

is easy to see that  $M_3$  accepts a word if and only if  $L_1 \neq L_2$ . Hence, by Theorem 3.7, there is an algorithm to determine if  $L_1 = L_2$ .  $\square$

### 3.4 THE MYHILL-NERODE THEOREM AND MINIMIZATION OF FINITE AUTOMATA

Recall from Section 1.5 our discussion of equivalence relations and equivalence classes. We may associate with an arbitrary language  $L$  a natural equivalence relation  $R_L$ ; namely,  $xR_L y$  if and only if for each  $z$ , either both or neither of  $xz$  and  $yz$  is in  $L$ . In the worst case, each string is in an equivalence class by itself, but there may be fewer classes. In particular, the *index* (number of equivalence classes) is always finite if  $L$  is a regular set.

There is also a natural equivalence relation on strings associated with a finite automaton. Let  $M = (Q, \Sigma, \delta, q_0, F)$  be a DFA. For  $x$  and  $y$  in  $\Sigma^*$  let  $xR_M y$  if and only if  $\delta(q_0, x) = \delta(q_0, y)$ . The relation  $R_M$  is reflexive, symmetric, and transitive, since “=” has these properties, and thus  $R_M$  is an equivalence relation.  $R_M$  divides the set  $\Sigma^*$  into equivalence classes, one for each state that is reachable from  $q_0$ . In addition, if  $xR_M y$ , then  $xzR_M yz$  for all  $z$  in  $\Sigma^*$ , since by Exercise 2.4,

$$\delta(q_0, xz) = \delta(\delta(q_0, x), z) = \delta(\delta(q_0, y), z) = \delta(q_0, yz).$$

An equivalence relation  $R$  such that  $xRy$  implies  $xzRyz$  is said to be *right invariant* (with respect to concatenation). We see that every finite automaton induces a right invariant equivalence relation, defined as  $R_M$  was defined, on its set of input strings. This result is formalized in the following theorem.

**Theorem 3.9** (*The Myhill-Nerode theorem*). The following three statements are equivalent:

- 1) The set  $L \subseteq \Sigma^*$  is accepted by some finite automaton.
- 2)  $L$  is the union of some of the equivalence classes of a right invariant equivalence relation of finite index.
- 3) Let equivalence relation  $R_L$  be defined by:  $xR_L y$  if and only if for all  $z$  in  $\Sigma^*$ ,  $xz$  is in  $L$  exactly when  $yz$  is in  $L$ . Then  $R_L$  is of finite index.

*Proof*

(1)  $\rightarrow$  (2) Assume that  $L$  is accepted by some DFA  $M = (Q, \Sigma, \delta, q_0, F)$ . Let  $R_M$  be the equivalence relation  $xR_M y$  if and only if  $\delta(q_0, x) = \delta(q_0, y)$ .  $R_M$  is right invariant since, for any  $z$ , if  $\delta(q_0, x) = \delta(q_0, y)$ , then  $\delta(q_0, xz) = \delta(q_0, yz)$ . The index of  $R_M$  is finite, since the index is at most the number of states in  $Q$ . Furthermore,  $L$  is the union of those equivalence classes that include a string  $x$  such that  $\delta(q_0, x)$  is in  $F$ , that is, the equivalence classes corresponding to final states.

(2)  $\rightarrow$  (3) We show that any equivalence relation  $E$  satisfying (2) is a *refinement* of  $R_L$ ; that is, every equivalence class of  $E$  is entirely contained in some equivalence class of  $R_L$ . Thus the index of  $R_L$  cannot be greater than the index of  $E$  and so is

finite. Assume that  $xEy$ . Then since  $E$  is right invariant, for each  $z$  in  $\Sigma^*$ ,  $xzEyz$ , and thus  $yz$  is in  $L$  if and only if  $xz$  is in  $L$ . Thus  $xR_L y$ , and hence the equivalence class of  $x$  in  $E$  is contained in the equivalence class of  $x$  in  $R_L$ . We conclude that each equivalence class of  $E$  is contained within some equivalence class of  $R_L$ .

