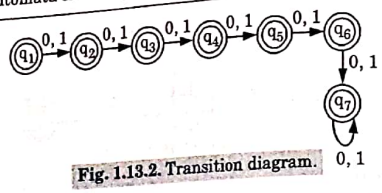


Fig. 1.13.2. Transition diagram.

Que 1.14. What do you mean by language of a DFA ?

Answer

1. The set of all strings that result in a sequence of state transition from start state to an accepting state.
2. Now we can define the language of a DFA $M = (Q, \Sigma, \delta, q_0, F)$. This language is denoted by $L(M)$, and defined by

$$L(M) = \{w / \hat{\delta}(q_0, w) \text{ is in } F\}$$

3. The language of M is the set of strings (w) that take the start state q_0 to one of the accepting states.
4. If L is $L(M)$ for some deterministic finite automata, then we say L is a regular language.

PART-3

Non-Deterministic Finite Automaton (NFA).

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.15. What do you understand by Non-Deterministic Finite Automata (NFA/NDFA) ? How is it represented ?

Answer

1. A finite automata is said to be non-deterministic, if there is more than one possible transition from one state on the same input symbol.
2. A Non-Deterministic Finite Automata is also a set of five tuples and represented as

$$M = \{Q, \Sigma, \delta, q_0, F\} \quad \text{where,}$$

Q : A set of non-empty finite states.
Σ : A set of non-empty finite input symbols.
q_0 : Initial state of NFA and member of Q.

F : A non-empty finite set of final states and member of Q.
δ : It is transition function that takes a state from Q and an input symbol from Σ and returns a subset of Q. The δ is represented as

$$\delta : Q * (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$$

3. The graphical representation of δ is as follows :

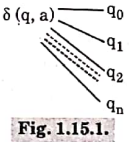


Fig. 1.15.1.

Que 1.16. Construct a NFA for the language L which accept all the strings in which the third symbol from right end is always a over $\Sigma = \{a, b\}$. **AKTU 2015-16, Marks 10**

Answer

Let NFA be

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

The δ is defined as follows :

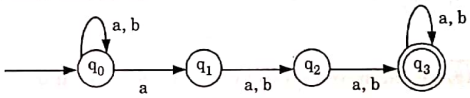


Fig. 1.16.1.

Que 1.17. Design a NFA for the language L which accepts all strings over {0, 1} that have at least two consecutive 0's or 1's.

Answer

By the analysis it is clear that NFA will accept the strings of the patterns like 00, 11, 101000, 101100, ...
The transition diagram is given by

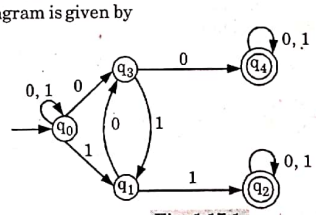


Fig. 1.17.1.

Que 1.18. **What do you understand by language of a NFA ?**

Answer

1. An NFA accepts a string w if it is possible to make any sequence of choice of next state while reading the characters of w, and go from the start state to any accepting state.
2. The language of NFA $M = (Q, \Sigma, \delta, q_0, F)$ is defined by

$$L(M) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \phi\}$$

3. $L(M)$ is the set of strings w in Σ^* such that $\hat{\delta}(q_0, w)$ contains at least one accepting state.

PART-4

Equivalence of DFA and NFA, NFA with ε-Transition, Equivalence of NFA with and without ε-Transition.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.19. **What do you mean by NFA with ε-transition ? Why we need it ?**

Answer

If a finite automata is modified to permit transition without input symbols, along with zero, one or more transitions on input symbols, then we get NFA with ε-transitions.

We need it to concatenate two different languages.

For example : We have design NFA for accepting language

$$L = \{(ab)^* \cup aa^*\}$$

To solve this problem first divide the language

$$L = L_1 \cup L_2, \quad \text{Where } L_1 = (ab)^* \text{ and } L_2 = aa^*$$

First construct NFA for L_1

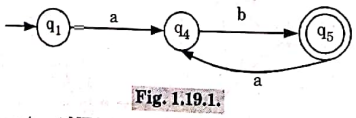


Fig. 1.19.1.

Secondly, we construct NFA for L_2

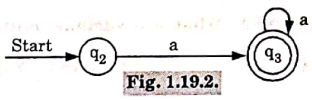


Fig. 1.19.2.

Now we combine two transition diagram *i.e.*, Fig. 1.19.1 and 1.19.2 as follows :

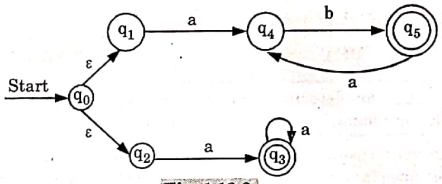


Fig. 1.19.3.

In this transition diagram we use ε-transitions, to reach at states q_1 and q_2.

Que 1.20. **Explain the working of ε-closure, with a suitable example.**

Answer

1. A string w in Σ^* will be accepted by NFA with ε-transition, if there exists at least one path corresponding 'w' which starts from initial state and ends at final state.
2. Since, this path may or may not contain ε-transition.
3. If the path contains ε-moves, then we need to define a function ε-closure (q), where 'q' is state of automata.
4. The function ε-closure is defined as follows :

ε-closure (q) = set of all those states of automata which can be reached from 'q' on a path labeled by ε *i.e.*, without consuming any input symbol. Consider the NFA

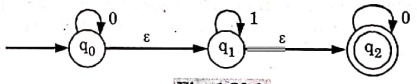


Fig. 1.20.1.

ε-closure (q_0) = $\{q_0, q_1, q_2\}$

ε-closure (q_1) = $\{q_1, q_2\}$

ε-closure (q_2) = $\{q_2\}$

Que 1.21. **Define NFA. What are various points of difference between NFA and DFA?** **AKTU 2018-19, Marks 10**

Answer

NFA : Refer Q. 1.15, Page 1–9A, Unit-1.

