UNIT - I

Prepared By : SUKRATI AGRAWAL

automata theory is the study of mathematical objects called abstract M/c or automata and the computational problems that can be solved using them.

The automaton consist of states (represented by ○) and transitions (represented by arrows →) as the automaton sees a symbol of i/p, it makes a transition (or jump) to another state, according to its transition function (which takes the current state and the recent symbol as its i/p).

* automata theory is also closely related to formal language theory.

* an automaton is a finite representation of a formal language that may be an ∞ set.

* automata are often classified by the class of formal languages they are able to recognize.

* automata play a major role in

finite automata used in text processing CD, & h/w Design.

cellular automata ↓ biology.

- ↳ Theory of computatn
- ↳ Compiler Design
- ↳ artificial intelligence
- ↳ parsing (Latin word part of speech)
- ↳ formal verification → act of proving / disap. the correctness of intended algo (eg: cryptographic tools, Combinational ckt, Digital ckt)

Parsing

process of analysis string of symbol, either in natural language or in computer lang, acc to rules of formal (CFG) grammar

computational → computer linguistic

SUKRATI AGRAWAL

* an automaton is supposed to run on some given sequence of inputs in discrete time steps. an automata gets one i/p every time step that is picked up from set of symbols or letters, which is called as alphabet. at any time, the symbol so far fed to the automaton as i/p from a finite sequence of symbols which is called word.

* The automaton reads the symbol of the i/p word one after the another and transit from one state to state acc to transition funcn, until the word is read completely.

Once the i/p has been read, the A is said to have been stopped and the state at which A has stopped is called final state.

Depending on the final state, its said the A either accepts or rejects an i/p word.

Subset of states ⇒ accepting states

* [The set of all the words accepted by an automaton is called the language recognized by the automaton.]

* String

Er. Sukrati Agrawal. UNIT - I

Date
19th june 2013

Theory of computation is the branch that deals with whether and how efficiently problems can be solved on model of computation, using an algorithm. •

mathematical abstraction of computer
↓
Turing M

The fields divided into three major branches:

* automata theory : study of abstract M/c (abstract Mathematical M/c) and the computational prob that can be solved using these M/c.

* computability theory : deals primarily with the question of the extent to which a problem is solvable on a computer. (halting problem of turing M/c).

* computational complexity theory.
Considers not only whether a problem can be solved at all on computer, but also how efficiently the problem can be solved.

- Time complexity
- Space complexity

automata - Greek word meaning "Something is doing something by itself". [self- acting]

Date 19th june 2013

Introduction to Finite Automata

* finite automata are computing devices that accept/recognize regular languages. and are used to model operation of many system we find in practice.

* used in text processing, compiler Design & H/w Design.

* for every ~~raw~~ regular language a unique finite automaton can be constructed which can recognize the language (i.e. tell whether or not given string belongs to regular language)

Formal Definition :-

Finite automaton is represented formally by 5-tuple $(Q, \Sigma, \delta, q_0, F)$

1> $Q \Rightarrow$ is a finite set of states

2> $\Sigma \Rightarrow$ is a finite set of symbol, called the alphabet of the automaton.

3> $\delta \Rightarrow$ is the transition function that is

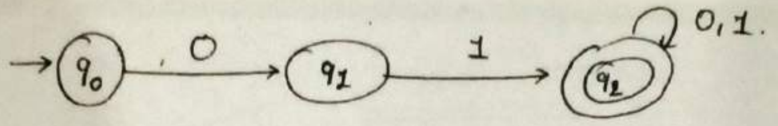
$\delta : Q \times \Sigma \rightarrow Q$ if its Deterministic FA.

$\delta : Q \times \Sigma \rightarrow 2^Q$ if its Non-Deterministic FA

4> q_0 is the initial state, that is, the state of the automaton before any i/p has been processed where $q_0 \in Q$

5> F is a set of states of Q (i.e. $F \subseteq Q$) called accept state.

#



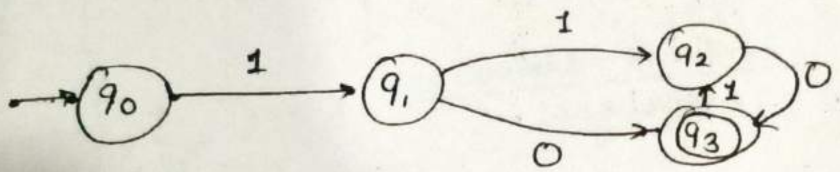
- Check whether the string 0100 accepted by the given DFA.

Soln

$$q_0 0100 \rightarrow q_1 100 \rightarrow q_2 00 \rightarrow q_2 0 \rightarrow q_2$$

the final state is q_2 & here at the end we are at q_2 hence the given string is accepted.

#



Check 1101010

$$q_0 1101010 \rightarrow q_1 101010 \rightarrow q_2 01010 \rightarrow q_3 1010$$

$$\rightarrow q_2 010 \rightarrow q_3 10 \rightarrow q_2 0 \rightarrow q_3$$ ← which is a final stat & hence string accep

Check 10101

$$q_0 10101 \rightarrow q_1 0101 \rightarrow q_3 101 \rightarrow q_2 01 \rightarrow q_3 1 \rightarrow q_2$$

at the end we reached at state q_2 which is not a final state & hence the

Formal languages

Problems/sets

Recursive RE.

