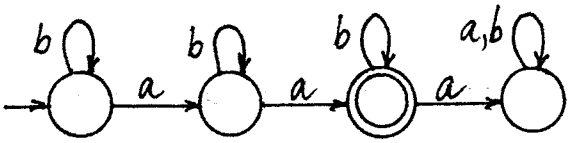
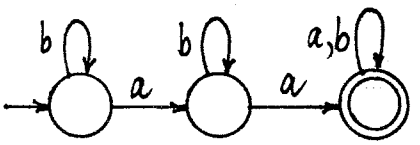
Chapter 2

Finite Automata and the Languages They Accept

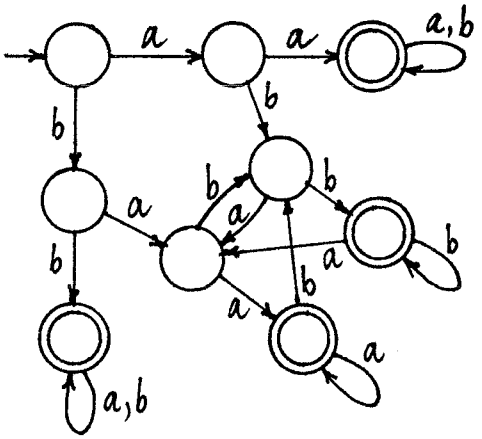
2.1 (a)



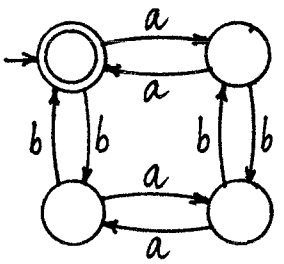
(b)



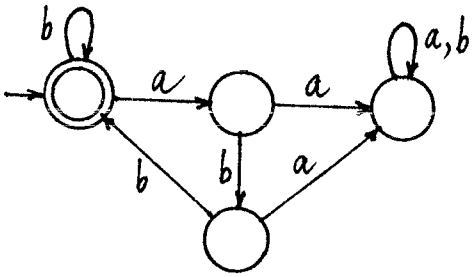
(d)



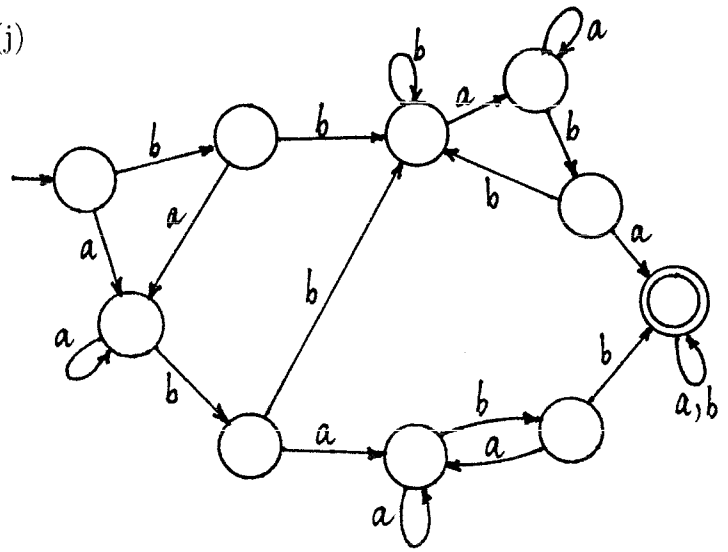
(g)



