Finite state machine ~~is~~ with output:

The finite Automat have binary output. i.e., They accept the string or donot accept the string. This acceptability is decided on the basis of reachability of the final state from the initial state.

This riestriction can be removed by considering the model where the output can be chosen from some other alphabet. There are two distinct approaches

(I) moore machine (II) mealy machine

moore machine:

Moore machine is a six tuple machine that can be represented as

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

where Q is a finite set of state

Σ is the set of input alphabets

Δ is the set of output alphabets.

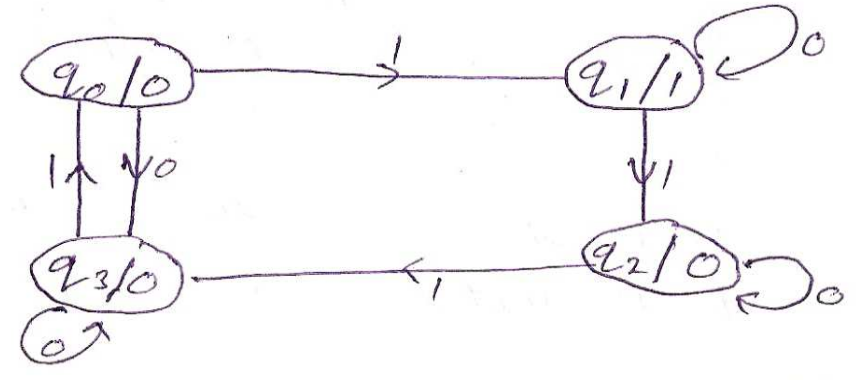
δ is the transition function which maps $Q \times \Sigma$ to Q

λ is the output function which maps Q to Δ

q_0 is the initial state

in Moore machine, the output is associated with the ~~transition~~ current states.

ex:



in this machine

$$m = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

where

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

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

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

$q_0 = \{q_0\}$

δ	0	1
q_0	q_3	q_1
q_1	q_1	q_2
q_2	q_2	q_3
q_3	q_3	q_0

λ	output
q_0	0
q_1	1
q_2	0
q_3	0

or merge both table

	Next state 0	Next state 1	output
q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

check the output of string 0111

$$q_0 \xrightarrow{0} q_3 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2$$

O/P → (0) (0) (0) (1) (0)

so the output is 00010

<u>mealy machine</u>:

mealy machine is a six tuple machine that can be represented as

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

where

Q is set of states.

Σ is set of input alphabets

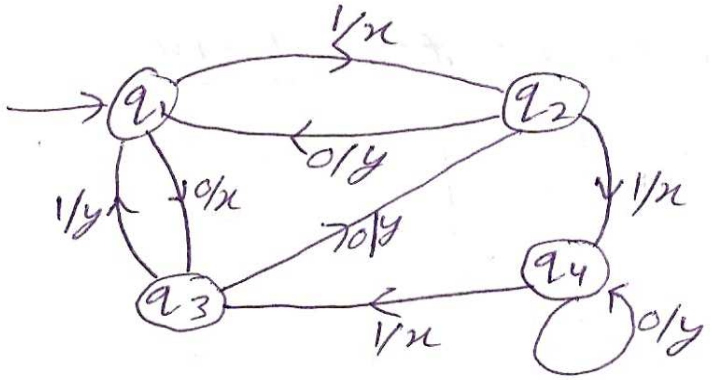
Δ is set of output alphabets

δ is the transition function which maps $Q \times \Sigma$ to Q

λ is the output function which maps $Q \times \Sigma$ to Δ

q_0 is initial state.

ex:



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

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

$\Delta = \{x, y\}$

$q_0 = \{q_1\}$

δ	0	1
q_1	q_3	q_2
q_2	q_1	q_4
q_3	q_2	q_1
q_4	q_4	q_3

λ	0	1
q_1	x	x
q_2	y	x
q_3	y	y
q_4	y	x

or

state	input 0	output	input 1	output
q_1	q_3	x	q_2	x
q_2	q_1	y	q_4	x
q_3	q_2	y	q_1	y
q_4	q_4	y	q_3	x

Conversion of moore m/c to mealy m/c

Q.