Generator	Acceptor
Formal grammar	Automaton

Automaton – a moving mechanical device made in imitation of human being. (robot).

Type 0	Recursively enumerable languages	Turing M/c
Type 1	Content - sensitive languages	Linear bounded automata
Type 2	Content - free language	Non - Determ nistic PDA
Type 3	Regular language	finite state M/c

Chomsky Hierarchy

enumberal - that can be counted.

automaton - robot.

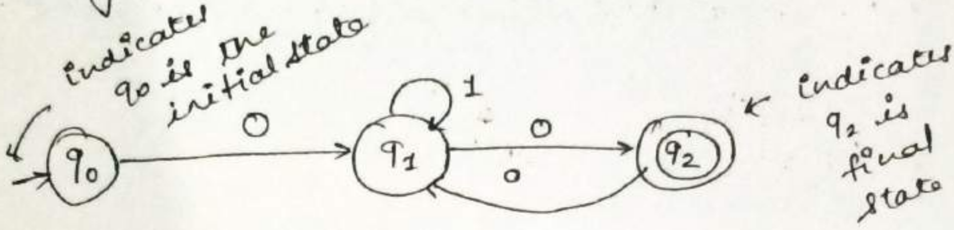
Classes of automata →
- Discrete
- continuous → analog data
- hybrid automata

Deterministic Finite Automata :→ (DFA)

In case of DFA from a state when we apply i/p then we have only one o/p state that means there is no choice for a given pair of i/p & state.

Thats why called DFA. Its more powerful than NDFA.

e.g.



state Transition diagram.

Transition table :→

State ↓ | i/p →	0	1
→ q_0	q_1	–
q_1	q_1	q_2
q_2	q_1	–

* For a given Transition only single o/p state appears hence its a deterministic FA.
* only single initial state is possible
* we can have more than one final / accepting state.

here 5 tuples are ⇒ $Q \Rightarrow \{q_0, q_1, q_2\}$

$\Sigma \Rightarrow \{0, 1\}$ 0, 1 are symbols here

$q_0 = q_0$ initial state $\delta \Rightarrow$ Transition function.

$F = \{q_2\}$ only single final state. $\delta(q_0, 0) = (q_1)$

$\delta(q_1, 0) = (q_2)$

$\delta(q_1, 1) = (q_1)$

$\delta(q_2, 0) = (q_1)$

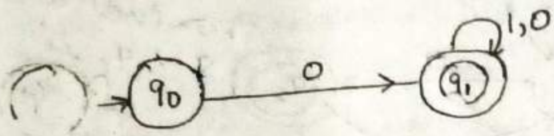
Question Pattern.

↳ String acceptance by DFA.

↳ Design of DFA.

List of question :→

① Design a DFA which begins with 0.



↳ NDFA to DFA Conversion

↳ Regular Expression to DFA

↳ DFA to Regular Expression (ARDEN'S Theorem)

→ Elimination of null production or ε move.

↳ To prove grammar is not regular (By pumping lemma)

↳ Minimization of finite automata (By My-Hill Nerode Theo)

↳ Mealy M/C

↳ Moore M/C

↳ Mealy to moore M/C

→ Moore to Mealy M/C

Non Deterministic Finite Automata

in this finite automata when we apply i/p from state then we can have more than one o/p state. Thats why called NDFA.

State transition Diagram:-

State transitn table:-

	0	1
q_0	q_1	–
q_1	–	q_1, q_2
q_2	q_2	q_2

contains multiple entry hence its a NDFA.

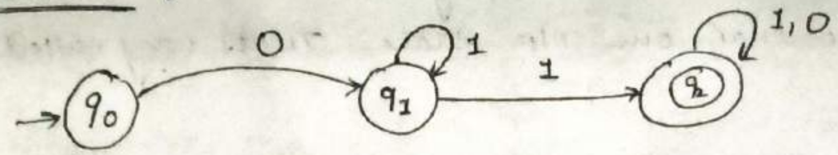
Transition function:→.

$$\delta(q_0, 0) = q_1$$
$$\delta(q_1, 1) = q_1, q_2$$
$$\delta(q_2, 0) = q_2$$
$$\delta(q_2, 1) = q_2$$

final state | accepting state:→ $\{q_2\}$

(NDFA) NonDeterministic FA to Deterministic FA (DFA).

Question is :



This is NDFA b/c when we apply i/p 1 from state q_1 then we have 2 ways to move further. either we can remain of q_1 itself or can move to q_2. ∴ NDFA.

$$\delta(q_1, 1) = \begin{cases} \to q_1 \\ \to q_2 \end{cases}$$

Now converting this NDFA to DFA.
b/c DFA is more powerful than NDFA.

draw the table

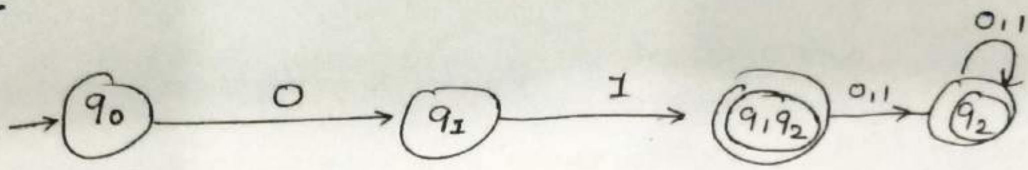
- starting from initial state here $I_s = q_0$