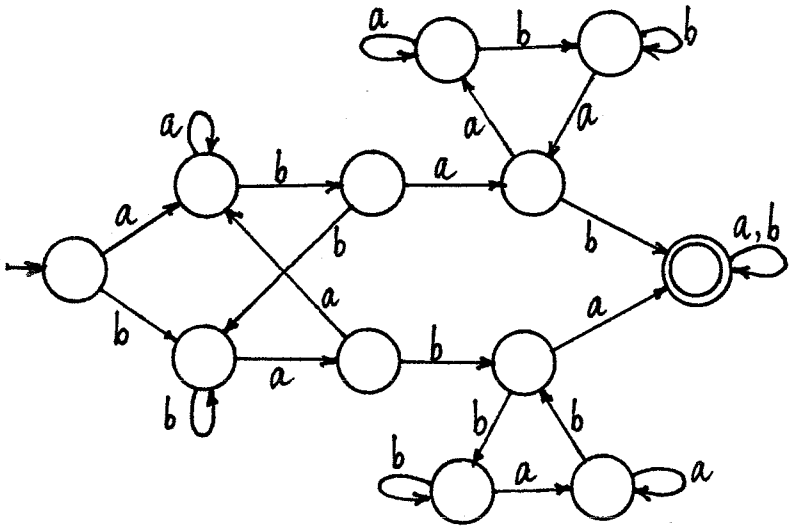
(i)



(j)



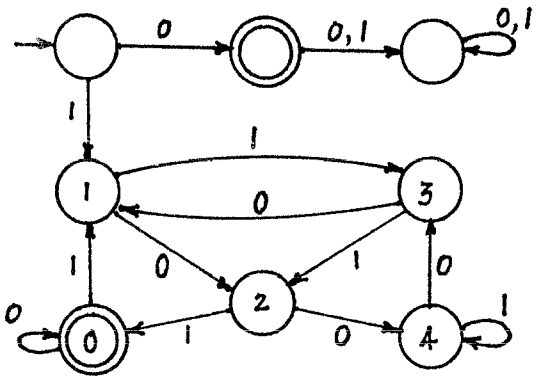
2.1(k)



2.3(b) There is an FA with $n+2$ states accepting $\{x\}$. It has one state for each of the $n+1$ prefixes of x, and one state N representing all the strings that are nonprefixes. For each state representing a prefix of x other than x itself, there is one transition to the next longer prefix and one to N; from the states x and N, both transitions go to N.

(c) There is an FA with $n+2$ states accepting $\{x\}\{a,b\}^*$. It is the same as the one in (b) in every way except that the transitions from the state x both go to x.

2.4.



2.6. In the basis step, $\delta^*(q,\Lambda) = q$ by definition of δ^*. In the induction step, suppose $\delta^*(q,x) = q$, and let $a \in \Sigma$. Then $\delta^*(q,xa) = \delta(\delta^*(q,x),a) = \delta(q,a) = q$.

2.7. Suppose $\delta^*(q_0,x) = q \in R$. By definition of R, there is a string y so that $\delta^*(q,y) \in A$. Therefore, $\delta^*(q_0,xy) = \delta^*(\delta^*(q_0,x),y) \in A$, and x is a prefix of an element of $L(M)$. On the other hand, if x is a prefix of an element of $L(M)$, then $\delta^*(q_0,xy) \in A$, for some string y, so that $\delta^*(q_0,x) \in R$. The conclusion is that M_1 accepts the language of all strings that are prefixes of elements of $L(M)$.

2.8(a) Invalid. For $\sigma \in \Sigma$, there is no way to use the definition to determine what $\delta^*(q,\sigma)$ is.

(b) This is a valid definition, since $\delta^*(q, \Lambda)$ is defined, and for each x with $|x| \geq 1$, $\delta^*(q, x)$ is defined in terms of $\delta^*(p, y)$ for some state p and some string y of length $|x| - 1$.