(3)  $\rightarrow$  (1) We must first show that  $R_L$  is right invariant. Suppose  $xR_L y$ , and let  $w$  be in  $\Sigma^*$ . We must prove that  $xwR_L yw$ ; that is, for any  $z$ ,  $xwz$  is in  $L$  exactly when  $ywz$  is in  $L$ . But since  $xR_L y$ , we know by definition of  $R_L$  that for any  $v$ ,  $xv$  is in  $L$  exactly when  $yv$  is in  $L$ . Let  $v = wz$  to prove that  $R_L$  is right invariant.

Now let  $Q'$  be the finite set of equivalence classes of  $R_L$  and  $[x]$  the element of  $Q'$  containing  $x$ . Define  $\delta'([x], a) = [xa]$ . The definition is consistent, since  $R_L$  is right invariant. Had we chosen  $y$  instead of  $x$  from the equivalence class  $[x]$ , we would have obtained  $\delta'([x], a) = [ya]$ . But  $xR_L y$ , so  $xz$  is in  $L$  exactly when  $yz$  is in  $L$ . In particular, if  $z = az'$ ,  $xaz'$  is in  $L$  exactly when  $yaz'$  is in  $L$ , so  $xaR_L ya$ , and  $[xa] = [ya]$ . Let  $q'_0 = [\epsilon]$  and let  $F' = \{[x] \mid x \text{ is in } L\}$ . The finite automaton  $M' = (Q', \Sigma, \delta', q'_0, F')$  accepts  $L$ , since  $\delta'(q'_0, x) = [x]$ , and thus  $x$  is in  $L(M')$  if and only if  $[x]$  is in  $F'$ .  $\square$

**Example 3.7** Let  $L$  be the language  $0^*10^*$ .  $L$  is accepted by the DFA  $M$  of Fig. 3.2. Consider the relation  $R_M$  defined by  $M$ . As all states are reachable from the start state,  $R_M$  has six equivalence classes, which are

$$\begin{aligned} C_a &= (00)^*, & C_d &= (00)^*01, \\ C_b &= (00)^*0, & C_e &= 0^*100^*, \\ C_c &= (00)^*1, & C_f &= 0^*10^*1(0+1)^* \end{aligned}$$

$L$  is the union of three of these classes,  $C_c$ ,  $C_d$ , and  $C_e$ .

The relation  $R_L$  for  $L$  has  $xR_L y$  if and only if either

- i)  $x$  and  $y$  each have no 1's,

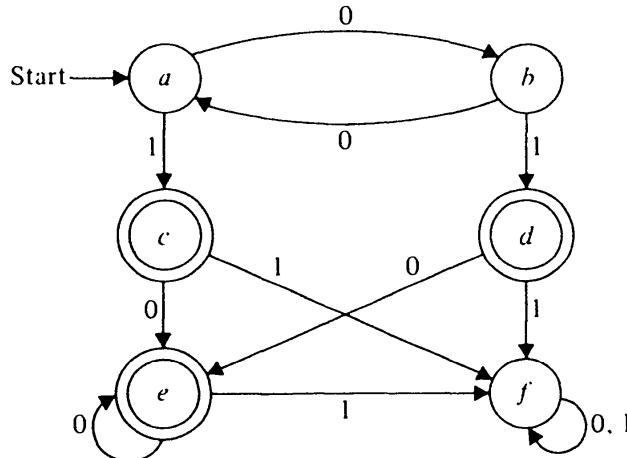


Fig. 3.2 DFA  $M$  accepting  $L$ .

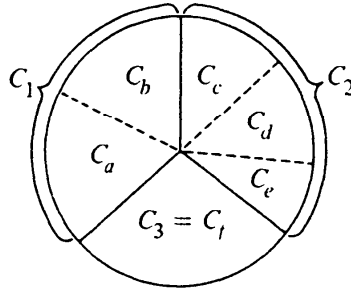


Fig. 3.3 Diagram showing  $R_M$  is a refinement of  $R_L$ .

- ii)  $x$  and  $y$  each have one 1, or
- iii)  $x$  and  $y$  each have more than one 1.