		0	1	
	$\to q_0$	q_1	—	
Now q_1 is new state	q_1	—	$q_1 q_2$	← this not 2 state is a new state named with these two.
Now $q_1 q_2$ are not 2 new state its a single new state	$q_1 q_2$	—	q_2	
	q_2	q_2	q_2	

$q_1 q_2 = (q_1 \cup q_2)$

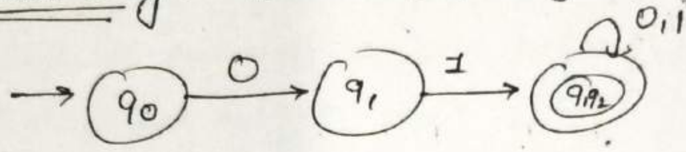
Now as $q_1 q_2$ & q_2 both has same o/p state that means these are not 2 diff state ∴ we can combine these 2 states directly.

soln is



Now q_1q_2 is also a final state b/c it consist of q_2 which is our final state in the given NDFA.

equivalent diag. (Minimized diagram)



Theorem : if L language accepted by a NDFA, then a DFA exist accepting L.

The proof of this theorem is constructive one. Given NFA we construct an equivalent DFA where the set of final states is the set of subsets that contains at least one final state of the starting NFA and the transition function is defined by applying the δ of the NFA to each state in a given subset and taking the union of the resulting states.

FOR EVERY NFA THERE Exists DFA

* when the NFA (N) has n states the corresponding DFA (D) has 2^n states. However we needn't construct ~~D~~ for all these 2^n states, but only for those states reachable from q_0 (starting state). so we start the construction of D for q_0, we continue by considering only states appear earlier under i/p columns and constructing D for such that, we halt when no more new state appear under the i/p columns.

Regular Expression

*. used in lexical anal phase of CD.

* is a compact description of a set of string.

* DFA is an abstract m/c that solves pattern match problem for regular expression (regxp)

* DFA & regxp have limitation.

* any regular language may be specified by regxp

Programmer:

- Regular expressions are powerful pattern matching to
- Implement regxp with finite state m/c.

Variation:

- Yes (accept) and No (reject) states sometimes drawn differently.
- Terminology: DFA, FSM, FSA are the same.
- DFA's can have o/p, specified on the arcs or in the states
  - These may not have explicit yes/no states.

Limitation of DFA:

- No DFA can recognize the language of all bit strings with an equal no of 0's & 1's. *

which language can't be described by any RE.

- previous one *
- Decimal strings that represent prime numbers.
- Genomic string that are watson - crick complemented palindrome.

Context free grammar
eg:- JAVA

extended REs.

late
repked
q_2
↑
ration

Definition of RE

a regular expression re over Σ is represents a set of strings (possibly finite) denoted by $L(r)$ is defined as follows –

(i) ϵ, ϕ, a are valid re where $a \in \Sigma$.

(ii) If R_1, R_2 are valid re then so are

* $R_1 + R_2$ representing $L(R_1) \cup L(R_2)$ $(R1|R2)$
* $R_1 \cdot R_2$ representing concatenation. $(R1 \cdot R2)$
* R_1^*

  representing $\epsilon \cup L(R_1) \cup L(R_1^2) \cup L(R_2^2) \ldots$
* (R_1) is regular (to limit the scope for use of other operators)

The class of language that can be represented using regular expression is called Regular language.

Theorem 4.1 → A Language L is regular iff there is a DFA accepting exactly the string in L.

$a^+ = \{a, aa, aaa, aaaa \ldots\ldots\}$

$a^* = \{\wedge, a, aa, aaa, \ldots\ldots\}$

$(aa)^* = \{aa, aaaa, aaaaaa, \ldots\ldots\}$

a or b = a|b = a+b → parallel ckt

a^* = self loop

a and b = a·b = ab → series ckt

Identities for Regular Expressions

I_1: $\phi + R = R$ $\{\}$

I_2: $\phi R = R\phi = \phi$

I_3: $\Lambda R = R\Lambda = R$ $\{\Lambda\}$

I_4: $\Lambda^* = \Lambda$ and $\phi^* = \Lambda$

I_5: $R + R = R$ } idempotent law

I_6: $R^* R^* = R^*$

I_7: $RR^* = R^*R$ commutative law

I_8: $(R^*)^* = R^*$

I_9: $\Lambda + RR^* = R^* = \Lambda + R^*R$

I_{10}: $(PQ)^*P = P(QP)^*$ commutative law

I_{11}: $(P+Q)^* = (P^*Q^*)^* = (P^*+Q^*)^*$

I_{12}: $(P+Q)R = PR + QR$ & $R(P+Q) = RP + RQ$ } Distributive law.

I_{13}: $L^+ = LL^* = L^*L$

I_{14}: $L^* = L^+ + \varepsilon$

(a) $L_1 \Rightarrow$ the set of all strings of 0's and 1's ending in 00.

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

(b) $L_2 \Rightarrow$ the set of all strings of 0's and 1's beginn with 0 & ending with 1.

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

- give a re for representing the set L of strings in which every 0 is immediately followed by atleast two 1's.

Soln: The given questn has 2 possibilities

(1) the string doesn't contain any 0

$\therefore$ string of 1 only.