State	Next state 0	Next state 1	output
q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

convert this moore m/c to mealy m/c

Solution :

Let moore m/c $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

then a equivalent mealy m/c $M' = (Q', \Sigma', \Delta', \delta', \lambda', q_0')$

where $Q' = Q$

$\Sigma' = \Sigma$

$\Delta' = \Delta$

$\delta' = \delta'$

$q_0' = q_0$

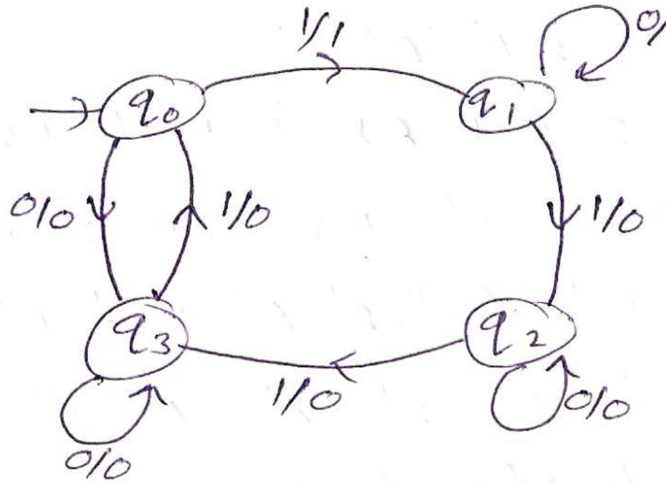
in λ' if $\delta(q, a) = P$ and output of P is x

then transition of a from q to P is associated with output x. here $q, P \in Q'$, $a \in \Sigma'$, $x \in \Delta'$

so ~~Note~~ mealy m/c output & transition table is

state	Next state 0	output	Next state 1	output
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

Transition Diagram



* Conversion of mealy m/c to moore m/c :

To construction of moore m/c which is equivalent to mealy m/c is as follows -

* firstly we convert mealy m/c to modified mealy m/c. in this step if in mealy m/c $\delta(q_1, a) = p$ with output x and $\delta(q_2, b) = p$ with output y then we splite into two different states { any state p splite into the states equal to the different outputs assosiated with p }

here $a, b \in \Sigma$ (may be same), $q_1, q_2 \in Q$ (may be same) $p \in Q$, $x, y \in \Delta$ $\{x \neq y\}$

then in modified mealy m/c transection are as $\delta^*(q_1, a) = p_a$, $\delta(q_2, b) = p_b$

* if initial state is splite then make anyone of them as initial state.

Now Let, modified mealy m/c (after splite of states) is

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

$\{ Q' \subseteq Q \times \Delta,\ \Sigma' = \Sigma,\ \Delta' = \Delta \}$

where $(Q, \Sigma, \Delta, q_0, \delta, \lambda)$ is mealy m/c

* Now convert modified mealy m/c to moore m/c

Let moore m/c is $m'' = (Q'', \Sigma'', \Delta'', \delta'', \lambda'', q_0'')$

where $Q'' = Q'$, $\Sigma'' = \Sigma'$, $\Delta'' = \Delta'$, $\delta'' = \delta'$, $q_0'' = q_0'$

in λ'', if in modified mealy m/c

$\delta(q, a) = p$ and output is x then

in moore m/c output x is associated with state p.