For example, if  $x = 010$  and  $y = 1000$ , then  $xz$  is in  $L$  if and only if  $z$  is in  $0^*$ . But  $yz$  is in  $L$  under exactly the same conditions. As another example, if  $x = 01$  and  $y = 00$ , then we might choose  $z = 0$  to show that  $xR_L y$  is false. That is,  $xz = 010$  is in  $L$ , but  $yz = 000$  is not.

We may denote the three equivalence classes of  $R_L$  by  $C_1 = 0^*$ ,  $C_2 = 0^*10^*$ , and  $C_3 = 0^*10^*1(0 + 1)^*$ .  $L$  is the language consisting of only one of these classes,  $C_2$ . The relationship of  $C_a, \dots, C_f$  to  $C_1, C_2$ , and  $C_3$  is illustrated in Fig. 3.3. For example  $C_a \cup C_b = (00)^* + (00)^*0 = 0^* = C_1$ .

From  $R_L$  we may construct a DFA as follows. Pick representatives for  $C_1, C_2$ , and  $C_3$ , say  $\epsilon, 1$ , and  $11$ . Then let  $M'$  be the DFA shown in Fig. 3.4. For example,  $\delta'([1], 0) = [1]$ , since if  $w$  is any string in  $[1]$  (note  $[1]$  is  $C_1$ ), say  $0^i10^j$ , then  $w0$  is  $0^{i+1}10^j$ , which is also in  $C_1 = 0^*10^*$ .

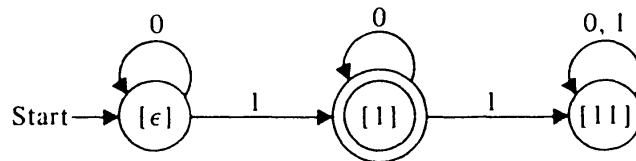


Fig. 3.4 The DFA  $M'$ .

### Minimizing finite automata

The Myhill-Nerode theorem has, among other consequences, the implication that there is an essentially unique minimum state DFA for every regular set.

**Theorem 3.10** The minimum state automaton accepting a set  $L$  is unique up to an isomorphism (i.e., a renaming of the states) and is given by  $M'$  in the proof of Theorem 3.9.

*Proof* In the proof of Theorem 3.9 we saw that any DFA  $M = (Q, \Sigma, \delta, q_0, F)$  accepting  $L$  defines an equivalence relation that is a refinement of  $R_L$ . Thus the number of states of  $M$  is greater than or equal to the number of states of  $M'$  of Theorem 3.9. If equality holds, then each of the states of  $M$  can be identified with one of the states of  $M'$ . That is, let  $q$  be a state of  $M$ . There must be some  $x$  in  $\Sigma^*$ , such that  $\delta(q_0, x) = q$ , otherwise  $q$  could be removed from  $Q$ , and a smaller automaton found. Identify  $q$  with the state  $\delta'(q'_0, x)$ , of  $M'$ . This identification will be consistent. If  $\delta(q_0, x) = \delta(q_0, y) = q$ , then, by the proof of Theorem 3.9,  $x$  and  $y$  are in the same equivalence class of  $R_L$ . Thus  $\delta'(q'_0, x) = \delta'(q'_0, y)$ .  $\square$

### A minimization algorithm

There is a simple method for finding the minimum state DFA  $M'$  of Theorems 3.9 and 3.10 equivalent to a given DFA  $M = (Q, \Sigma, \delta, q_0, F)$ . Let  $\equiv$  be the equivalence relation on the states of  $M$  such that  $p \equiv q$  if and only if for each input string  $x$ ,  $\delta(p, x)$  is an accepting state if and only if  $\delta(q, x)$  is an accepting state. Observe that there is an isomorphism between those equivalence classes of  $\equiv$  that contain a state reachable from  $q_0$  by some input string and the states of the minimum state FA  $M'$ . Thus the states of  $M'$  may be identified with these classes.