(2) if string contain 0 then it must followed by 2 consecutive 1's.

i.e. 011

$\therefore$ soln is $(011+1)^* = (1+011)^*$

Prove that:

$$R = \Lambda + 1^*(011)^*(1^*(011)^*)^*$$

equivalent to

$$(1+011)^*$$

Soln By property

$$\Lambda + PP^* = P^*$$

here $P = 1^*(011)^*$

$$R = \Lambda + \underbrace{1^*(011)^*}_{P}\ \underbrace{(1^*(011)^*)^*}_{P^*}$$

$$= \Lambda + PP^*$$

$$= P^* = (\underbrace{1^*(011)^*}_{P})^*$$

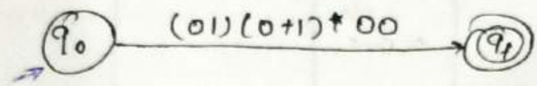
Now applying $(P+Q)^* = (P^*Q^*)^*$

$$(1+011)^* = (\underbrace{1^*}_{P}\ \underbrace{(011)^*}_{Q})^*$$ Hence Proved.

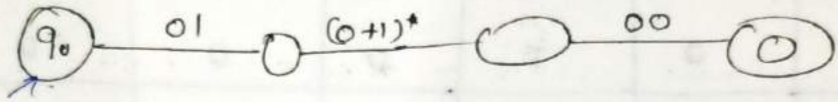
Designing of DFA

Q. Design a DFA which begins with 01 and ends $\bar{c}$ 00.

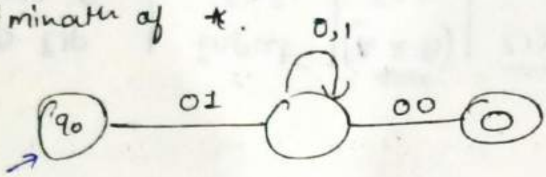
given:- $(01)(0+1)^*00$



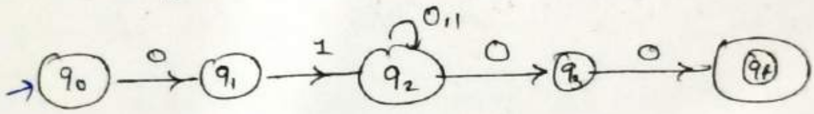
Eliminate concatination



Elimination of *.

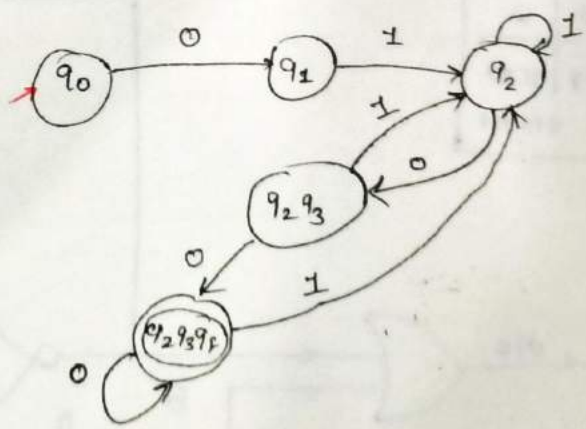


Elimination of concatination.



Now its NDFA ∴ converting it into DFA.

state	i/p 0	1
q_0	q_1	–
q_1	–	q_2
q_2	q_2q_3	q_2
q_2q_3	$q_2q_3q_f$	q_2
$q_2q_3q_f$	$q_2q_3q_f$	q_2



Date
22nd june, 2013
Saturday.

TRANSITION SYSTEM CONTAINING ∧-MO

The transition system can be generalized by permitting ∧-transi or ∧-move which are associated with a null symbol ∧. These transitn can occur when no i/p is applied. But its possible to convert a transitn system with ∧-move into an equivalent transition system without ∧-move.

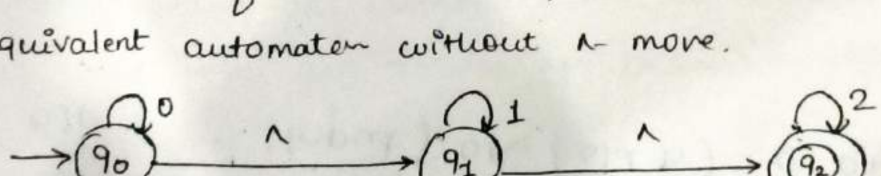
Procedure:→

Suppose we want to replace a ∧-move from Vertex $V_1 - V_2$. Then we proceed as follows:-

* Find all the edges starting from V_2.
* Duplicate all these edges starting from V_1, without changing the edge labels.
* if V_1 is initial state, make V_2 also as initial state.
* if V_2 is final state, make V_1 also final state.

Question:

Consider a finite automaton, with ∧-move. Obtain an equivalent automaton without ∧-move.