S. No.	DFA	NFA
1.	It stands for deterministic finite automata.	It stands for non-deterministic finite automata.
2.	Only one transition is possible from one state to another on same input symbol.	More than one transition is possible from one state to another on same input symbol.
3.	Transition function δ is written as : $\delta : Q \times \Sigma \rightarrow Q$	Transition function δ is written as : $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$
4.	In DFA, ε-transition is not possible.	In NFA, ε-transition is possible.
5.	DFA cannot be converted into NFA.	NFA can be converted into DFA.

Que 1.22. **Can we convert a NFA into a DFA ?**

Answer

1. Yes, we can convert a NFA into DFA. For every NFA there exists an equivalent DFA.
2. The equivalence is defined in terms of language acceptance. Since NFA is nothing but a finite automata in which zero, one or more transition on an input symbols are permitted.
3. We can always construct finite automata which will simulate all moves of NFA on a particular input symbol in parallel, then get a finite automata in which there will be exactly one transition on every input symbol.

Que 1.23. **Explain the procedure for converting NFA to equivalent DFA.**

Answer

Procedure for converting NFA to equivalent DFA :

1. Let $M_2 = \{Q_2, \Sigma, q'_0, \delta_2, F_2\}$ be NFA that recognizes the languages L. Then the DFA $M = \{Q, \Sigma, q_0, \delta, F\}$ that satisfies the following conditions recognizes L :

$Q = 2^{Q_2}$, *i.e.*, is the set of all subsets of Q_2.

$q_0 = \{q'_0\}$

$\delta(q, a) = \bigcup_{p \in q} \delta(p, a)$ for each state q in Q and each symbol a in Σ and

$F = \{q \in Q \mid q \cap F_2 \neq \phi\}$.

2. To obtain a DFA $M = \{Q, \Sigma, \delta, q_0, F\}$ which accepts the same language as given NFA $M_2 = \{Q_2, \Sigma, q'_0, \delta_2, F_2\}$ does, we may proceed as follows :

Step 1 : Initially $Q = \phi$.

Step 2 : Put $\{q'_0\}$ into Q. $\{q'_0\}$ is the initial state of the DFA M.

Step 3 : Then for each state q in Q do the following :

a. add this new state.

b. add $\delta(q, a) = \bigcup_{p \in q} \delta(p, a)$ to δ, where the δ on the right hand side is that of NFA M_2.

Step 4 : Repeat step 3 till new states are there to add in Q, if there is new state found to add in Q, the process terminates. All the states of Q that contain accepting states of M_2 are accepting states of M.

Que 1.24. **Construct DFA equivalent to NFA where δ is defined in the following table :**

Q	$\delta(q, a)$	$\delta(q, b)$
A	A, B	C
B	A	B
C*(final state)	–	A, B

AKTU 2015-16, Marks 10

Answer

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

$\delta(A, b) = \{C\}$

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

$\delta(\{A, B\}, b) = \delta(A, b) \cup \delta(B, b) = \{C\} \cup \{B\} = \{B, C\}$

$\delta\{C, a\} = \{\phi\}$

$\delta\{C, b\} = \{A, B\}$

$\delta(\{B, C\}, a) = \delta(B, a) \cup \delta(C, a) = \{A\} \cup \{\phi\}$
$= \{A\}$

$\delta(\{B, C\}, b) = \delta(B, b) \cup \delta(C, b) = \{B\} \cup \{A, B\}$
$= \{A, B\}$

Let, $A \rightarrow q_0$

$C \rightarrow q_1$

$\{A, B\} \rightarrow q_2$

$\{B, C\} \rightarrow q_3$

Transition table for DFA,

δ/Σ	a	b
$\rightarrow q_0$	q_2	q_1
$*q_1$	$\{\phi\}$	q_2
q_2	q_2	q_3
$*q_3$	q_0	q_2

Transition diagram for DFA,

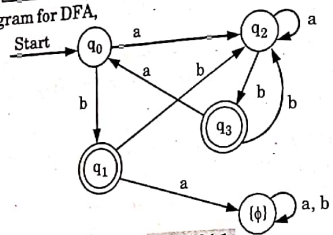


Fig. 1.24.1.

Here $\{\phi\}$ state is nothing but interpreted as if there is a input 'a' on state q, then there is no path for it and machine will be in dead stage.

Que 1.25. Convert the following NFA $\{p, q, r, s\}, \{0, 1\}, \delta, p, \{q, s\}$ into DFA where δ is given by

δ/Σ	0	1
$\rightarrow p$	q, s	q
$*q$	r	q, r
r	s	p
$*s$	ϕ	p

AKTU 2018-19, Marks 10

Answer

The transition table for DFA will be :

δ/Σ	0	1
p	$\{q, s\}$	$\{q\}$
q	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
s	–	$\{p\}$
$\{q, s\}$	$\{r\}$	$\{p, q, r\}$
$\{q, r\}$	$\{r, s\}$	$\{p, q, r\}$
$\{r, s\}$	$\{s\}$	$\{p\}$
$\{p, q, r\}$	$\{q, r, s\}$	$\{p, q, r\}$
$\{q, r, s\}$	$\{r, s\}$	$\{p, q, r\}$

Now let, $p \rightarrow A$
$q \rightarrow B$
$r \rightarrow C$
$s \rightarrow D$
$\{q, s\} \rightarrow E$
$\{q, r\} \rightarrow F$
$\{r, s\} \rightarrow G$
$\{p, q, r\} \rightarrow H$
$\{q, r, s\} \rightarrow I$

Therefore, transition table for DFA after relabeling and tuples will be :

δ/Σ	0	1
$\rightarrow A$	E	B
$*B$	C	F
C	D	A
$*D$	–	A
$*E$	C	H
$*F$	G	H
$*G$	D	A
$*H$	I	H
$*I$	G	H

$Q = \{A, B, C, D, E, F, G, H, I\}$
$\Sigma = \{0, 1\}$
$q_0 = \{A\}$
$F = \{B, D, E, F, G, H, I\}$

∴ Transition diagram will be :

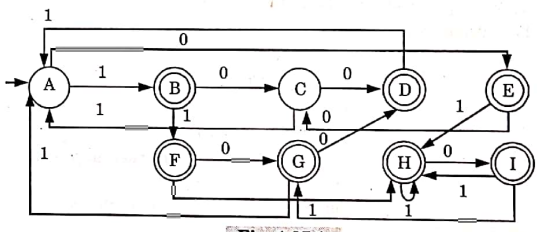


Fig. 1.25.1.

