نظرية الحاسبات

المحاضرة الثانية عشر

الزمن: ساعة

Difference between Moore and Mealy Machine

	Moore Machine		Mealy Machine
1.	Its output depends on the current state	1.	Its output depends on the current state and input.
2.	If input string is of length n, then the output string is of length $n + 1$.	2.	If input string is of length n then the output string is also of length n.
3.	At the different input on the same state, its output is same.	3.	At the different input on the same state, its output is also different.
4.	It prints characters when in state.	4.	It prints characters when traversing an arc.

Equivalence between Moore and Mealy Machines

In case of finite automata, the two machines are equivalent if they accept the same language. owever, in case of finite automata with output, the two machines are equivalent if they produce same output string when the input string is same. Moore machine can never be directly equivalent to Mealy machine because the length of the output string from a Moore machine is one longer than that from a Mealy machine for the same input because Moore machine always begins with one automatic start symbol.

For a given Mealy machine Me and the Moore machine M_o which have the automatic start state character '*a*', we will say that these two machines are equivalent if for every input string the output string from M_o is exactly '*a*' concatenated with the output from Me i.e.

$$a f_{Me} = f_{Mo}(w)$$

where w is the input string and a is the output of M_o from its initial state.

Mealy machine equivalent of given Moore machine

Theorem 16.1 If M_o is a Moore machine, then there is a Mealy machine M_e equivalent to M_o.

Proof: Let $M_o = (Q, \Sigma, \delta, q_0, O, f)$

and $M_e = (Q, \Sigma, \delta, q_0, O, f')$. We define

$$f'(q, a) = f(\delta(q, a)) \text{ for all } q \text{ and } a$$

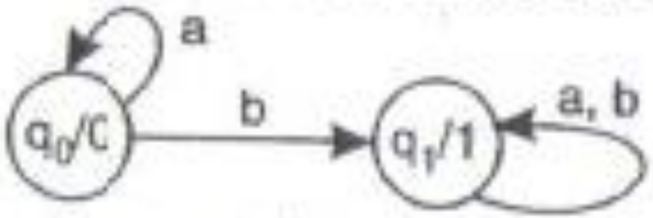
Then M_o and M_e enter the same sequence of states on the same input and with each transition M_e produces the output that M_o associates with the state entered.

Alternatively, M_e is constructed from M_o as follows:

Consider any state q of M. The output corresponding to this state is t. Let us consider all the edges that enter this state. Each of them is labeled with an input letter, say, a, b, c, ... and having output t. Now if we change the labels *a*, *b*, *c*, ... to *a*/*t*, *b*/*t*, *c*/*t*, ... and erase *t* from the inside the state *q*, we shall have output t on the incoming edges before they enter *q*.

If we repeat the procedure for every state $q_0, q_1, ..., q_n$, we get M_e which is equivalent to given M_o.

Example 54. Convert the following Moore machine to an equivalent Mealy machine.



Solution. The processing steps of output functions are

$f'(q_0, a) = f(\delta(q_0, a)) = f(q_0) = 0$

$f'(q_0, b) = f(\delta(q_0, b)) = f(q_1) = 1$

$f'(q_1, a) = f(\delta(q_1, a)) = f(q_1) = 1$

$f'(q_1, b) = f(\delta(q_1, b)) = f(q_1) = 1$

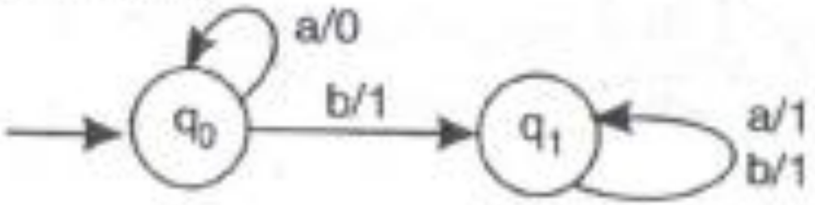
Alternatively,

At q_0, one arc labeled by a is incoming. Now it will be labeled by $a/0$, since output for q_0 was 0.