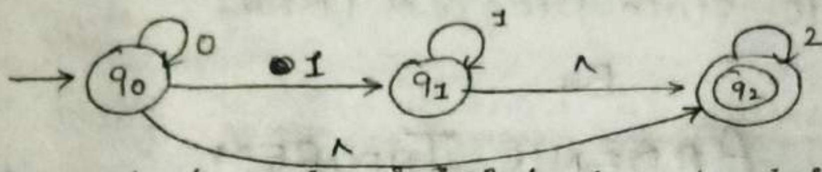


Now $q_0 = V_1$ & $q_1 = V_2$

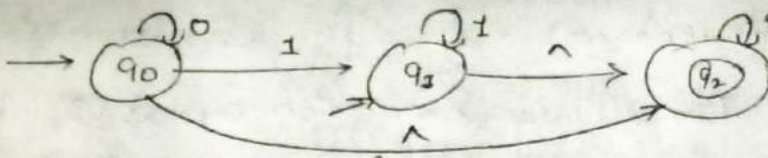
∴ applying all these previous four rules.

when we apply 1 on q_1, then we move to q_0

• copy edge of V_2 on V_1.



• Now V_1 (q_0) is initial ∴ V_2 initial.



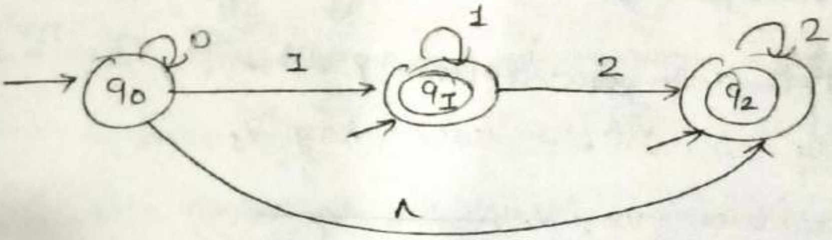
• Now V_2 is not final here. ∴ No change in diag.

★ Now we have to eliminate ^ move from q_1 to q_2

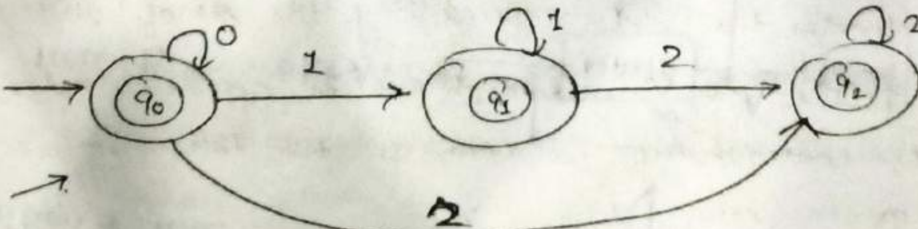
∴ we copy all the edges of q_2 on q_1

& then we make q_2 initial state b/c q_1 is I.S.

& finally we make q_1 as final state b/c q_2 is final state



Similarly when we eliminate ^- move b/w q_0 & q_2 then we will get following transition diag.



ans →

Construction of Regular Expression Corresponding to state Diagram (DFA) By

ARDEN'S THEOREM

Notes By
Sukrati Agrawal
Asst. Prof.
CSE.

Theorem 5.1 (Arden's theorem)

Let P and Q be two regular expression over Σ. If P doesn't contain Λ, then the following equation in R, namely,

$$R = Q + RP$$

has a unique solution (i.e. one and only one) given by

$$R = QP^*$$

The following assumptions are made regarding the Tx system.

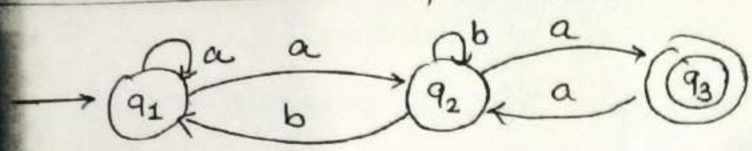
* The transition graph doesn't have Λ-move.
* it has only one initial state, say V_1.
* its vertices are $V_1, V_2 \ldots\ldots V_n$.
* V_i the r.e represents the set of string accepted by the system even though V_i is a final state.
* α_{ij} denotes the r.e. representing the set of labels of edges from V_i to V_j. when there is no such edge $\alpha_{ij} \neq \phi$. Consequently, we can get the following set of equatn in $V_1 \ldots\ldots V_n$.

$$V_1 = V_1\alpha_{11} + V_2\alpha_{21} + \ldots\ldots V_n\alpha_{n1} + \Lambda$$
$$V_2 = V_1\alpha_{12} + V_2\alpha_{22} + \ldots\ldots V_n\alpha_{n2}$$
$$\vdots$$
$$V_n = V_1\alpha_{1n} + V_2\alpha_{2n} + \ldots\ldots V_n\alpha_{nn}$$

By Repeatedly applying substitution and Theorem 5.1 we can express V_i in terms of α_{ij}'s.

* for getting the set of string recognized by the transit system, we have to take union of all final states

Let us take an example.



we have to write equation in the form of states and labels that states have.

when we write the eq^n for a state then we have to know that which are the incoming edges on that state.

suppose $\boxed{q_1}$ Now incoming edges on q_1 state are only (2)

i.e. (1) when we apply a from q_1 itself
(2) when we apply b from q_2 state
} we reach at stat $\boxed{q_1}$

$\therefore$ eq^n becomes

$$q_1 = q_1 a + q_2 b$$

Now here q_1 is initial state $\therefore$ we add Λ in that so eq^n becomes

$$q_1 = q_1 a + q_2 b + \Lambda$$

Similarly for q_2 eqn is

$$q_2 = q_1 a + q_2 b + q_3 b$$

no need to add $\wedge$ b/c q_2 & q_3 are not initial states.