Que 1.26. Construct a NFA without ε-moves corresponding to following ε-NFA. Explain each step.

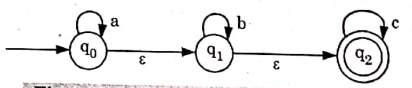


Fig. 1.26.1. NFA with ε-transition (ε-NFA).

Answer

Transition table for ε-NFA

Σ/δ	a	b	c	ε
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_2	ϕ	q_2
q_2	ϕ	ϕ	q_2	ϕ

Step 1 : Find the states of NFA without ε-transition including initial state and final states as follows :

i. There will be the same number of states but the names can be constructed by writing the state name as the set of states in the ε-closure.

ii. Initial state will be ε-closure of initial state of NFA with ε-transition in Fig. 1.26.1 as

$\varepsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$ (New initial state for NFA, without ε-transition)

Rest of the states are

$\varepsilon\text{-closure}(q_1) = \{q_1, q_2\}$ (New state)

$\varepsilon\text{-closure}(q_2) = \{q_2\}$ (New state)

iii. The final states of NFA without ε-transition are all those new states which contain final state of NFA with ε-transition as member.
So $\{q_0, q_1, q_2\}$, $\{q_1, q_2\}$ and $\{q_2\}$ all are final states.

So if NFA without ε-transition is

$$M' = (Q', \Sigma, \delta', q_0', F')$$
$$Q' = (\{q_0, q_1, q_2\}, \{q_1, q_2\}, \{q_2\})$$
$$q_0' = \{q_0, q_1, q_2\}$$
$$F' = (\{q_0, q_1, q_2\}, \{q_1, q_2\}, \{q_2\})$$

Step 2 : Now we have to decide δ' to find out the transitions as follows :

$$\delta'(\{q_0, q_1, q_2\}, a) = \varepsilon\text{-closure}(\delta(q_0, q_1, q_2), a)$$
$$= \varepsilon\text{-closure}(\delta(q_0, a) \cup \delta(q_1, a), a) \cup \delta(q_2, a))$$
$$= \varepsilon\text{-closure}(q_0 \cup \phi \cup \phi)$$
$$= \varepsilon\text{-closure}(q_0)$$
$$= \{q_0, q_1, q_2\}$$

Similarly

$$\delta'(\{q_0, q_1, q_2\}, b) = \varepsilon\text{-closure}(\delta(q_0, q_1, q_2), b)$$
$$= \varepsilon\text{-closure}(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b))$$
$$= \varepsilon\text{-closure}(\phi \cup q_1 \cup \phi)$$
$$= \varepsilon\text{-closure}(q_1)$$
$$= \{q_1, q_2\}$$

$$\delta'(\{q_0, q_1, q_2\}, c) = \varepsilon\text{-closure}(\delta(q_0, q_1, q_2), c)$$
$$= \varepsilon\text{-closure}(\delta(q_0, c) \cup \delta(q_1, c) \cup \delta(q_2, c))$$
$$= \varepsilon\text{-closure}(\phi \cup \phi \cup q_2)$$
$$= \varepsilon\text{-closure}(q_2)$$
$$= \{q_2\}$$

Similarly we can write,

$$\delta'(\{q_1, q_2\}, a) = \phi$$
$$\delta'(\{q_1, q_2\}, b) = \{q_1, q_2\}$$
$$\delta'(\{q_1, q_2\}, c) = \{q_2\}$$
$$\delta'(q_2, a) = \phi$$
$$\delta'(q_2, b) = \phi$$
$$\delta'(q_2, c) = \{q_2\}$$

So transition table for NFA without ε-transition will be as following :

δ'/Σ	a	b	c
→ * $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
* $\{q_1, q_2\}$	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
* $\{q_2\}$	ϕ	ϕ	$\{q_2\}$

Let us say $\{q_0, q_1, q_2\}$ as q_x

$\{q_1, q_2\}$ as q_y

and $\{q_2\}$ as q_z

So transition diagram is as follows :

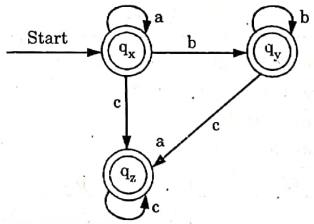


Fig. 1.26.2. NFA without ε-transitions.

Que 1.27.

i. Convert the NFA-ε to DFA.

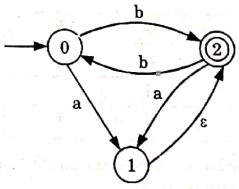


Fig. 1.27.1.

ii. Check with the comparison method for testing equivalence of two FA given below.

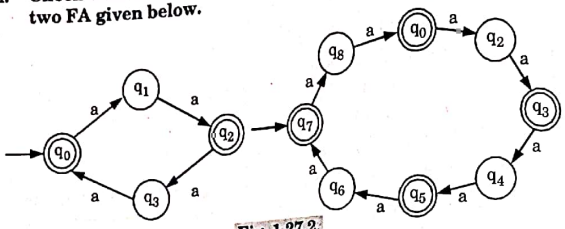


Fig. 1.27.2.

