4CS4-6: Theory of Computation (Mealy and Moore Machines)

Lecture 06: Feb. 01, 2019

Prof. K.R. Chowdhary *: Professor of CS*

6.1 Introduction

The finite automata have input a sequence of symbols on input tape and the output in the form of yes/no (accepted/rejected). When compared with the real computers, this turns out to be a limitation as there is no alphabet sequence produced at the output. Many times, there is requirement of sequence of symbols in the output rather than simply accepted or rejected. This is because, in real computers, input is in the form of programs and data, and the output is alphanumeric string, like - in speech processing the input may be sound signals (phonetic symbols) and output in text form (English language characters), and just the reverse in the case of text-to-speech translation. In addition, the output may appear immediately after the input is processed, or it may be stored in a memory and sent to the output device later at one stretch. The machines to implement these are called finite automata with output or *finite state transducers*.

6.2 Moore Machine

Moore machine is a finite state automaton, where outputs is determined by the current state alone, and do not depend directly on input. Most digital electronic systems designed as clocked sequential systems are a restricted form of Moore machine, where the state changes are decided by global clock signal. Typically, the current state is stored in flip-flops, and a global clock signal is connected to the "clock" input of the flop-flop. A typical electronic Moore machine includes a combinational logic chain to decode the current state into the output. The instant the current state changes, it ripples through a chain and almost instantaneously the outputs changes. Thus, in effect a Moore machine duplicates finite automaton.

A Moore machine is defined as a six-tuple,

$$M_o = (Q, \Sigma, \delta, s, \Gamma, \lambda) \tag{6.1}$$

where, the first four tuples: Q, Σ, δ, s are the same as in the FA. The symbol Γ (Gamma) is a set of output alphabets, and λ (Lambda) is mapping function from states to output, i.e., $\lambda : Q \rightarrow \Gamma$. For the sake of convenience the reference to symbols and alphabets means the same.

If the states of M_o are q_0, q_1, q_2, ..., then output corresponding to these states is $\lambda(q_0)$, $\lambda(q_1)$, $\lambda(q_2)$, ..., respectively. Thus, every input string creates an output string corresponding to the states through which the machine moves, $|\Gamma| \leq |Q|$.

A Moore machine does not define a language on its input alphabets. Thus, there is no concept of final state in the Moore machine. However, since it has finite number of states, it is a finite automaton. Processing of an input string gets terminated when last input alphabet is received and last output alphabet is printed. However, it is possible to turn a Moore machine into a language recognizer, i.e., produces Yes/No. This is demonstrated later in this chapter.

Let us consider that there is a Moore machines with states q_0, q_1, q_2, q_3 and the corresponding outputs a, b, c, d, respectively. The q_0 is a start state. Suppose that input 101 causes the Moore machine to switch to states q_1, q_2, q_3 respectively, starting from the q_0. Since each output symbol associates with a state and during the input 101 the machine has seen the states q_0, q_1, q_2, and q_3, the output will also correspond to these four states. Hence, the output corresponding to input 101 is *abcd*, which is one character longer than input.

Example 6.1 *Construct the transition diagram for the Moore machine given below:*

$$
\begin{aligned}
M_0 &= (Q, \Sigma, \delta, s, \Gamma, \lambda), \textit{where,} \\
Q &= \{q_0, q_1, q_2, q_3\}, \\
\Sigma &= \{a, b\}, \\
s &= q_0, \\
\Gamma &= \{0, 1\}.
\end{aligned}
$$

The transition function δ and the mapping function $\lambda : Q \to \Gamma$ are given in Table 6.1. The transition diagram for Moore machines can be obtained as shown in Figure 6.1.

Table 6.1: Transition Table for Moore Machine.