Let the function being defined here be called δ_1. Then we must show that $\delta_1(q, x) = \delta^*(q, x)$ for every q and every x. In the basis step, we check that $\delta_1(q, \Lambda) = \delta^*(q, \Lambda)$. This is clear, since both are defined to be q. Suppose that for some y, $\delta_1(q, y) = \delta^*(q, y)$ for every state q. Then for $a \in \Sigma$ and $q \in Q$,

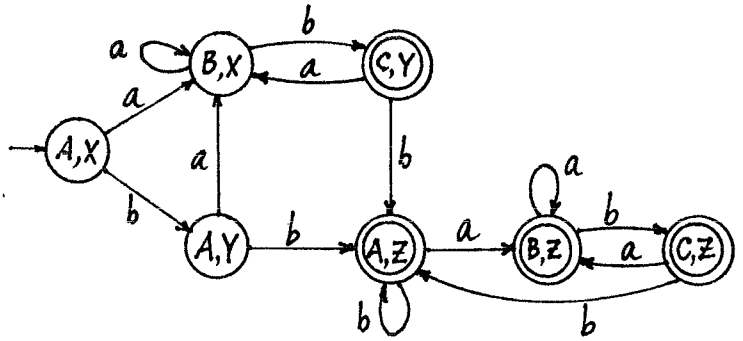
$$\delta_1(q, ay) = \delta_1(\delta(q, a), y) = \delta^*(\delta(q, a), y) = \delta^*(\delta(\delta^*(q, \Lambda), a), y) = \delta^*(\delta^*(q, a), y) = \delta^*(q, ay)$$

The first equality is the definition of δ_1; the second uses the induction hypothesis, with the state $\delta(q, a)$; the third and fourth use the definition of δ^*; and the last uses the formula in Exercise 2.5.

(c) This definition certainly leaves something to be desired. If $|xy| > 1$, there are strings w and z other than x and y for which $xy = wz$. If for some such x, y, w, and z, $\delta^*(\delta^*(q, x), y)$ and $\delta^*(\delta^*(q, w), z)$ were different, then the definition would not be a valid definition, because it would give different answers for $\delta^*(q, xy)$ and $\delta^*(q, wz)$ and these are supposed to be the same. However, it is possible to show that this can't happen, and so the definition may be said to be valid.

2.9(c) Let $L = \{ax \mid x \in \{a, b\}^*\}$. Suppose n is any integer and S is any set of strings of length n. If $S = \emptyset$, the equivalence cannot be correct, because L is nonempty. However, if x is any element of S, the string $1x$ is not an element of L, even though it ends with an element of S. Therefore, L does not have this property.

2.10. The diagram below works for all three parts, except that in part (b) the only accepting state is (C, Y) and in (c) (C, Z) is the only one.

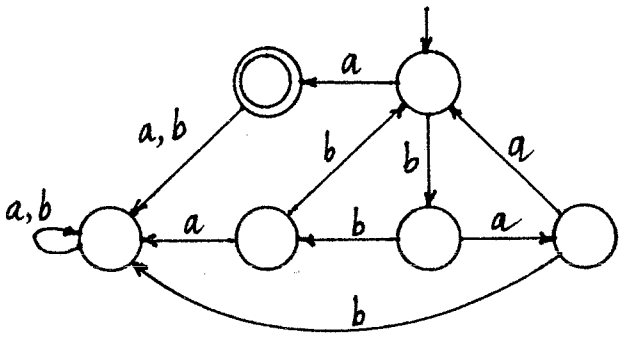


2.11. The proof is by structural induction on x. $\delta^*((p, q), \Lambda) = (p, q) = (\delta_1^*(p, \Lambda), \delta_2^*(q, \Lambda))$, from the definitions of δ^*, δ_1^*, and δ_2^*. Suppose x is a string for which $\delta^*((p, q), x) = (\delta_1^*(p, x), \delta_2^*(q, x))$, and let $\sigma \in \Sigma$. Then