Similarly,

$$q_3 = q_2 a$$

Now as q_3 is the final state then its clear that answer should be in the form of q_3 only & it contain only ε not any state.

Now we have to apply substitution and Arden's theorem in order to find out a soln.

We have

$$q_1 = q_1 a + q_2 b + \wedge \quad \text{———} \quad (1)$$

$$q_2 = q_1 a + q_2 b + q_3 a \quad \text{———} \quad (2)$$

$$q_3 = q_2 a \quad \text{———} \quad (3)$$

Putting value of q_3 in (2)

$$q_2 = q_1 a + q_2 b + \underline{q_2 a} a$$

$$q_2 = q_1 a + q_2 (b + aa)$$

Now ARDEN'S theorem

$R = Q + RP$ then soln

is $R = QP^*$

here $R = q_2$ & $P = (b + aa)$

$\therefore$ soln becomes:

$$q_2 = q_1 a\,(b+aa)^* \quad \text{——— (4)}$$

Now put this value of q_2 in eqn (1)

$$q_1 = q_1 a + \underline{q_1 a\,(b+aa)^*} b + \wedge$$

$$q_1 = q_1\,[a + a\,(b+aa)^* b] + \wedge$$

$$\underset{R}{q_1} = \underset{Q}{\wedge} + \underset{R}{q_1}\,\underset{P}{[a + a\,(b+aa)^* b]}$$

em

applying arden's theo.

$$R = Q + RP \Rightarrow R = QP^*$$

here $R = q_1 \quad P = (a + a(b+aa)^* b)$

$\therefore$ soln becomes

$$q_1 = \wedge . (a + a\,(b+aa)^* b)^*$$

By property of Regular Expression $\Rightarrow \wedge . R = R$.

$$\therefore q_1 = (a + a(b+aa)^* b)^*$$

put this value of q_1 in q_2 i.e. eqn (4)

$$q_2 = (a + a(b+aa)^* b)^* a\,(b+aa)^*$$

Now as we know ans. is in the form of final state & here final state is q_3 $\therefore$ putting value of q_2 in eqn (3) we get

$$\boxed{q_3 = (a + a\,(b+aa)^* b)^* a\,(b+aa)^* a}$$

This is the regular expression which is equivalent to given transition system.

CONSTRUCTION OF FINITE AUTOMATA EQUIVALENT TO A REGULAR EXPRESSION

Method is called <u>Subset Method</u> which involves 2 steps.

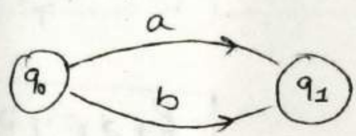
<u>Step 1</u>: Construct a transition graph equivalent to the given regular expression using $\wedge$- moves using Theorem.

<u>case 1</u>: $R = P + Q$ (solve using || ckt)

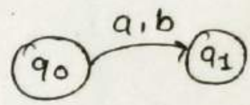
then apply 'OR' operatn that means we are getting 2 ways to moves further [|| operⁿ] <u>|| ckt</u> parallel ckt

eq: $a+b$

$\therefore$

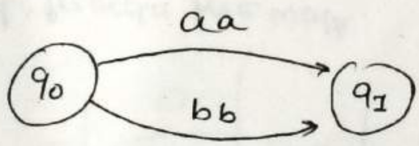


$\equiv$ equivalent to



eq: $\underset{P}{\underline{aa}} + \underset{Q}{\underline{bb}}$

$\Rightarrow$



<u>case 2</u>: $R = PQ$ (solve using series ckt)

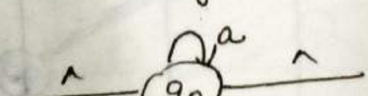
apply series ckt

eq: $a.b$

$\Rightarrow$

<u>case 3</u>: $R = P^*$ (then apply self loop)

eq: a^*



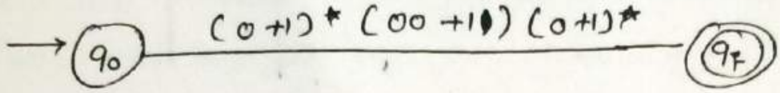
apply self loop with adding $\wedge$ product on both direction of addⁿ with 2 new states with $\wedge$- move

Step 2: Now construct the equivalent DFA for given transition system

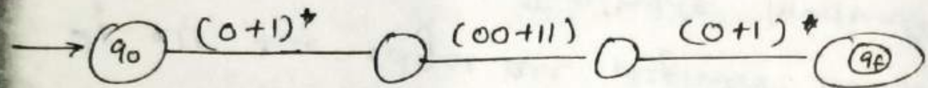
- having no ^ move
- reduce the no. of states if possible.

Let take an example.