At q_1, three arcs labeled by b, a and b are incoming. Now they become $b/1$, $a/1$ and $b/1$ as output for q_1 was 1.

Change the labels of states to q_0 and q_1.

∴ The equivalent Mealy machine is



Example 55. Convert to a Mealy machine which is equivalent to the given Moore machine given in table.

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

$f(q_0) = 0$
$f(q_1) = 1$
$f(q_2) = 0$
$f(q_3) = 0$

Solution. For each input symbol symbol in Mo machine we construct the pair of next state and its corresponding output. The processing steps of output functions are

$f(\delta(\cdot)) = f(\text{state}) = \text{output}$

$$f'(q_0, 0) = f(\delta(q_0, 0)) = f(q_3) = 0$$
$$f'(q_0, 1) = f(\delta(q_0, 1)) = f(q_1) = 1$$
$$f'(q_1, 0) = f(\delta(q_1, 0)) = f(q_1) = 1$$
$$f'(q_1, 1) = f(\delta(q_1, 1)) = f(q_2) = 0$$
$$f'(q_2, 0) = f(\delta(q_2, 0)) = f(q_2) = 0$$
$$f'(q_2, 1) = f(\delta(q_2, 1)) = f(q_3) = 0$$
$$f'(q_3, 0) = f(\delta(q_3, 0)) = f(q_3) = 0$$
$$f'(q_3, 1) = f(\delta(q_3, 1)) = f(q_0) = 0$$

The state transition table for equivalent Mealy machine is

Present state	Next state			
	Input a = 0		Input *a* = 1	
	State	*Output*	*State*	*Output*
$\rightarrow q_0$	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

Moore machine equivalent of given Mealy machine.

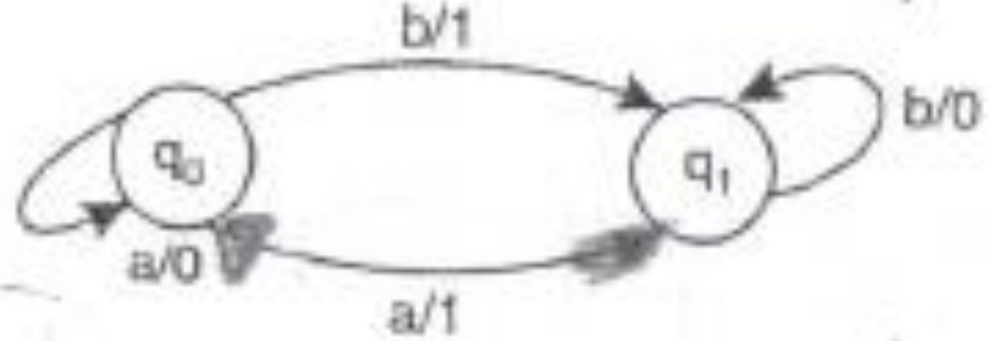
Theorem 16.2. Let Me = $(Q, \Sigma, \delta, q_0, O, f)$, then there is a Moore machine M_0 equivalent to Me.

Proof. Let $M_o = (Q, \Sigma, O, \delta', f', [q_0, b_0])$ where b_0 is an arbitrary selected member of O. That is, the states of M_o are pairs $[q, b]$ consisting of a state of M_e and an output symbol. We define

$$\delta'([q, b], a) = [\delta(q, a), f(q, a)] \text{ and } f'(q, b) = b.$$

The second component of $[q, b]$ of M_o is the output made by M_e on some transition into state q. The first component of M_o determines the moves.

Example 56. Convert the given Mealy machine into an equivalent Moore machine.



