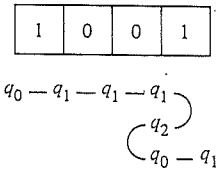

Fig. 2.22 Crossing sequences of 2DFA on input 1001.

justifies the second and third. Since $\delta(q_1, 1) = (q_2, L)$ we see the justification for the fourth move, which reverses the direction of travel. Then $\delta(q_2, 0) = (q_0, R)$ again reverses the direction, and finally $\delta(q_0, 1) = (q_1, R)$ explains the last move.

2.7 FINITE AUTOMATA WITH OUTPUT

One limitation of the finite automaton as we have defined it is that its output is limited to a binary signal: "accept"/"don't accept." Models in which the output is chosen from some other alphabet have been considered. There are two distinct approaches; the output may be associated with the state (called a *Moore machine*) or with the transition (called a *Mealy machine*). We shall define each formally and then show that the two machine types produce the same input-output mappings.

Moore machines

A Moore machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where Q, Σ, δ, and q_0 are as in the DFA. Δ is the *output alphabet* and λ is a mapping from Q to Δ giving the output associated with each state. The *output* of M in response to input $a_1 a_2 \cdots a_n$, $n \geq 0$, is $\lambda(q_0)\lambda(q_1) \cdots \lambda(q_n)$, where $q_0, q_1, \ldots, q_n$ is the sequence of states such that $\delta(q_{i-1}, a_i) = q_i$ for $1 \leq i \leq n$. Note that any Moore machine gives output $\lambda(q_0)$ in response to input ϵ. The DFA may be viewed as a special case of a Moore machine where the output alphabet is $\{0, 1\}$ and state q is "accepting" if and only if $\lambda(q) = 1$.

Example 2.17 Suppose we wish to determine the residue mod 3 for each binary string treated as a binary integer. To begin, observe that if i written in binary is followed by a 0, the resulting string has value $2i$, and if i in binary is followed by a 1, the resulting string has value $2i + 1$. If the remainder of $i/3$ is p, then the remainder of $2i/3$ is $2p$ mod 3. If $p = 0$, 1, or 2, then $2p$ mod 3 is 0, 2, or 1, respectively. Similarly, the remainder of $(2i + 1)/3$ is 1, 0, or 2, respectively.

It suffices therefore to design a Moore machine with three states, q_0, q_1, and q_2, where q_j is entered if and only if the input seen so far has residue j. We define

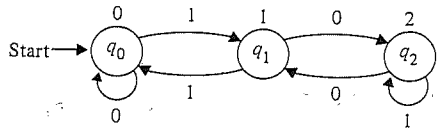

Fig. 2.23 A Moore machine calculating residues.

$\lambda(q_j) = j$ for $j = 0$, 1, and 2. In Fig. 2.23 we show the transition diagram, where outputs label the states. The transition function δ is designed to reflect the rules regarding calculation of residues described above.

On input 1010 the sequence of states entered is q_0, q_1, q_2, q_2, q_1, giving output sequence 01221. That is, ϵ (which has "value" 0) has residue 0, 1 has residue 1, 2 (in decimal) has residue 2, 5 has residue 2, and 10 (in decimal) has residue 1.

Mealy machines

A Mealy machine is also a six-tuple $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where all is as in the Moore machine, except that λ maps $Q \times \Sigma$ to Δ. That is, $\lambda(q, a)$ gives the output associated with the transition from state q on input a. The *output* of M in response to input $a_1 a_2 \cdots a_n$ is $\lambda(q_0, a_1)\lambda(q_1, a_2) \cdots \lambda(q_{n-1}, a_n)$, where $q_0, q_1, \ldots, q_n$ is the sequence of states such that $\delta(q_{i-1}, a_i) = q_i$ for $1 \leq i \leq n$. Note that this sequence has length n rather than length $n + 1$ as for the Moore machine, and on input ϵ a Mealy machine gives output ϵ.

Example 2.18 Even if the output alphabet has only two symbols, the Mealy machine model can save states when compared with a finite automaton. Consider the language $(\mathbf{0} + \mathbf{1})^*(\mathbf{00} + \mathbf{11})$ of all strings of 0's and 1's whose last two symbols are the same. In the next chapter we shall develop the tools necessary to show that this language is accepted by no DFA with fewer than five states. However, we may define a three-state Mealy machine that uses its state to remember the last symbol read, emits output y whenever the current input matches the previous one, and emits n otherwise. The sequence of y's and n's emitted by the Mealy machine corresponds to the sequence of accepting and nonaccepting states entered by a DFA on the same input; however, the Mealy machine does not make an output prior to any input, while the DFA rejects the string ϵ, as its initial state is nonfinal.

The Mealy machine $M = (\{q_0, p_0, p_1\}, \{0, 1\}, \{y, n\}, \delta, \lambda, q_0)$ is shown in Fig. 2.24. We use the label a/b on an arc from state p to state q to indicate that $\delta(p, a) = q$ and $\lambda(p, a) = b$. The response of M to input 01100 is *nnyny*, with the sequence of states entered being $q_0 p_0 p_1 p_1 p_0 p_0$. Note how p_0 remembers a zero and p_1 remembers a one. State q_0 is initial and "remembers" that no input has yet been received.