Current state	Next state for Input a	Next state Input b	Output $\lambda(q)$
q_0	q_1	q_3	1
q_1	q_3	q_1	0
q_2	q_0	q_3	0
q_3	q_3	q_2	1

The output associated with each state is shown in the circle of state itself. The slash ('/') sign separates the state name and the output. If the input is *abab* the corresponding outputs can be computed as follow. Initial state is q_0, and other sequence of states are:

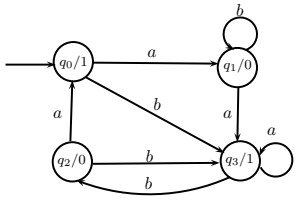

Figure 6.1: Moore Machine

$$\delta(q_0, a) = q_1$$
$$\delta(q_1, b) = q_1$$
$$\delta(q_1, a) = q_3$$
$$\delta(q_3, b) = q_2.$$

Hence, for the input sequence *abab* the output string is 10010. We note that Moore machine gives an output string of length $n+1$ for an input string of length n. One extra character in the output is due to the start state, which Moore machine was occupying before first input was applied. □

Example 6.2 *Design a Moore Machine which can count the number of occurrences of substring* 111 *in the input string of alphabet* $\Sigma = \{0, 1\}$.

We associate an output with each state in the Moore Machine as follows. Let us associate 1 to the state confirming completion of substring 111 in the input, and 0 to rest of all the states. The desired Moore Machine for this is shown in figure 6.2.

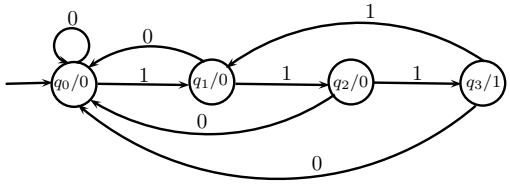

Figure 6.2: Moore machine for "111" pattern counter

Table 6.2: Transition table for Moore machine.

Current State	Next state for input = 0	Next state for input = 1	Output $\lambda(q)$
q_0	q_0	q_1	0
q_1	q_0	q_2	0
q_2	q_0	q_3	0
q_3	q_0	q_1	1

The transition table 6.2 for and output table for λ are determined from transition diagram. Considering an input $x = 01110111$, the output sequence is 000010001. Two appearance of 1 in the output shows two occurrences of substring 111 in input x. □

From the preceding example we note that, based on the accepting criteria of input string, new automata becomes a finite automata, which accepts all the languages having one or more occurrences of input 111 in the input. Thus, it demonstrates that it is possible to use a Moore machine as a language recognizer also.

6.3 Mealy Machine

Mealy machine is a finite state machine (and more accurately, a finite state transducer) that generates an output based its current state and input. This means that the state diagram will include both an input and output symbol for each transition edge.

The Mealy Machine is represented by,

$$M_e = (Q, \Sigma, \delta, s, \Gamma, \lambda), \tag{6.2}$$

where Q, Σ, δ, s and Γ are same as that in the Moore Machine, and $\lambda : Q \times \Sigma \to \Gamma$ is output function, i.e., $\lambda(q, a)$ is output associated with the transition from state q with input a for $q \in Q$ and $a \in \Sigma$. Consider that input sequence *abcd* and Mealy machine goes through sequence of transition states q_0, q_1, q_2, q_3 due to this input. The output sequence corresponding to this is denoted by $\lambda(q_0, a) \circ \lambda(q_1, b) \circ \lambda(q_2, c) \circ \lambda(q_3, d)$. The q_0 is start state, and other states are given by $\delta(q_0, a) = q_1$, $\delta(q_1, b) = q_2$ and $\delta(q_2, c) = q_3$. Note that, for the input string $|abcd| = 4$, the output string also has length 4. Thus, in Mealy Machine length of input and output strings are equal.

In Moore Machine, we noted that for an input ε (null string) there is finite output, even though there is no transition. However, for the Mealy machine, for ε input, there is no output. In addition, like the Moore Machine, the Mealy Machine also does not define a language, since it does not have any final state.

Example 6.3 *Construct a Mealy Machine to invert an arbitrary binary string with alphabet* $\Sigma = \{0, 1\}$.

Figure 6.3 shows the desired Mealy machine. The output of Mealy machine associated with the transition edge, is shown 1 and 0 corresponding to the input of 0 and 1, respectively. Various parameters of this Mealy Machine are as follows.

$\Sigma = \{0.1\}, \Gamma = \{0, 1\}$
$Q = \{q_0\}, s = q_0,$
δ and λ are given in Table 6.3.

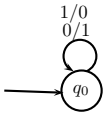

Figure 6.3: Mealy Machine

Table 6.3: Transition table for Mealy machine.

Current State	Next state / output for Input = 0	Next state / output For Input = 1
q_0	$q_0/1$	$q_1/0$

Note that the job performed by this simple machine is to complement the binary bits of input string with 1's complement.