Q. convert mealy m/c to equivalent moore m/c

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

First we convert this mealy m/c to modified mealy m/c.

in this m/c state q_2, q_4, are splite because

$q_1 \xrightarrow{1} q_2$, with output x }

and $q_3 \xrightarrow{0} \underbrace{q_2}_{\text{Same}}$, with output y } o/p different so state q_2 splite into two state q_{2x}, q_{2y}

$$q_2 \xrightarrow{1} q_4,\ o/p = x$$
$$q_4 \xrightarrow{0} \underbrace{q_4}_{\text{same}},\ o/p = y$$
} different

So state q_4 splite into two state q_{4x}, q_{4y}

Now modified mealy m/c is

Present state	Next State			
	input a=0		input a=1	
	State	o/p	State	o/p
$\rightarrow q_1$	q_3	x	q_{2x}	x
q_{2x}	q_1	y	q_{4x}	x
q_{2y}	q_1	y	q_{4x}	x
q_3	q_{2y}	y	q_1	y
q_{4x}	q_{4y}	y	q_3	x
q_{4y}	q_{4y}	y	q_3	x

Now convert in moore m/c.
if in mealy m/c table

$q_i \xrightarrow{\Sigma} q_j$ with K output then in moore m/c state q_j associate with K, transection same as modified mealy m/c.

So moore m/c is

Present state	Next State		o/p
	a=0	b=1	
$\rightarrow q_1$	q_3	q_{2x}	y
q_{2x}	q_1	q_{4x}	x
q_{2y}	q_1	q_{4x}	y
q_3	q_{2y}	q_1	x
q_{4x}	q_{4y}	q_3	x
q_{4y}	q_{4y}	q_3	y

Grammars:

A Grammar for the english language tells us weather a Particular sentence is well-formed or not. A typical rule of ~~a~~ english grammar is "a sentence can consist of a noun ~~Phrase~~ Phrase followed by verb Phrase.

we write this as

<sentence> $\longrightarrow$ <noun Phrase> <verb phrase>

and <noun phrase> $\longrightarrow$ <article> <noun>

<verb Phrase> $\longrightarrow$ <verb>

if we associate a with article, boy with noun and runs with verb then by grammar ~~that~~ a sentence ~~is~~ "a boy runs" is Properly formed.

if we were to give a complete grammar, then every Proper sentence could be explained this way. the generalization of these ideas leads us to formal grammars.

A Grammar G is represented by four touple

$$G = (V, T, S, P)$$

wheere V is finite set of objects called variables.

T is finite set of objects called terminal symbols.

$S \in V$ is called start variable

P is finite set of rules / Productions.

here $V \cap T = \phi$

The Production rules are the heart of a grammar, they specify How the grammar transforms one string into another and through this they define a language associated with the grammar.

Let assume one Production rule is

$$x \rightarrow y$$

where $x \in (V \cup T)^+$, $y \in (V \cup T)^*$

Let a string $w = uxv$, then by appling the Production we get "uyv" (Let $= z$)

$$z = uyv$$

This is writen as $w \rightarrow z$

we say z is derived from w.

* a Production can be used whenever it is applicable

if $w_1 \rightarrow w_2 \rightarrow w_3 \cdots \rightarrow w_n$

* means w_n derived by w_1 and write as

$$w_1 \xrightarrow{*} w_n.$$

* by applying the Production rules in a different order, a given grammar can normally generate many strings.

" The set of all such terminal strings is the language defined or generated by grammar.

Let $G = (V, T, S, P)$ be a grammar. then the set

$$L(G) = \{ w \in T^* : S \overset{*}{\Rightarrow} w \}$$

is the language generated by G.

if $w \in L(G)$ then the sequence

$$S \rightarrow w_1 \rightarrow w_2 \rightarrow w_n \rightarrow w$$

is a derivation of sentence w. The strings $S, w_1, w_2, \ldots w_n$ which contain variables as well as terminals are called "sentential form" of the derivation.

ex:

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

with P Given by $S \rightarrow aSb$,
$S \rightarrow \varepsilon$.