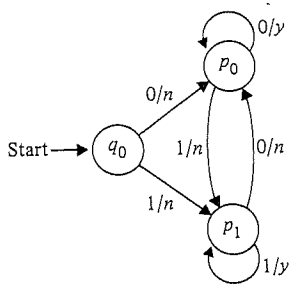

Fig. 2.24 A Mealy machine.

Equivalence of Moore and Mealy machines

Let M be a Mealy or Moore machine. Define $T_M(w)$, for input string w, to be the output produced by M on input w. There can never be exact identity between the functions T_M and $T_{M'}$ if M is a Mealy machine and M' a Moore machine, because $|T_M(w)|$ is one less than $|T_{M'}(w)|$ for each w. However, we may neglect the response of a Moore machine to input ϵ and say that Mealy machine M and Moore machine M' are *equivalent* if for all inputs w, $bT_M(w) = T_{M'}(w)$, where b is the output of M' for its initial state. We may then prove the following theorems, equating the Mealy and Moore models.

Theorem 2.6 If $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ is a Moore machine, then there is a Mealy machine M_2 equivalent to M_1.

Proof Let $M_2 = (Q, \Sigma, \Delta, \delta, \lambda', q_0)$ and define $\lambda'(q, a)$ to be $\lambda(\delta(q, a))$ for all states q and input symbols a. Then M_1 and M_2 enter the same sequence of states on the same input, and with each transition M_2 emits the output that M_1 associates with the state entered. □

Theorem 2.7 Let $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ be a Mealy machine. Then there is a Moore machine M_2 equivalent to M_1.

Proof Let $M_2 = (Q \times \Delta, \Sigma, \Delta, \delta', \lambda', [q_0, b_0])$, where b_0 is an arbitrarily selected member of Δ. That is, the states of M_2 are pairs $[q, b]$ consisting of a state of M_1 and an output symbol. Define $\delta'([q, b], a) = [\delta(q, a), \lambda(q, a)]$ and $\lambda'([q, b]) = b$. The second component of a state $[q, b]$ of M_2 is the output made by M_1 on some transition into state q. Only the first components of M_2's states determine the moves made by M_2. An easy induction on n shows that if M_1 enters states $q_0, q_1, \ldots, q_n$ on input $a_1a_2 \cdots a_n$, and emits outputs $b_1, b_2, \ldots, b_n$, then M_2 enters states $[q_0, b_0], [q_1, b_1], \ldots, [q_n, b_n]$ and emits outputs $b_0, b_1, b_2, \ldots, b_n$. □

Example 2.19 Let M_1 be the Mealy machine of Fig. 2.24. The states of M_2 are $[q_0, y]$, $[q_0, n]$, $[p_0, y]$, $[p_0, n]$, $[p_1, y]$, and $[p_1, n]$. Choose $b_0 = n$, making $[q_0, n]$

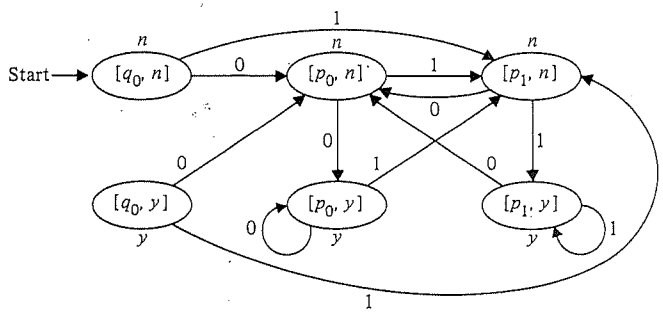

Fig. 2.25 Moore machine constructed from Mealy machine.

M_2's start state. The transitions and outputs of M_2 are shown in Fig. 2.25. Note that state $[q_0, y]$ can never be entered and may be removed.

2.8 APPLICATIONS OF FINITE AUTOMATA

There are a variety of software design problems that are simplified by automatic conversion of regular expression notation to an efficient computer implementation of the corresponding finite automaton. We mention two such applications here; the bibliographic notes contain references to some other applications.

Lexical analyzers

The tokens of a programming language are almost without exception expressible as regular sets. For example, ALGOL identifiers, which are upper- or lower-case letters followed by any sequence of letters and digits, with no limit on length, may be expressed as

$$(\text{letter})(\text{letter} + \text{digit})^*$$

where "letter" stands for $\mathbf{A} + \mathbf{B} + \cdots + \mathbf{Z} + \mathbf{a} + \mathbf{b} + \cdots + \mathbf{z}$, and "digit" stands for $\mathbf{0} + \mathbf{1} + \cdots + \mathbf{9}$. FORTRAN identifiers, with length limit six and letters restricted to upper case and the symbol \$, may be expressed as

$$(\text{letter})(\epsilon + \text{letter} + \text{digit})^5$$

where "letter" now stands for $(\$ + \mathbf{A} + \mathbf{B} + \cdots + \mathbf{Z})$. SNOBOL arithmetic constants (which do not permit the exponential notation present in many other languages) may be expressed as

$$(\epsilon + -)(\text{digit}^+(\cdot\ \text{digit}^* + \epsilon) + \cdot\ \text{digit}^+)$$

A number of *lexical-analyzer generators* take as input a sequence of regular expressions describing the tokens and produce a single finite automaton recogniz-