Rather than give a formal algorithm for computing the equivalence classes of  $\equiv$  we first work through an example. First some terminology is needed. If  $p \equiv q$ , we say  $p$  is *equivalent* to  $q$ . We say that  $p$  is *distinguishable* from  $q$  if there exists an  $x$  such that  $\delta(p, x)$  is in  $F$  and  $\delta(q, x)$  is not, or vice versa.

**Example 3.8** Let  $M$  be the finite automaton of Fig. 3.5. In Fig. 3.6 we have constructed a table with an entry for each pair of states. An  $X$  is placed in the table each time we discover a pair of states that cannot be equivalent. Initially an  $X$  is placed in each entry corresponding to one final state and one nonfinal state. In our example, we place an  $X$  in the entries  $(a, c)$ ,  $(b, c)$ ,  $(c, d)$ ,  $(c, e)$ ,  $(c, f)$ ,  $(c, g)$ , and  $(c, h)$ .

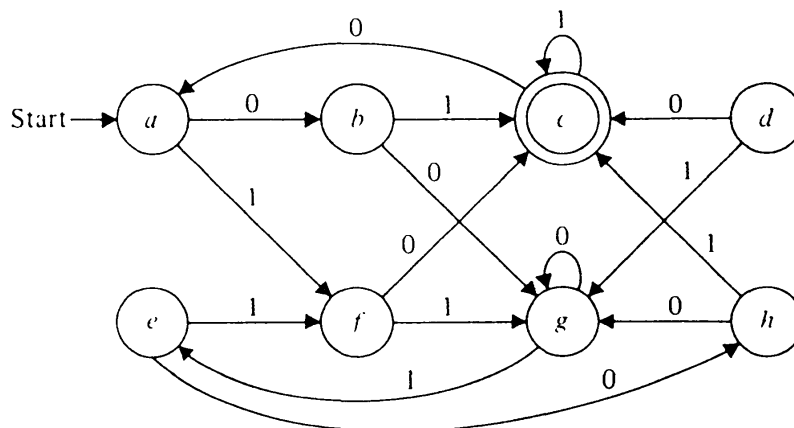


Fig. 3.5 Finite automaton.

<i>b</i>	<i>X</i>						
<i>c</i>	<i>X</i>	<i>X</i>					
<i>d</i>	<i>X</i>	<i>X</i>	<i>X</i>				
<i>e</i>		<i>X</i>	<i>X</i>	<i>X</i>			
<i>f</i>	<i>X</i>	<i>X</i>	<i>X</i>		<i>X</i>		
<i>g</i>	<i>X</i>	<i>X</i>	<i>X</i>	<i>X</i>	<i>X</i>	<i>X</i>	
<i>h</i>	<i>X</i>		<i>X</i>	<i>X</i>	<i>X</i>	<i>X</i>	<i>X</i>
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>

Fig. 3.6 Calculation of equivalent states.

Next for each pair of states  $p$  and  $q$  that are not already known to be distinguishable we consider the pairs of states  $r = \delta(p, a)$  and  $s = \delta(q, a)$  for each input symbol  $a$ . If states  $r$  and  $s$  have been shown to be distinguishable by some string  $x$ , then  $p$  and  $q$  are distinguishable by string  $ax$ . Thus if the entry  $(r, s)$  in the table has an  $X$ , an  $X$  is also placed at the entry  $(p, q)$ . If the entry  $(r, s)$  does not yet have an  $X$ , then the pair  $(p, q)$  is placed on a list associated with the  $(r, s)$ -entry. At some future time, if the  $(r, s)$  entry receives an  $X$ , then each pair on the list associated with the  $(r, s)$ -entry also receives an  $X$ .

Continuing with the example, we place an  $X$  in the entry  $(a, b)$ , since the entry  $(\delta(b, 1), \delta(a, 1)) = (c, f)$  already has an  $X$ . Similarly, the  $(a, d)$ -entry receives an  $X$  since the entry  $(\delta(a, 0), \delta(d, 0)) = (b, c)$  has an  $X$ . Consideration of the  $(a, e)$ -entry on input 0 results in the pair  $(a, e)$  being placed on the list associated with  $(b, h)$ . Observe that on input 1, both  $a$  and  $e$  go to the same state  $f$  and hence no string starting with a 1 can distinguish  $a$  from  $e$ . Because of the 0-input, the pair  $(a, g)$  is placed on the list associated with  $(b, g)$ . When the  $(b, g)$ -entry is considered, it receives an  $X$  on account of a 1-input, and hence the pair  $(a, g)$  receives an  $X$  since it was on the list for  $(b, g)$ . The string 01 distinguishes  $a$  from  $g$ .