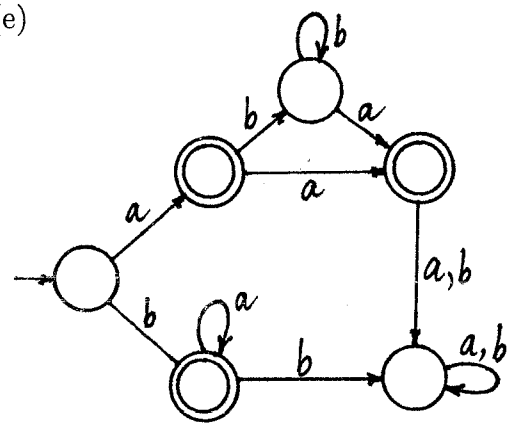
$$\begin{aligned}
\delta^*((p, q), x\sigma) &= \delta(\delta^*((p, q), x), \sigma) \\
&= \delta((\delta_1^*(p, x), \delta_2^*(q, x)), \sigma) \\
&= (\delta_1(\delta_1^*(p, x), \sigma), \delta_2(\delta_2^*(q, x), \sigma)) \\
&= (\delta_1^*(p, x\sigma), \delta_2^*(q, x\sigma))
\end{aligned}$$

The first inequality is true by definition of δ^*; the second is true by the induction hypothesis; the third is true by the definitions of δ; and the last is true by the definitions of δ_1^* and δ_2^*.

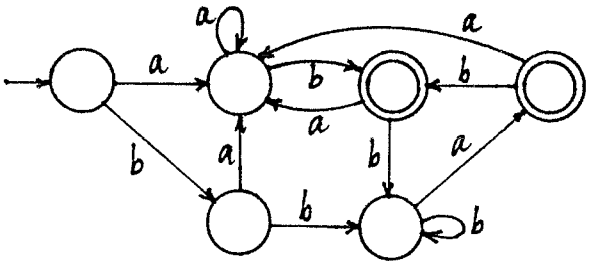
2.12(d)



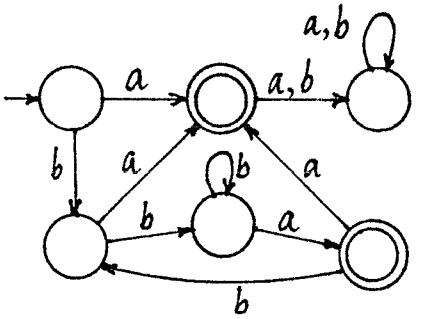
(e)



(f)



(g)



2.13. The simplest strings corresponding to the four states of the FA are Λ, a, ab, and aa. It is possible to show that these strings are pairwise L-distinguishable. For example, Λ and a are distinguished by the string a; Λ and ab are distinguished by the string Λ, as are a and ab, and a and aa, and Λ and aa; and aa and ab are distinguished by the string a.

2.14. Let x_1 and x_2 be distinct prefixes of x, and suppose $x_1y_1 = x_2y_2 = z$ and $|x_1| < |x_2|$. Then $x_1y_2 \notin L$ and $x_2y_2 \in L$.

2.15. A simple counter-example is provided by the language L of all even-length strings and the sequence $x_0, x_1, \ldots$, where $x_i = a^i$. For every n, x_n and x_{n_1} are distinguished with respect to L by the string Λ. However, L can be accepted by an FA with two states.

2.16. If L_n is any nonempty language such that every $x \in L_n$ has length n, then an FA accepting L_n must have at least $|n| + 2$ states, because for every $x \in L_n$, the $n + 2$ strings consisting of the $n + 1$ prefixes of x and a string y of length $n + 1$ are pairwise L_n-distinguishable. (If $x_1y_1 = x_2y_2 = x$, and $|x_1| < |x_2|$, then x_1 and x_2 are distinguished by the string y_1. Furthermore, each of the the prefixes is distinguishable from y with respect

to L_n.) The language $L = \{a, b\}^*$ is one example. Any infinite language that does not contain two strings of the same length is another.