AKTU 2016-17, Marks 10

Answer

i. First we convert NFA-ε to NFA.

Transition table for NFA-ε :

δ/Σ	a	b	ε
0	1	2	
1	ϕ	ϕ	2
2	1	0	

ε-closure of (0) = {0}

ε-closure of (1) = {1, 2}

ε-closure of (2) = {2}

Transition table for NFA :

δ/Σ	a	b
{0}	{1, 2}	{2}
{1, 2}	{1, 2}	{2}
{2}	{1, 2}	{0}

Let {0} as A
{1, 2} as B
{2} as C

Transition table for NFA :

δ/Σ	a	b
A	B	C
B	B	C
C	B	A

Transition table for DFA will be same as NFA because there is only one transition from A with input a.
So DFA is given as

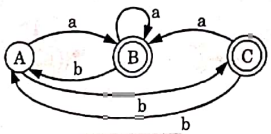


Fig. 1.27.3.

ii. Language accepted by both FAs

$L = \{w \in \{a\}/\text{string with even number of } a\}$

So, both, FAs are equivalent as they accept same set of string.

Que 1.28. Compute the epsilon-closure for the given NFA. Convert it into DFA.

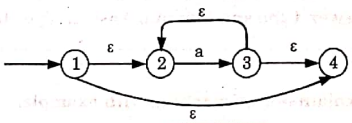


Fig. 1.28.1.

AKTU 2016-17, Marks 7.5

Answer

ε-closure of (1) = {1, 2, 4}
ε-closure of (2) = {2}
ε-closure of (3) = {2, 3, 4}
ε-closure of (4) = {4}

Transition table for ε-NFA :

δ/Σ	a	ε
1	ϕ	{2, 4}
2	3	ϕ
3	ϕ	{4, 2}
4	ϕ	ϕ

Transition table for NFA :

Let {1, 2, 4} as A
{2, 3, 4} as B

δ/Σ	a
{1, 2, 4}	{2, 3, 4}
{2}	{2, 3, 4}
{2, 3, 4}	{2, 3, 4}
{4}	ϕ

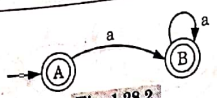


Fig. 1.28.2.

PART-5

Finite Automata with Output : Moore Machine, Mealy Machine, Equivalence of Moore and Mealy Machine.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.29. Explain Moore machine with example.

Answer

1. Moore machine is a finite automata in which output is associated with each state.
2. The output symbol at a given time depends only upon the present state of machine.
3. Mathematically, Moore machine is a six tuple machine and defined as

$$M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$$

where,

Q : A non-empty finite set of states.

Σ : A non-empty finite set of input symbol.

Δ : A non-empty finite set of outputs.

δ : A transition function which takes two arguments as in finite automata, one is input state and another is input symbol. The output of this function is a single state and represented by

$$\delta : Q * \Sigma \rightarrow Q$$

λ : It is a mapping function, which maps Q to Δ, giving the output associated with each state. It is represented by $\lambda : Q \rightarrow \Delta$

q_0 : Initial state of machine.

4. When machine is in a particular state, it produces the output, irrespective of what was the input on which transition is made.

Representation of Moore machine :

Moore machine can be represented by transition table as well as transition diagram same as finite automata.

Let M be Moore machine having following transition table with $\Sigma = \{0, 1\}$

Present state	Next state at Input		Output
	$a = 0$	$a = 1$	
$\rightarrow q_0$	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

$$M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$$

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

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

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

$$\lambda(q_0) = 0, \lambda(q_1) = 1, \lambda(q_2) = 0, \lambda(q_3) = 0$$

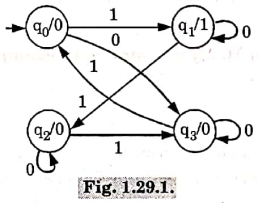


Fig. 1.29.1.

q_0 is the initial state.

Que 1.30. Design a Moore machine which determines the residue Mod-3 for each binary string treated as binary integer.

Answer

We have to design a Moore machine which prints the remainder, when decimal equivalent is divided by 3.

Therefore, $\Delta = \{0, 1, 2\}$

Since, 0, 1 and 2 are possible remainder, when a decimal number is divided by the 3.

Therefore, we need three states in Moore machine.

Let Moore machine $M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$

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

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

$$\Delta = \{0, 1, 2\}$$

$$\lambda(q_0) = 0, \lambda(q_1) = 1, \lambda(q_2) = 2$$

q_0 is initial state.

The transition table is

Present state	Next state at Input		Output
	$a = 0$	$a = 1$	
$\rightarrow q_0$	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

and transition diagram will be

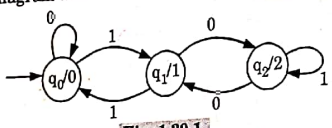


Fig. 1.30.1.

Let us assume input string is 111, decimal equivalent is 7.7/3 = 1 (remainder) when we pass 111 to M, it ends at q_1, and $\lambda(q_1) = 1$, which is remainder of 7/3.

Que 1.31. Explain Mealy machine with example.

Answer

Mealy machine :

1. It is finite automata in which output is associated with each transition.
2. In Mealy machine, every transition for a particular input symbol has a fixed output.
3. Mathematically, Mealy machine is a six tuple machine and defined as

Where, $M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$

Q : A non-empty finite set of state in M.

Σ : A non-empty finite set of input symbol.

Δ : A non-empty finite set of outputs.

q_0: Initial state of M.

δ : A transition function which takes two arguments, one is input state and another is input symbol. The output of this function is a single state and represented as $\delta : Q * \Sigma \rightarrow Q$

λ: It is mapping/machine function which maps $Q * \Sigma$ to Δ, giving the output associated with each transition. It is represented by

λ: $Q * \Sigma \rightarrow \Delta$

Thus, for this type of machine output depends on both current state and the current input symbol.