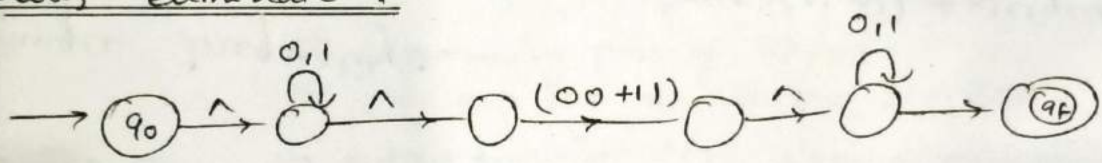
$$(0+1)^*\,(00+11)\,(0+1)^*$$



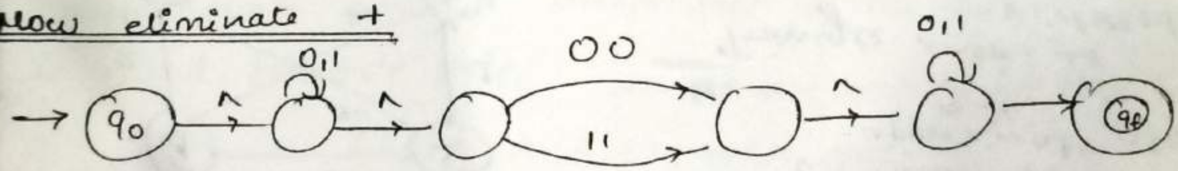
Step 1: Elimination of concatenation i.e. $R = PQ$.



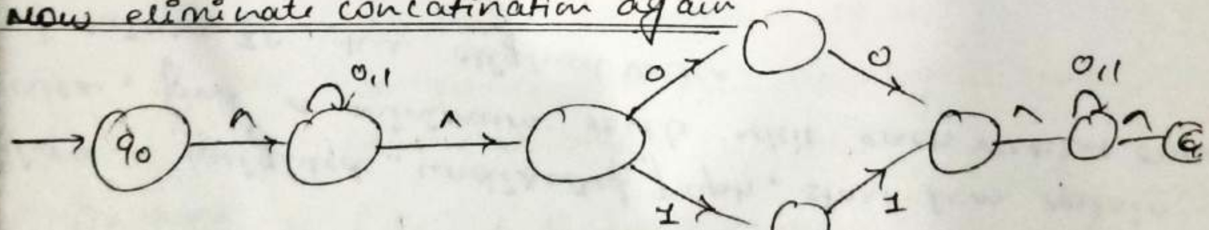
Now, eliminate *



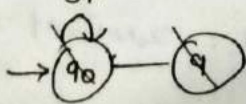
Now eliminate +

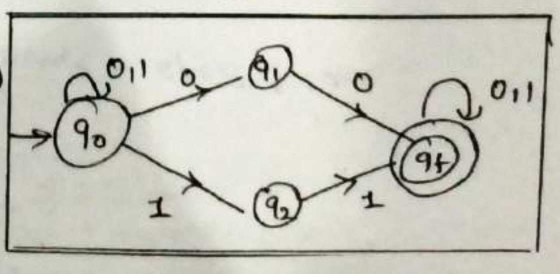


Now eliminate concatination again



∴ finally,





Mealy and Moore Models

i.e. Finite Automata With Outputs

Notes By: Sukrati Agrawal (Asst Prof CSE)

The finite automata which we studied earlier have binary output i.e. either they accept the string or not. This acceptability was decided on the basis of reachability of the final state by the initial state.

Now, we remove this restriction and consider the model where the output can be choosen from some other alphabet

* The value of output function $z(t)$ depends on present state as well as the current/present i/p $x(t)$ is called Mealy M/C.

present state — present i/p at time t

$$z(t) = \lambda(q(t), x(t))$$

edge having pair of i/p / o/p.
can get the o/p in b/w the trans^n.

* The value of output function $z(t)$ depends only on present state and is independent of the current i/p is called Moore M/C.

o/p is func^n of i/p only state

$$z(t) = \lambda(q(t)) \rightarrow \text{i/p state}$$

state having o/p attached to it.
can get the o/p after reaching at the next state only.

* Six tuple m/c $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where
  - # Q is a set of finite state.
  - # Σ is the i/p alphabet
  - # Δ is the o/p alphabet
  - # δ is the transition func^n $\Sigma \times Q$ into Q
  - # λ is the o/p func^n [mapping depends on m/c]
  - # q_0 is initial state.

Mealy Machine

$$\boxed{\lambda \Rightarrow \Sigma \times Q \text{ into } \Delta}$$

Transition table :→

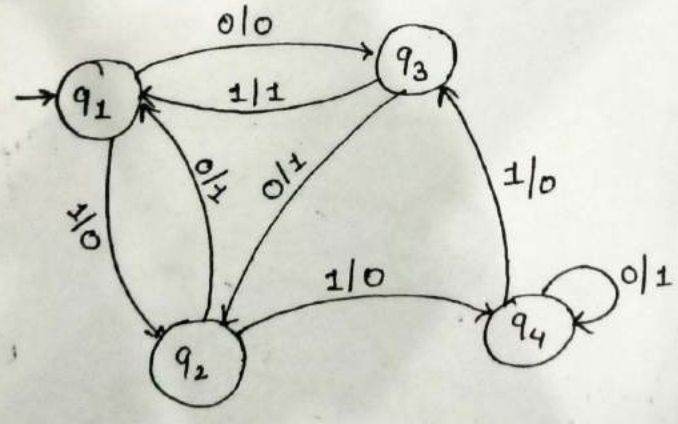
Present state	Next state			
	i/p		i/p	
	a = 0		a = 1	
	state	o/p	state	o/p
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

for i/p string 0011 o/p string is 0100

$q_1\,0011 \rightarrow q_3\,011 \rightarrow q_2\,11 \rightarrow q_4\,1 \rightarrow q_3$

o/p 0 1 0 0 $\therefore$ o/p = 0100

state transition diagram :→



Moore Machine

* **Transition table :→**

λ maps → Q into Δ

present state	Next state δ		Output
	a=0	a=1	λ
→ 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

* for the i/p string 0111, the o/p string is 00010

the o/p is of (n+1) string b/c if we are not giving any i/p to the initial state then also we are getting o/p.
as the o/p is corresponds to the state not to the i/p.