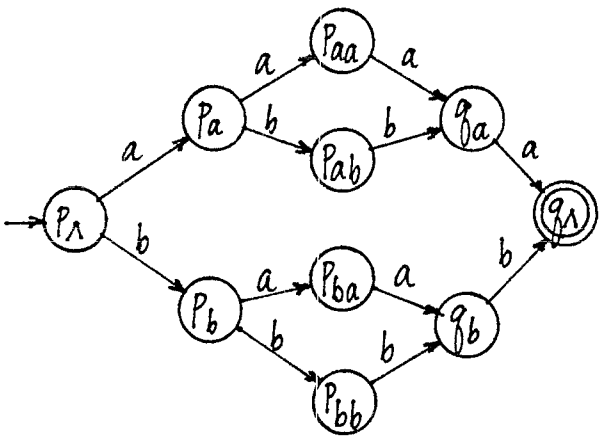
2.17(a) One answer is provided by two strings, neither of which is a prefix of an element of L (b and ba, for example). Two distinct nonnull elements of L provide another answer. In both these answers, the two strings are indistinguishable because nothing can be appended to either string (except Λ in the second case) so as to obtain an element of L. Another answer is two strings like aab and $aaabb$. Here an element of L is obtained in both cases if b is added to the end, and an element of L' is obtained in both cases if anything else is added to the end.

(b) The elements of $\{a^n \mid n \geq 0\}$ are pairwise L-distinguishable.

2.18. For L to be nonempty, n must obviously be even. If $n = 2m$, there is an FA with $(m+1)^2 + 1$ states accepting L. It has a state for each of the ordered pairs (i, j), where $0 \leq i \leq m$ and $0 \leq j \leq m$, and i and j represent the numbers of a's and b's, respectively, in the input string. For such a pair (i, j), each string with i a's and j b's is a prefix of an element of L. There is one additional state N corresponding to all the nonprefixes of elements of L. The transitions are the natural ones: if x has i a's and j b's, and $i < m$, then $\delta((i, j), a) = (i + 1, j)$; similarly for $\delta((i, j), b)$ if $j < m$. For each $i \leq m$, $\delta((m, i), a) = \delta((i, m), b) = N$.

This is the minimum possible number of states, because $(m+1)^2 + 1$ strings obtained by choosing one for each state are pairwise L-distinguishable.

2.19. Here is a diagram of an FA accepting L in the case $n = 2$, which we call M_2.



We have simplified the picture by leaving out one of the states. Every transition not shown explicitly is assumed to go to a nonaccepting state, from which both transitions return to that state.

It is straightforward to generalize this to arbitrary n, so as to obtain an FA M_n. For each string x of length $\leq n$, there is a state p_x corresponding to x (that is, for which $\{y \mid \delta^*(q_0, y) = p_x\}$ is $\{x\}$. In addition, for each x with $|x| < n$, there is another state,

which we call q_x, corresponding to the set $\{y \mid yx \in L\}$. The states p_x and q_x account collectively for all the strings that are prefixes of elements of L, and all the others are rejected. Since there are $2^{n+1}-1$ strings of length $\leq n$ and 2^n-1 strings of length $< n$, the total number of states in the FA is $(2^{n+1}-1)+(2^n-1)+1 = 3*2^n-1$. Now we show that no FA with fewer states can accept L, by showing that two strings corresponding to different states of M_n are distinguishable with respect to L.

Clearly any string that is a prefix of an element of L is distinguishable from any string that is not. Also, if x and y are both prefixes of elements of L and $|x| \neq |y|$, then x and y are distinguishable. It is therefore sufficient to show: (i) if $|x_1| = |x_2| \leq n$ and $x_1 \neq x_2$, then x_1 and x_2 are distinguishable; and (ii) if $|y_1| = |y_2|$, $y_1 \neq y_2$, $x_1y_1 \in L$, and $x_2y_2 \in L$, then x_1 and x_2 are distinguishable. Statement (i) is easy:

$$x_1a^{2n-2|x_1|}x_1^r \in L \text{ and } x_2a^{2n-2|x_1|}x_1^r \notin L$$

Statement (ii) is just as easy: if $x_1y_1 \in L$ and $y_1 \neq y_2$, then since $x_2y_2 \in L$, we must have $x_2y_1 \notin L$. Therefore, y_1 distinguishes x_1 and x_2.