Solution. Let Me be the given Mealy machine. The state transition table of M_e is

Present state	*Next state*			
	Input a		Input b	
	State	*Output*	*State*	*Output*
q_0	q_0	0	q_1	1
q_1	q_0	1	q_1	0

Now the states of Moore machine (M_o) are $[q_0, 0]$, $[q_0, 1]$, $[q_1, 0]$ and $[q_1, 1]$. We have to find out

$$\delta'([q, b], a) = [\delta(q, a), \lambda(q, a)] \text{ and } \lambda'[q, b] = b$$

So,

$$\delta'([q_0, 0], a) = [\delta(q_0, a), \lambda(q_0, a)] = [q_0, 0]$$
$$\delta'([q_0, 0], b) = [\delta(q_0, b), \lambda(q_0, b)] = [q_1, 1]$$
$$\delta'([q_0, 1], a) = [\delta(q_0, a), \lambda(q_0, a)] = [q_0, 0]$$
$$\delta'([q_0, 1], b) = [\delta(q_0, b), \lambda(q_0, b)] = [q_1, 1]$$

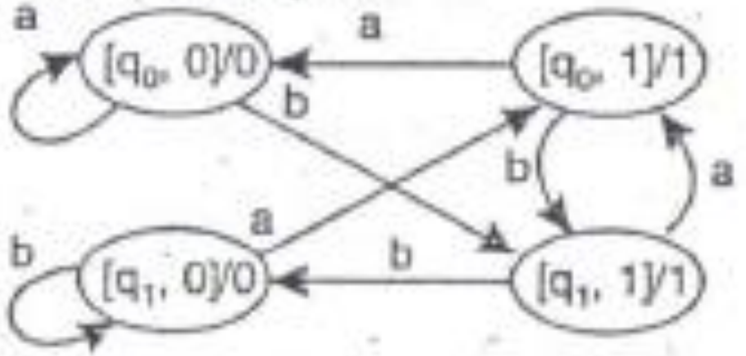
Similarly

$$\delta'([q_1, 0], a) = [q_0, 1]$$
$$\delta'([q_1, 1], a) = [q_0, 1]$$
$$\delta'([q_1, 0], b) = [q_1, 0]$$
$$\delta'([q_1, 1], b) = [q_1, 0]$$

The transition table for equivalent Mo is

Present state	Next state		Output
	a	b	
$[q_0, 0]$	$[q_0, 0]$	$[q_1, 1]$	0
$[q_0, 1]$	$[q_0, 0]$	$[q_1, 1]$	1
$[q_1, 0]$	$[q_0, 1]$	$[q_1, 0]$	0
$[q_1, 1]$	$[q_0, 1]$	$[q_1, 0]$	1

The transition diagram for equivalent M_o is



16.9 Pushdown Automata

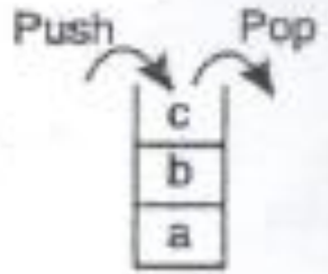
why FSA can not accept CFL?

Finite state automata (FSA) accept only regular languages. FSA can not accept other type of languages such as context free language (CFL), since they have only finite amount of memory and the recognition of a CFL may require storing an unbounded amount of information.

A pushdown automata (PDA) is similar to FSA, except that PDA has an auxiliary stack which provides an unlimited amount of memory. A language L is recognised by a pushdown automaton, iff L is context free.

Note: A PDA uses three stack seprations

- The pop operation that reads the top symbol and removes it from the stack.
- The push operation that writes a designated symbol onto the top of the stack.
- The nop operation does nothing to the stack.



Definition: Pushdown Automata (PDA)

A PDA is formally defined as a 7-tuple machine

$$M = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$$

where

(*i*) Q is a finite set of states.

(*ii*) Σ is a finite set of input alphabets.

(*iii*) Γ is the finite set of stack symbols.

(*iv*) $q_0 \in Q$ is the initial state

(*v*) z_0 is the initial stack symbol, placed on the top of the stack.

(*vi*) $F \subseteq Q$ is the set of finite states.

(*vii*) δ is the transition function and is defined by

PDA accepts a given string if it completely processes the string and ends up in a final state.

- Note that, if PDA terminates in a stack empty or no transition situation without finishing the input string, then it rejects the string, regardless the resulting state.

→ Based on the processing of input string by the machine, a PDA can be Deterministic PDA. Non deterministic PDA

→ A PDA is deterministic, if each input string can only be processed by the machine in only one way i.e. for the same input symbol and same stack symbol, there must be only one choice.