Representation of Mealy machine :

Mealy machine can be represented by transition table as well as transition diagram.

Let M be a Mealy machine having following transition table with $\Sigma = \{0, 1\}$.

Present state	For input $a = 0$		For input $a = 1$	
	State	Output	State	Output
q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

$M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$

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

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

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

$\lambda(q_1, 0) = 0$ $\quad \lambda(q_1, 1) = 0$

$\lambda(q_2, 0) = 1$ $\quad \lambda(q_2, 1) = 0$

$\lambda(q_3, 0) = 1$ $\quad \lambda(q_3, 1) = 1$

$\lambda(q_4, 0) = 1$ $\quad \lambda(q_4, 1) = 0$

The transition diagram

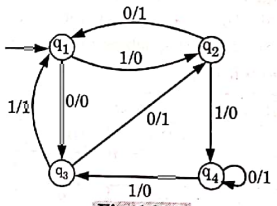


Fig. 1.31.1.

Here the output is associated with each input.

Que 1.32. Design a Mealy machine, which prints 1's complement of input bit string over alphabet $\Sigma = \{0, 1\}$.

Answer

If input string is 101110 then 1's complements of 101110 will be 010001, so, we have to design a Mealy machine which will point output 010001 for input string 101110.

Let $M_e = (Q, \Sigma, \Delta, \delta, \lambda', q_0)$

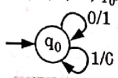


Fig. 1.32.1.

$Q = \{q_0\}$

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

q_0 is also initial state.
Let us pass input $w = 011$ to M_t

$$q_0 \xrightarrow{0/1} q_0 \xrightarrow{1/0} q_0 \xrightarrow{1/0} q_0$$

Output string is $w' = 100$ which 1's complement of $w = 011$.

Que 1.33. Write the procedure for converting a Moore machine to a Mealy machine with an example.

Answer

1. Draw the tabular format of Mealy machine.
2. Put the present states of the Moore machine in the present state column of constructing Mealy machine.
3. Put the next states of Moore machine for corresponding present states and input combination, in the next state columns of the constructing Mealy machine for the same present states and input combination.
4. For the output, look into the present state column and output column of the Moore machine. The output for Q_{Next} (Next state for present state $Q_{Present}$ and input I/P of the constructing Mealy machine) will be the output of that state (Q_{Next}) as a present state in the given Moore machine.

Example : Conversion of Moore machine to Mealy machine.

Given Moore machine :

Present state	Next state		O/P
	I/P = 0	I/P = 1	
$\rightarrow q_0$	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_3	0

Mealy machine :

Present state	I/P = 0		I/P = 1	
	Next state	O/P	Next state	O/P
$\rightarrow q_0$	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_3	0

In the Moore machine, for q_1 as a present state the output is 1. So in the constructing Mealy machine for q_1 as a next state the output will be 1.

In the Moore machine, for q_2 as a present state the output is 0. Hence in the constructing Mealy Machine for q_2 as a next state the output will be 0.

Que 1.34. What are various points of difference between Moore and Mealy machine ? Explain the procedure to convert a Moore machine into Mealy machine. **AKTU 2018-19, Marks 10**

Answer

Difference between Moore and Mealy machine :

S. No.	Moore machine	Mealy machine
1.	Its output depends only on present state.	Its output depends on the transition (input) and present state.
2.	Its transition function is excess into Q (finite set of states).	Its transition function is excess into D (output alphabet).
3.	If input string is of length n the output string is of length $n + 1$.	If input string is of length n then the output string is of length n.
4.	At the different input on the same state, its output is same.	At the different input on the same state, its output is different.

Procedure to convert a Moore machine into Mealy machine : Refer Q. 1.33, Page 1–25A, Unit-1.

Que 1.35. Write the procedure for converting Mealy machine to Moore machine with example.

Answer

1. Draw the tabular format of Mealy machine.
2. Check the next state and output columns of the given Mealy machine.
3. If for same next state in the next state column, the output differs in the output column, break the state q_i into different number of states. The number is equal to the number of different outputs associated with q_i.
4. Put the states of the present state column in the new table. The states which are broken into number of different states put the broken states in the place of those states.
5. Change the next states in the next state columns according to the new set of states.
6. Put the output from the output column of the original Mealy machine in the new machine.
7. Draw the tabular format of Moore machine.

8. Put the present states and next states from the constructed new Mealy machine to the constructed Moore machine.
9. For output check the next state and output column of the new constructed Mealy machine. For the state let q_i as a next state in the new constructed Mealy machine if output is 0 then for q_i as a present state in the constructing Moore machine; the output will be 0.
10. For Moore machine the output depends only on the present state. This means from the beginning state for ε-input we can get an output. If the output is 1, the new constructed Moore machine can accept zero length string, which was not accepted by given Mealy machine.
11. To make the Moore machine does not accept ε-string, we have to add an extra state q_b (new beginning state), whose state transactions will be identical with those of the existing beginning state but output is 0.

Example : Conversion of Mealy machine to Moore machine.

Given Mealy machine :

Present state	I/P = 0		I/P = 1	
	Next state	O/P	Next state	O/P
$\rightarrow q_0$	q_0	1	q_1	0
q_1	q_3	1	q_3	1
q_2	q_1	1	q_2	1
q_3	q_2	0	q_0	1

Check the next state and output columns of the given Mealy machine. For I/P 0 for q_1 as a next state output is 1. For I/P 1 for q_1 as a next state output is 0. Same thing happens for q_2 as a next state for input 0 and input 1. So the state q_1 will be broken as q_{10} and q_{11} and the state q_2 will be broken as q_{20} and q_{21}. After breaking the modified Mealy machine will be

Present state	I/P = 0		I/P = 1	
	Next state	O/P	Next state	O/P
$\rightarrow q_0$	q_0	1	q_{10}	0
q_{10}	q_3	1	q_3	1
q_{11}	q_3	1	q_3	1
q_{20}	q_{11}	1	q_{21}	1
q_{21}	q_{11}	1	q_{21}	1
q_3	q_{20}	0	q_0	1

For present state q_0 for input 1 the next state will be q_{10} because there is no q_1 in the modified Mealy machine. It has been broken into q_{10} and q_{11} depending on the output 0 and 1, respectively.