2.20(a) This follows immediately from the fact that if M is an FA accepting L relative to L_{i+1}, then M accepts L relative to L_i.

(b) Suppose there is a constant N so that $n_i \leq N$ for every i. Let M_i be an FA with n_i states that accepts L relative to L_i. We may assume that the states of each M_i are elements of the set $\{q_0, q_1, \ldots, q_{N_1}\}$. Therefore, there must be infinitely many i's for which the FAs M_i are all equal, say to M. Then M must accept L relative to L_i for every i; therefore, since $\cup_{i=1}^{\infty} L_i = \Sigma^*$, M accepts L.

2.21. Assume in each part that $0 \leq i < j$.

(a) The string ba^{2i} distinguishes a^i and a^j.

(c) The string b^{2j} distinguishes a^i and a^j.

(d) The string ab^{i+1} distinguishes a^i and a^j.

(e) If i is odd, say $i = 2p+1$, the string b^{p+1} distinguishes a^i and a^j; if i is even, say $i = 2p$, the string ab^{p+1} distinguishes a^i and a^j.

(f) The string b^j distinguishes a^i and a^j.

(g) Suppose $0 \leq i < j$. For each integer m so that $m^3 \geq i$, $a^ia^{m^3-i} \in L$, and $a^ja^{m^3-i} = a^{m^3+(j-i)}$. If m is a number large enough that $m^3 + (j-i) < (m+1)^3$, then a^{m^3-i} distinguishes a^i and a^j.

(h) The string ba^ib distinguishes a^i and a^j.

2.22(a) $L = \{a^nba^{2n} \mid n \geq 0\}$. Suppose L can be accepted by an FA, and let n be the integer in the statement of the pumping lemma. Let $x = a^nba^{2n}$. Then $x \in L$, and $|x| \geq n$. Therefore, by the statement of the pumping lemma, $x = uvw$ for some strings u, v, and w satisfying $|uv| \leq n$, $|v| > 0$, and $uv^mw \in L$ for every $m \geq 0$. The first of these three conditions implies that v is a string of a's from the first group of a's in x, and the second implies that $v \neq \Lambda$. Therefore, for some $j > 0$, $uv^2w = a^{n+j}ba^{2n}$. However, the third condition says that uv^2w must be in L. This contradiction proves that L cannot be

accepted by an FA.

(c) $L = \{a^i b^j \mid j = i \text{ or } j = 2i\}$. Suppose $L = L(M)$ for some FA M, let n be the integer in the pumping lemma, and let $x = a^n b^n$. Then $x = uvw$ for some u, v, and w satisfying the usual three conditions. As before, v must be a^j for some $j > 0$. Therefore, $uv^2w = a^{n+j}b^n$, and this string cannot be in L because n is neither $n+j$ nor $2(n+j)$. This is the necessary contradiction.

(d) $L = \{a^i b^j \mid j \text{ is a multiple of } i\}$. Suppose L can be accepted by an FA, let n be the integer in the pumping lemma, and let $x = a^n b^n$. Then $x = uvw$ for some u, v, and w satisfying the usual three conditions. As in (a) and (c), v must be a^j for some $j > 0$, and $uv^2w = a^{n+j}b^n$. Since n cannot be a multiple of $n + j$, this is a contradiction.

(e) $L = \{x \in \{a, b\}^* \mid n_a(x) < 2n_b(x)\}$. Suppose L is accepted by an FA, let n be the integer in the pumping lemma, and let $x = a^{2n-1}b^n$. Then $x = uvw$ for some u, v, and w satisfying the usual three conditions. As before, v must be a^j for some $j > 0$, and $uv^2w = a^{2n+j-1}b^n$. Since $j - 1 \geq 0$, $2n + j - 1 \geq 2n$, and this is a contradiction.