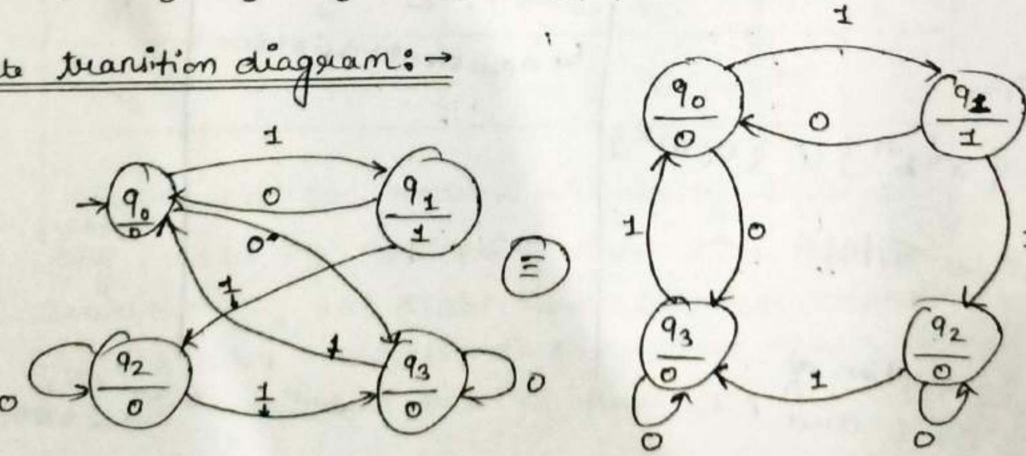
$q_0\,0111 \rightarrow q_3\,111 \rightarrow q_0\,11 \rightarrow q_1\,1 \rightarrow q_2$

→ at $q_0 = 0$ 0   0   1   0

$q_0 \rightarrow q_3 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2$

now o/p: 0 0 0 1 0

* **State transition diagram :→**



Procedure for Transforming A Moore M/c into A Mealy M

PREPAREDBY: SUKRATI AGRAWAL

Transition table of Moore M/c :→.

Present state	Next state		output
	a=0	a=1	
→ 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

∴ Transition table of Mealy M/c :→ b/c λ for Mealy M/c Σ×Q into Δ.

present state	Next state				output
	a=0		a=1		
	state	o/p	state	o/p	
→ 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_0	0	

Note:→ we can reduce the no. of states in any model by considering states with identical transition. If two states have identical transition (i.e. the rows corresponding to two states are identical then we can delete one of them.

PTO

Procedure for Transforming Mealy M/C into Moore M/C

- we split q_i into several different states, the no. of such state being equal to the no. of different o/p associated with q_i.

for eg: suppose q_1 is associated with single o/p through out the table i.e. $(q_1 \rightarrow 1)$ ∴ we don't split it.

But if a state like q_2 is associated with 2 o/p ∴ we split q_2 into 2 parts $\left[q_2 \rightarrow \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \Rightarrow q_{20} \rightarrow 0 \ \& \ q_{21} \rightarrow 1 \right]$

- Given: Mealy Machine :

present state	Next state				Test for Moore M/C
	i/p				
	a=0		a=1		
	state	o/p prob	state	o/p prob	
$\rightarrow q_1$	q_3 $\square$	0	q_2 *	0	$q_1 \Rightarrow 1$
q_2	q_1	1	q_4	0	$q_2 \rightarrow \begin{smallmatrix} 0 \\ 1 \end{smallmatrix}$
q_3	q_2 *	1	q_1	1	$q_3 \rightarrow 0$
q_4	q_4	1	q_3 $\square$	0	$q_4 \rightarrow \begin{smallmatrix} 0 \\ 1 \end{smallmatrix}$

Transition table for Mealy M/c

- Solution: ∴ Transition table for Moore M/c is.

Present State	i/p a=0	i/p a=1	output
$\rightarrow q_1$	q_3	q_{20}	1 * causes prob
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_2	1

Now, observe the thing carefully that in given Mealy M/c the q_1 state is associated with or generates o/p 0.

But here Moore m/c the initial state q_1 is associated with output 1. This means that with i/p $\wedge$ we get an o/p of 1, if the m/c starts at state q_1, thus this moore m/c accepts a zero length sequence (null sequence) which is not accepted by Mealy M/c

To, overcome this situation, either we must neglect the response of a Moore M/c to i/p $\wedge$, or we must add a new starting state q_0, whose state transitions are identical with those of q_1, but whose o/p is 0.

finally, the converted Moore Machine is:→

prob in Mealy → q_1 (initial state) gives o/p → 0
∴ in Moore we create new initial state with o/p → 0

present state	Next state		output
	a=0	a=1	
→ q_0	q_3	q_{20}	0
q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

Solutn provided to the Problem

from the foregoing procedure it's clear that if we have m o/p, n state MM the corresponding m o/p Moore M/c has no more than mn+1 states.