From this the Moore machine will be

Present state	Next state		O/P
	I/P = 0	I/P = 1	
$\rightarrow q_0$	q_0	q_{10}	0
q_{10}	q_3	q_3	1
q_{11}	q_3	q_3	1
q_{20}	q_{11}	q_{21}	1
q_{21}	q_{11}	q_{21}	1
q_3	q_{20}	q_0	1

For Moore machine beginning states q_0 and the output is 1. That means with null length input (no input) we are getting an output 1. Therefore, the Moore machine accepts 0 length sequence which is not acceptable for mealy machine. To overcome this situation we must add a new beginning state q_b with same transactions as q_0 but output 0. By including the new state the Moore machine will be :

Moore machine :

Present state	Next state		O/P
	I/P = 0	I/P = 1	
$\rightarrow q_b$	q_0	q_{10}	0
q_0	q_0	q_{10}	1
q_{10}	q_3	q_3	0
q_{11}	q_3	q_3	1
q_{20}	q_{11}	q_{21}	0
q_{21}	q_{11}	q_{21}	1
q_3	q_{20}	q_0	1

Que 1.36. **Describe Mealy and Moore machines with example. Convert the given Mealy machine as shown in Fig. 1.36.1 into Moore machine.**

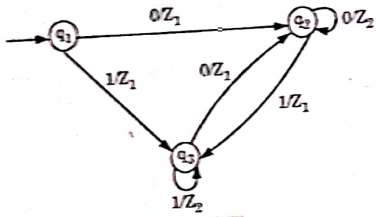


Fig. 1.36.1.

AKTU 2017-18, Marks 07

Answer

Mealy machine : Refer Q. 1.31, Page 1–23A, Unit-1.
Moore machine : Refer Q. 1.29, Page 1–21A, Unit-1.
Numerical :

1. Let us convert the transition diagram into the transition Table 1.36.1.
2. For the given problem : q_1 is not associated with any output; q_2 is associated with two different outputs Z_1 and Z_2; q_3 is associated with two different outputs Z_1 and Z_2.
3. Thus we must split q_2 into q_{21} and q_{22} with outputs Z_1 and Z_2, respectively and q_3 into q_{31} and q_{32} with outputs Z_1 and Z_2, respectively. Table 1.36.1 may be reconstructed as Table 1.36.2.

Table 1.36.1. Transition table

Present state	Next state			
	$a = 0$		$a = 1$	
	state	output	state	output
$\rightarrow q_1$	q_2	Z_1	q_3	Z_1
q_2	q_2	Z_2	q_3	Z_1
q_3	q_2	Z_1	q_3	Z_2

Table 1.36.2. Transition table of Moore machine

Present state	Next state		Output
	$z = 0$	$a = 1$	
$\rightarrow q_1$	q_{21}	q_{31}	
q_{21}	q_{22}	q_{31}	Z_1
q_{22}	q_{22}	q_{31}	Z_2
q_{31}	q_{21}	q_{32}	Z_1
q_{32}	q_{21}	q_{32}	Z_2

Fig. 1.36.1 gives the transition diagram of the required Moore machine.

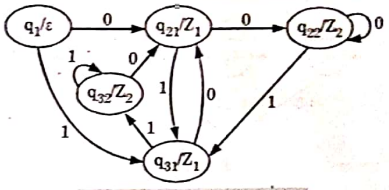


Fig. 1.36.1. Moore machine.

PART-6

Minimization of Finite Automata, Myhill-Nerode Theorem, Simulation of DFA and NFA.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.37. What is the need of minimization of finite automata ?

Answer

1. The language (regular expression) produced by a DFA is always unique.
2. However, the reverse, *i.e.*, a language produces a unique DFA is not true.
3. Hence for a given language there may be different DFAs.
4. By minimizing we can get a minimized DFA with minimum number of states and transitions which produces that particular language.
5. DFA determines how computers manipulate regular languages (expressions). DFA size determines space/time efficiency.
6. Hence from a DFA with minimized states need less time to manipulate a regular expression.

Que 1.37. Define dead state, inaccessible state, equivalent state, distinguishable state and *k*-equivalence in case of finite automata.

Answer

1. **Dead state :** A state q_i is called dead state if q_i is not a final state and for all the inputs to this state the transitions confined to that state. In mathematical notation, we can denote $q_i \notin F$ and $\delta(q_i, \Sigma) \rightarrow q_i$.

2. **Inaccessible state :** The states which can never be reached from the initial state are called in accessible state.

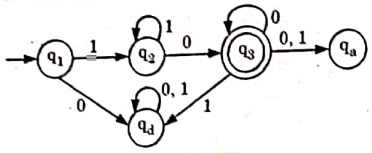


Fig. 1.38.1.

Here q_d is dead state and q_a is inaccessible state.

3. **Equivalent state :** Two states q_i and q_j of a finite automata M are called equivalent if $\delta(q_i, x)$ and $\delta(q_j, x)$ both produces final states or both of them produces non-final states for all $x \in \Sigma^*$. It is denoted by $q_i \equiv q_j$.
4. **Distinguishable state :** Two states q_i and q_j of a finite automata M are called distinguishable if for a minimum length string x, for $\delta(q_i, x)$ and $\delta(q_j, x)$ one produces final state and another produces non-final state or vice versa for all $x \in \Sigma^*$.
5. ***k*-equivalent :** Two states q_i and q_j of a finite automata M are called k-equivalent ($k \geq 0$) if $\delta(q_i, x)$ and $\delta(q_j, x)$ both produces final states or both of them produces non-final states for all $x \in \Sigma^*$ of length k or less.