On completion of the table in Fig. 3.6, we conclude that the equivalent states are  $a \equiv e$ ,  $b \equiv h$ , and  $d \equiv f$ . The minimum-state finite automaton is given in Fig. 3.7.

---

The formal algorithm for marking pairs of inequivalent states is shown in Fig. 3.8. Lemma 3.2 proves that the method outlined does indeed mark all pairs of inequivalent states.

**Lemma 3.2** Let  $M = (Q, \Sigma, \delta, q_0, F)$  be a DFA. Then  $p$  is distinguishable from  $q$  if and only if the entry corresponding to the pair  $(p, q)$  is marked in the above procedure.

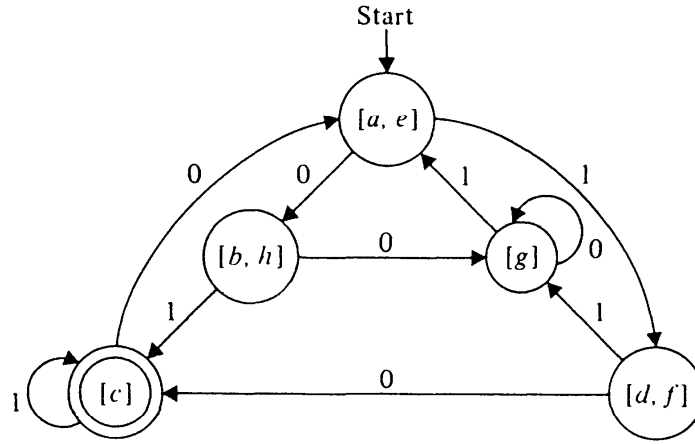


Fig. 3.7 Minimum state finite automaton.

```

begin
1) for  $p$  in  $F$  and  $q$  in  $Q - F$  do mark  $(p, q)$ ;
2) for each pair of distinct states  $(p, q)$  in  $F \times F$  or  $(Q - F) \times (Q - F)$  do
3)   if for some input symbol  $a$ ,  $(\delta(p, a), \delta(q, a))$  is marked then
4)     mark  $(p, q)$ ;
5)     recursively mark all unmarked pairs on the list for  $(p, q)$  and on the lists
       of other pairs that are marked at this step.
6)   end
7)   else /* no pair  $(\delta(p, a), \delta(q, a))$  is marked */
8)     for all input symbols  $a$  do
9)       put  $(p, q)$  on the list for  $(\delta(p, a), \delta(q, a))$  unless
10)       $\delta(p, a) = \delta(q, a)$ 
11)     end
12)   end
end

```

Fig. 3.8 Algorithm for marking pairs of inequivalent states.

*Proof* Assume  $p$  is distinguishable from  $q$ , and let  $x$  be a shortest string distinguishing  $p$  from  $q$ . We prove by induction on the length of  $x$  that the entry corresponding to the pair  $(p, q)$  is marked. If  $x = \epsilon$  then exactly one of  $p$  and  $q$  is a final state and hence the entry is marked in line (1). Assume that the hypothesis is true for  $|x| < i$ ,  $i \geq 1$ , and let  $|x| = i$ . Write  $x = ay$  and let  $t = \delta(p, a)$  and  $u = \delta(q, a)$ . Now  $y$  distinguishes  $t$  from  $u$  and  $|y| = i - 1$ . Thus by the induction hypothesis the entry corresponding to the pair  $(t, u)$  eventually is marked. If this event occurs after the pair  $(p, q)$  has been considered, then either the  $(p, q)$  entry has already been marked when  $(t, u)$  is considered, or the pair  $(p, q)$  is on the list associated with  $(t, u)$ , in which case it is marked at line (5). If  $(p, q)$  is considered after  $(t, u)$  then  $(p, q)$  is marked at the time it is considered. In any event the entry  $(p, q)$  is marked. A similar induction on the number of pairs marked shows that if the entry  $(p, q)$  is marked then  $p$  and  $q$  are distinguishable.  $\square$