(f) $L = \{x \in \{a, b\}^* \mid \text{no prefix of } x \text{ has more b's than a's}\}$. Suppose L is accepted by an FA, let n be the integer in the pumping lemma, and let $x = a^n b^n$. Then $x = uvw$, where u, v, and w satisfy the usual three conditions. As above, $v = a^j$ for some $j > 0$. Therefore, $uv^0w = uw = a^{n-j}b^n$, and this string cannot be in L because the string itself, which is a prefix of itself, has more b's than a's.

2.23. Statement II is sufficient to show that the language in part (g) cannot be accepted by an FA. If p and q are numbers so that $q > 0$ and for every $i \geq 0$, $p + iq = n^3$ for some n, then letting $i = q^2$, we have $p + q^2q = m^3$ for some integer m. Then, however, $p + (q^2 + 1)q = m^3 + q < m^3 + m < (m + 1)^3$. This is a contradiction, because an integer strictly between m^3 and $(m + 1)^3$ cannot be the cube of an integer. It is easy to see that statement II is not sufficient for any other part of this exercise.

Statement I is sufficient to show that the languages in (a) and (c) cannot be accepted by an FA. In both cases, if $uv^iw \in L$ for every i and $|v| > 0$, then v cannot contain both a's and b's, because then uv^2w would contain b's separated by a's; and v cannot contain only one distinct symbol, because for some i, uv^iw would not have the necessary ratio of a's to b's. In parts (d), (e), (f), and (h), statement I is not sufficient. In (d), for example, let $u = a$, $v = b$, and $w = \Lambda$; then for every i, $uv^iw = ab^i$, and each of these strings is in L.

2.24. Let n be the number of states in an FA $M = (Q, \Sigma, q_0, A, \delta)$ accepting L, and suppose $x \in L$ and $x = x_1x_2x_3 = x_1(a_1a_2 \ldots a_k)x_3$, where $k \geq n$. Denote by p_i the state $\delta^*(q_0, x_1a_1a_2 \ldots a_i)$ for each i with $0 \leq i \leq k$. Then just as in the proof of the pumping lemma, at least two of the states p_i must be the same; suppose $p_i = p_{i+j}$, where $j > 0$. Then let $u = a_1 \ldots a_i$, $v = a_{i+1} \ldots a_{i+j}$, and $w = a_{i+j+1} \ldots a_k$. We have $x_2 = uvw$ and $|v| > 0$, and the same argument as in the proof of the pumping lemma shows that $x_1uv^mwx_3 \in L$ for every $m \geq 0$.

2.25. The language L in Example 2.39 is such a language. Suppose L can be accepted by an FA, let n be the integer in the statement, let $x = ab^nc^n$, and let $x_1 = a$, $x_2 = b^n$, and $x_3 = c^n$. The statement says that for some u, v, and w with $|uv| \leq n$ and $|v| > 0$, $b^n = uvw$

and $auv^mwc^n \in L$ for every $m \geq 0$. This would mean that there are strings ab^ic^n in L for which $i \neq n$, which contradicts the definition of L.

2.26. According to the pumping lemma, if n is the number of states in an FA accepting L, and there is a string x in L with $|x| \geq n$, then L must be infinite.

2.27(a) We have an algorithm to construct an FA accepting the complement of the language accepted by a given FA. Apply this to find M_1' and M_2' accepting $L(M_1)'$ and $L(M_2)'$, respectively. Now construct the FA accepting $L(M_1') \cap L(M_2)'$; finally, apply the algorithm on page 67 to determine whether this FA accepts any strings.

(b) The algorithm for finding the states reachable from the initial state of an FA (see Example 2.34) can be modified so that it determines the states reachable from q using nonnull strings. The problem is then reduced to determining whether q is an element of this set.