Que 1.39. Construct a minimum state automaton from the transition table given below.

PS	NS	
	I/P = 0	*I/P* = 1
A	*F*	*B*
B	*C*	*G*
C	*C*	*A*
D	*G*	*C*
E	*F*	*H*
F	*G*	*C*
G	*E*	*G*
H	*C*	*G*

***A* is initial state and *C* is final state**

Answer

1. In the finite automata the states are $\{A, B, C, D, E, F, G, H\}$.
   $S_0 : \{A, B\ C, D, E, F, G, H\}$.

2. All of the states in S_0 are 0-equivalent.
3. In the finite automata there are two types of states final state and non-final states.
4. Hence divide the set of states into two parts Q_1 and Q_2.
   $Q_1 = \{C\}$, $Q_2 = \{A, B, D, E, F, G, H\}$,
   $S_1 : \{\{C\}\ \{A, B, D, E, F, G, H\}\}$.
5. States belong to same subset are 1-equivalent because they are in the same set for string length 1.
6. States belong to different subsets are 1-distinguishable.
7. The C is single state, hence it cannot be divided. Among the states $\{A, B, D, E, F, G, H\}$ for $\{B, D, F, H\}$ for input either 0 or 1, the next state belong to $\{C\}$ which is different subset. So,
   $S_2 : \{\{C\}, \{A, E, G\}, \{B, D, F, H\}\}$.
8. Similarly if we divide further,
   $S_3 : \{\{C\}, \{A, E\}, \{G\}, \{B, H\}, \{D, F\}\}$.
9. The subsets cannot be divided further. Hence these are the states of minimized DFA.
10. Let $\{C\} \rightarrow q_0$, $\{A, E\} \rightarrow q_1$, $\{G\} \rightarrow q_2$, $\{B, H\} \rightarrow q_3$, $\{D, F\} \rightarrow q_4$
11. Initial state was A, hence here initial states is $\{A, E\}$, *i.e.*, q_1.
12. Final state was C, hence here final state is $\{C\}$, *i.e.*, q_0.
13. The tabular representation of minimized DFA is

PS	NS	
	I/P = 0	*I/P* = 1
$\textcircled{q_0}$	q_0	q_1
$\rightarrow q_1$	q_4	q_3
q_2	q_1	q_2
q_3	q_0	q_2
q_4	q_2	q_0

Que 1.40. State and prove Myhill-Nerode theorem.

Answer

The Myhill-Nerode Theorem : Let $L \subseteq \Sigma^*$ be a language. The following three statements are equivalent :

1. $L \subseteq \Sigma^*$ is regular, that is, $L = L(A)$ for some DFA A.
2. L is the union of some of the equivalent classes of a right-invariant equivalence relation on all strings taken from Σ^* of finite index.
3. Let an equivalence relation R_L on Σ^* be defined by :
   uR_Lv if and only if for all $w \in \Sigma^*$; $uw \in L \Leftrightarrow uv \in L$
   Then R_L is of finite index.

The Myhill-Nerode theorem gives yet another characterization of regular languages. Its proof involves the construction of minimal automata, which is an important technique in its own right.

Proof :

1. Any equivalence relation R satisfying 2 could be shown is a refinement of R_L, that means, every equivalence class of R is entirely contained in some equivalence class of R_L. This way, the index of R_L cannot be greater than the index of R and therefore, it is finite.
2. Let us assume that uRv. Since R is right invariant, then for every $w \in \Sigma$, we have $uwRvw$, and therefore, uR_Lv. Thus, R is a refinement of uR_Lv.
3. We first prove that R_L is right invariant. Let us assume that uR_Lv. For any $y \in \Sigma^*$ we must show that uyR_Lvy, that means, for any $z \in \Sigma^*$, $uyz \in L \Leftrightarrow vyz \in L$. But this follows from the definition of R_L in 3, if we put $w = yz$.
4. We now construct a DFA A_{MY} accepting L. The basic technique is to use states as the equivalence classes of R_L (of which there is only a finite number). So,
   $A_{MN} = (Q, \Sigma, \delta, q_0, f) = (\{[w]_{RL} \mid w \in \Sigma^*\}, q_0, \delta_{MN}, F_{MN})$ we put
   i. $q_0 = [\varepsilon]_{RL}$
   ii. For all $w \in \Sigma^*$, $a \in \Sigma$ and $\delta_{MN}([w]_{RL}, a) = [wa]_{RL}$
   iii. $F_{MN} = \{[w]_{RL} \mid w \in L\}$
5. We first have to show that this is well-defined, especially that (ii) makes sense, that is, the definition of δ_{MN} is independent of the choice of representative w of the equivalence class $[w]_{RL}$. So we have to show that if wR_Lv, then $[w]_{RL} = \delta_{MN}([w]_{RL}, a) = \delta_{MN}[v]_{RL}, a) = [va]_{RL}$. But if wR_Lv, then waR_Lva because R_L is right-invariant, therefore $[wa]_{RL} = [va]_{RL}$. So δ_{MN} is well-defined.
6. Finally, we have to show that $L = L(A_{MN})$. This follows from $E_{MN}([\varepsilon]_{RL}, w) = [w]_{RL}$, where $E(q, x) = \varepsilon$ – closure $(\delta(q, x))$ and ε – closure (q) is set of all states reachable from q on input ε.

Que 1.41. How Myhill-Nerode theorem can be applied in minimizing a DFA ?

Answer

Step 1 : Build a two-dimensional matrix labelled by the states of the given DFA at the left and bottom side. The major diagonal and the upper triangular part will be put dash.