The algorithm of Fig. 3.8 is more efficient than the obvious marking algorithm, although it is not the most efficient possible. Let  $\Sigma$  have  $k$  symbols and  $Q$  have  $n$  states. Line 1 takes  $O(n^2)$  steps.<sup>†</sup> The loop of lines 2 through 7 is executed  $O(n^2)$  times, at most once for each pair of states. The total time spent on lines 2 through 4, 6, and 7 is  $O(kn^2)$ . The time spent on line 5 is the sum of the length of all lists. But each pair  $(r, s)$  is put on at most  $k$  lists, at line 7. Thus the time spent on line 5 is  $O(kn^2)$ , so the total time is also  $O(kn^2)$ .

**Theorem 3.11** The DFA constructed by the algorithm of Fig. 3.8, with inaccessible states removed, is the minimum state DFA for its language.

*Proof* Let  $M = (Q, \Sigma, \delta, q_0, F)$  be the DFA to which the algorithm is applied and  $M' = (Q', \Sigma, \delta', [q_0], F')$  be the DFA constructed. That is,

$$Q' = \{[q] \mid q \text{ is accessible from } q_0\},$$

$$F' = \{[q] \mid q \text{ is in } F\}$$

and

$$\delta'([q], a) = [\delta(q, a)].$$

It is easy to show that  $\delta'$  is consistently defined, since if  $q \equiv p$ , then  $\delta(q, a) \equiv \delta(p, a)$ . That is, if  $\delta(q, a)$  is distinguished from  $\delta(p, a)$  by  $x$ , then  $ax$  distinguishes  $q$  from  $p$ . It is also easy to show that  $\delta'([q_0], w) = [\delta(q_0, w)]$  by induction on  $|w|$ . Thus  $L(M') = L(M)$ .

Now we must show that  $M'$  has no more states than  $R_L$  has equivalence classes, where  $L = L(M)$ . Suppose it did; then there are two accessible states  $q$  and  $p$  in  $Q$  such that  $[q] \neq [p]$ , yet there are  $x$  and  $y$  such that  $\delta(q_0, x) = q$ ,  $\delta(q_0, y) = p$ , and  $xR_L y$ . We claim that  $p \equiv q$ , for if not, then some  $w$  in  $\Sigma^*$  distinguishes  $p$  from  $q$ . But then  $xwR_L yw$  is false, for we may let  $z = \epsilon$  and observe that exactly one of  $xwz$  and  $ywz$  is in  $L$ . But since  $R_L$  is right invariant,  $xwR_L yw$  is true. Hence  $q$  and  $p$  do not exist, and  $M'$  has no more states than the index of  $R_L$ . Thus  $M'$  is the minimum state DFA for  $L$ .  $\square$

## EXERCISES

**3.1** Which of the following languages are regular sets? Prove your answer.

- a)  $\{0^{2^n} \mid n \geq 1\}$
- b)  $\{0^m 1^n 0^{m+n} \mid m \geq 1 \text{ and } n \geq 1\}$
- c)  $\{0^n \mid n \text{ is a prime}\}$
- d) the set of all strings that do not have three consecutive 0's.
- e) the set of all strings with an equal number of 0's and 1's.
- f)  $\{x \mid x \text{ in } (0+1)^*, \text{ and } x = x^R\}$   $x^R$  is  $x$  written backward; for example,  $(011)^R = 110$ .
- g)  $\{xwx^R \mid x, w \text{ in } (0+1)^+\}$
- \*h)  $\{xx^Rw \mid x, w \text{ in } (0+1)^+\}$

<sup>†</sup> We say that  $g(n)$  is  $O(f(n))$  if there exist constants  $c$  and  $n_0$  such that  $g(n) \leq cf(n)$  for all  $n \geq n_0$ .