then $S \rightarrow aSb \rightarrow aaSbb \rightarrow aabb$

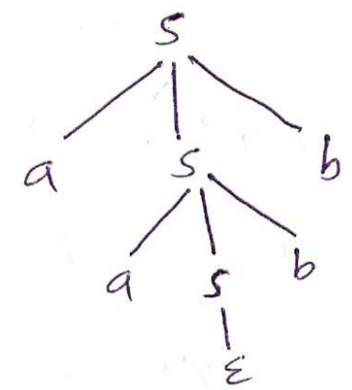
so we can write $S \overset{*}{\Rightarrow} aabb$

The string aabb is a sentence in the language generated by G. while aaSbb is a sentential form.

The Language generated by G is $\{a^n b^n : n \geq 0\}$

Note: Two different grammar G_1 and G_2 are equivalent if they generate the same language. $L(G_1) = L(G_2)$

Parsetree / syntax tree:

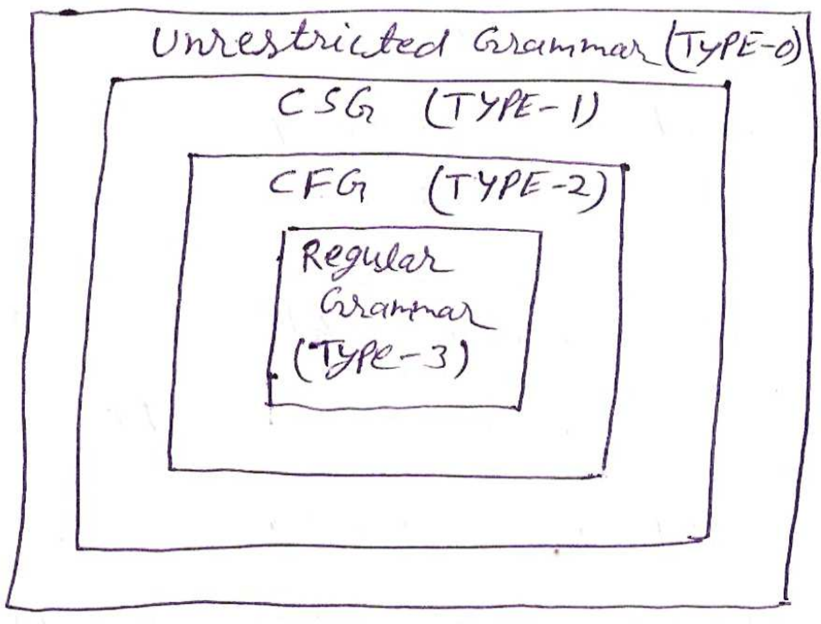


read leaf node from left to right

a a ε b b

⇒ a a b b

CHOMSKY HIERARCHY of GRAMMAR:



(i) Regular grammar (TYPE 3):

it is set generation technique, same capability as regular expression. it is always be left linear or right linear grammar

left linear grammar : $A \to Bw / w$
where w is string of Terminals

Right linear grammar : $A \to wB / w$

ex. $S \to 0A / 10A$
$A \to 01$

* it is equivalent to a DFA.

content free grammar (Type-2) :

in CFG, Productions are as

$$A \to \alpha$$

where $A \in V$, α is a string of Variables & Terminal

* CFG is equivalent to a Push down Automata.

ex. $S \to aAb$
$A \to ab / \varepsilon$

content sensitive Grammar (CSG) : (Type: 1)

in CSG Productions are as

$$\alpha \to \beta$$

where α, β is the string of Terminal & Variable
and $|\alpha| \le |\beta|$

* CSG is equivalent to a linear bounded Automata.

ex: $01A \to 001A$

Unrestricted grammar (Type 0) : in this grammar Production are as $\alpha \rightarrow \beta$ where α, β are the string of terminals & variables.

ex: $01A \rightarrow 001A$

$01A \rightarrow 1A$

* unrestricted grammar is equivalent to turing m/c