Step 2 : One of the three symbols, X, x or 0 will be put in the locations where is no dash.

a. Mark X at p, q in lower triangular part such that p is final state and Q is non-final state.

b. Make distinguished pair combination of the non-final states. If there are n number of non-final states there will be nC_2 number of distinguished pairs.
   Take a pair (p, q) and find (r, s), such that $r = \delta(p, a)$ and $s = \delta(q, a)$. If in the place of (r, s) there is X or x, in the place of (p, q) there will be x.

c. If (r, s) is neither X nor x, then (p, q) will be 0.

d. Repeat b and c for final states also.

Step 3 : The combination of states where there is 0, they will be the states of the minimized machine.

Que 1.42. Minimize the automata given below

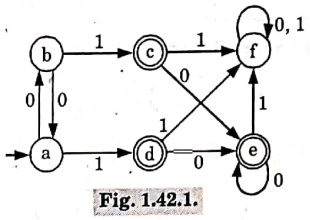


Fig. 1.42.1.

AKTU 2016-17, Marks 7.5

Answer

DFA minimization using equivalence theorem :

Transition table of DFA :

q	$\delta(q, 0)$	$\delta(q, 1)$
a	b	c
b	a	d
c	e	f
d	e	f
e	e	f
f	f	f

Let us apply the equivalence theorem to the given DFA :

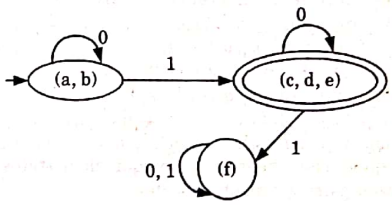
$P_0 = \{(c, d, e), (a, b, f)\}$

$P_1 = \{(c, d, e), (a, b), (f)\}$

$P_2 = \{(c, d, e), (a, b), (f)\}$

Hence, $P_1 = P_2$.

There are three states in the reduced DFA. The reduced DFA is as follows :



Transition table of DFA :

Q	δ(q, 0)	δ(q, 1)
(a, b)	(a, b)	(c, d, e)
(c, d, e)	(c, d, e)	(f)
(f)	(f)	(f)

Que 1.43. **Minimize the following automata :**

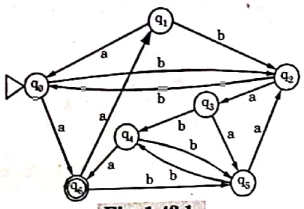


Fig. 1.43.1.

AKTU 2018-19, Marks 10

Answer

Transition table of DFA :

δ	a	b
→ q_0	q_6	q_2
q_1	q_0	q_2
q_2	q_3	q_0
q_3	q_5	q_4
q_4	q_6	q_5
q_5	q_2	q_4
*q_6	q_1	q_5

Let us apply equivalence theorem to the given DFA :

$P_0 = [(q_0, q_1, q_2, q_3, q_4, q_5), (q_6)]$

$P_1 = [(q_0), (q_1, q_2, q_3, q_4, q_5), (q_6)]$

$P_2 = [(q_0), (q_1), (q_2, q_3, q_4, q_5), (q_6)]$

$P_3 = [(q_0), (q_1), (q_2), (q_3, q_4, q_5,), (q_6)]$

$P_4 = [(q_0), (q_1), (q_2), (q_3), (q_4, q_5), (q_6)]$

$P_5 = [(q_0), (q_1) (q_2), (q_3), (q_4), (q_5), (q_6)]$

By using equivalence theorem the 5-equivalent states contain all the states without pairing with other states.

So the given DFA is already minimized. So minimized DFA is

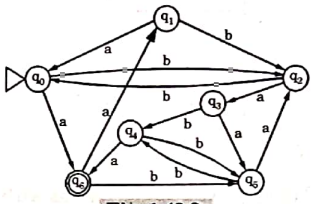


Fig. 1.43.2.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. **What do you understand by Deterministic Finite Automata (DFA) and how it is represented ?**

Ans. Refer Q. 1.5.

Q. 2. **Design a FA which accepts set of strings containing exactly four 1's in every string over Σ = {0, 1}.**

Ans. Refer Q. 1.9.

Q. 3. **What do you understand by Non-Deterministic Finite Automata (NFA/NDFA) ? How is it represented ?**

Ans. Refer Q. 1.15.

Q. 4. **Draw DFA of following over {0, 1} :**

i. **All strings with even number of 0's and even number of 1's.**

ii. **All strings of length at most 5.**

Ans. Refer Q. 1.13.

Q. 5. **Define NFA. What are various points of difference between NFA and DFA ?**

Ans. Refer Q. 1.21.

Q. 6. **Convert the following NFA {p, q, r, s}, {0, 1}, δ, p, {q, s} into DFA where δ is given by**

δ/Σ	0	1
→ p	q, s	q
*q	r	q, r
r	s	p
*s	ϕ	p

Ans. Refer Q. 1.25.

Q. 7. Describe Mealy and Moore machines with example. Convert the given Mealy machine as shown in Fig. 1 into Moore machine.

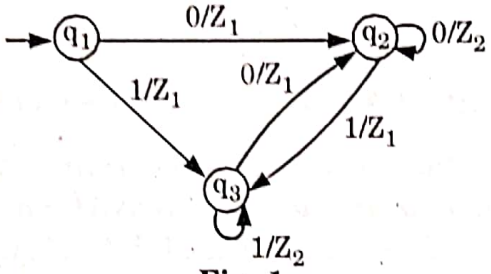


Fig. 1.

Ans. Refer Q. 1.36.

Q. 8. Minimize the automata given below

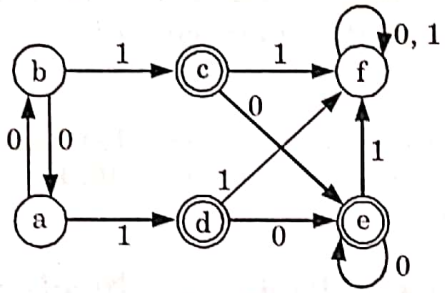


Fig. 2.

Ans. Refer Q. 1.42.