→ A PDA is nondeterminisitc, if there is some string that can be processed by it in more than one way.

16.10. Turing Machine

There are another class of models for computing machines, called Turing machine names after A. M. Turing. A Turing Machine (TM) is a very simple machine, but logically speaking, has all the power of any digital computer. There are two principal components of a TM, its unlimited auxiliary memory and instruction rules. The instructions open the memory, reading and writing to it as the computation proceeds. The memory component is a tape that is infinitely long in both directions. The tape is divided into squares (blocks), so that each squares stores at any moment a single symbol. (Blank squares will be said to store the symbol blank, denoted by B). The head is capable of performing three operations, reading the symbol contained in the square being scanned, writing a new (not necessarily) symbol in the scanned square, and shifting the tape one square to either direction. When a new symbol is written in the tape, it replaces the symbol previously there. **The machine starts in some state S_i, reads the symbol currently scanned by the head, writes there a** new symbol, shifts right or left according to its state table, and enter state Sj. The machine receives its imputs by reading the pattern of symbols written on the tape. Its output has the dual function of providing the head with the new symbols to be written on the tape and shifting the head to either direction. At the end of the computation, a new pattern of symbols is written on the tape. This pattern is the final objective of the entire computation.

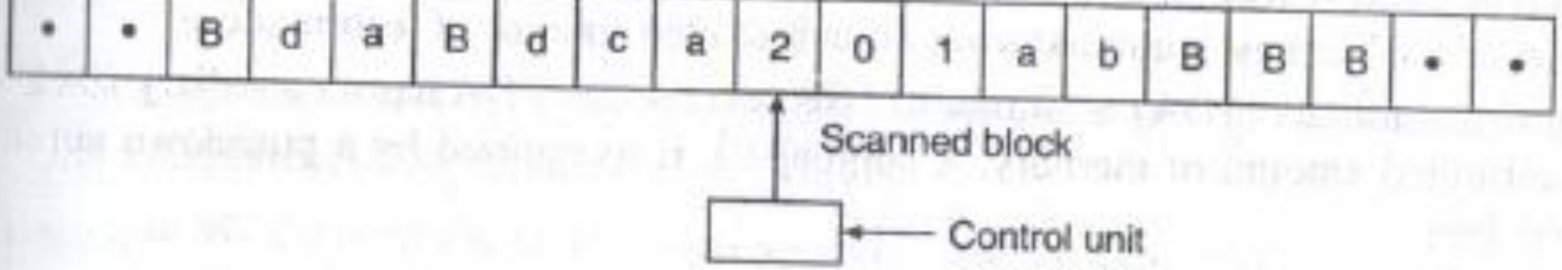


Fig. 16.20. A representation of Turing machine

Representation of Turing machine

The machine is deterministic in the sense that at each moment its next act is completely determined by its internal state at that moment and the symbol printed on the block scanned at that moment. Specifically, in terms of a finite alphabet of symbols which the machine is able to recognize, the machine is capable of the following acts, given an internal state and a symbol on the scanned block.

(i) Erase that symbol, print a new symbol from its alphabet, and (possibly) go into a different predetermined internal state.

(ii) Move one block to the right (that is, scan the block located immediately to the right of the original scanned block) and (possibly) go into another predetermined internal state.

(iii) Move one block to the left and (possibly) go into another predetermined internal state.

(iv) Come to a complete halt of operations.

Definition. The difinition of a Turing machine parallels that of a finite automaton. Formally,

A Turing machine M is defined by

$$M = (Q \, . \, S, G, d, q_0, B, F).$$

where

Q is the set of internal states.

S is the input alphabet,

G is a finite set of symbols called the tape alphabet.

d is the transition function,

$B \in G$ is a special symbol called the blank.

$q_0 \in Q$ is the initial state,

$F \subseteq Q$ is the set of final states.

The transition function d is defined as

$$d : Q \times G \rightarrow Q \times G \times \{L, R\}$$

In general d is a partial function on $Q \times G$; its interpretation gives the principle by which a uring machine operates.