(c) First eliminate the states that are unreachable from the initial state of M. Then apply the minimization algorithm described in Section 2.6 to obtain a minimum-state FA $M_1 = (Q_1, \Sigma, q_1, A_1, \delta)$ accepting L. x and y are L-distinguishable if and only if $\delta_1^*(q_1, x) \neq \delta_1^*(q_1, y)$.

(d) If $\delta^*(q_0, x) = q$, then x is a prefix of an element of L if and only if the set of states reachable from q includes at least one accepting state.

(e) Let $M = (Q, \Sigma, q_0, A, \delta)$. Eliminate the states of M that are unreachable from q_0. Then x is a suffix of an element of L if and only if $\delta^*(q, x) \in A$ for one of the remaining states q.

(f) Again let $M = (Q, \Sigma, q_0, A, \delta)$. Eliminate the states that are unreachable from q_0. Then x is a substring of an element of L if and only if, for some state q that remains, the set of states reachable from $\delta^*(q, x)$ contains an element of A.

(g) $L(M_1)$ is a subset of $L(M_2)$ if and only if $L(M_1) - L(M_2) = \emptyset$. Use the algorithm in the proof of Theorem 2.15 to find an FA M accepting the language $L(M_1) - L(M_2)$; then apply the first algorithm in Example 2.34 to determine whether $L(M) = \emptyset$.

(h) Let M_3 be the FA obtained from M_2 as follows: M_3 has the same states and transitions as M_2, and for every state q, if an accepting state of M_2 is reachable from q, then q is an accepting state of M_3. Then $L(M_3)$ is the language of prefixes of elements of $L(M_2)$. We can now determine whether $L(M_1) \subseteq L(M_3)$ as in (g).

2.28. No. Let L be the language of nonpalindromes over $\{a, b\}$. L cannot be accepted by an FA because its complement cannot. However, if x begins with a, $xb \in L$; if x begins with b, $xa \in L$; and if $x = \Lambda$, $xab \in L$. Therefore, L satisfies the condition for $k = 2$.

2.29(a) False. Every language over $\{a, b\}$, including ones that cannot be accepted by an FA, is a subset of $\{a, b\}^*$, which can.

(b) False. A counter-example is a finite subset of a language that cannot be accepted by an FA.

(c) False. If $L \subseteq \Sigma^*$ cannot be accepted by an FA, then L' cannot, but $L \cup L' = \Sigma^*$.

(d) False. If $L \subseteq \Sigma^*$ cannot be accepted by an FA, then L' cannot, but $L \cap L' = \emptyset$.

(e) True.

(f) False. L_2 could be a subset of L_1, for example.

(g) True. If $L_1 \cap L_2$ can be accepted by an FA, so can $L_1 - (L_1 \cap L_2) = L_1 - L_2$. Now $L_2 = (L_1 \cup L_2) - (L_1 - L_2)$. Therefore, if $L_1 \cup L_2$ could be accepted by an FA, L_2 could be also.

(h) False. L_1 could be Σ^*, for example.

(i) False. Every language is the union of one-element languages; for example, $\{a^n b^n \mid n \geq 0\} = \{\Lambda\} \cup \{ab\} \cup \{aabb\} \cup \ldots$.

(j) False. Here is a counterexample. For each $k \geq 1$, let $S_k = \{a^{2^k i} b^{2^k i} \mid i \geq 0\}$. Thus $S_1 = \{\Lambda, a^2b^2, a^4b^4, a^6b^6, \ldots\}$, $S_2 = \{\Lambda, a^4b^4, a^8b^8, \ldots\}$, and so forth. Then $S_{k+1} \subseteq S_k$ for each k. The set $\cap_{k=0}^{\infty} S_k$ is $\{\Lambda\}$, but it is easy to show that for every k, S_k cannot be accepted by an FA. Now let $L_k = S_k'$. It follows that L_k cannot be accepted by an FA and $L_k \subseteq L_{k+1}$, but $\cup_{k=1}^{\infty} L_k = (\cap_{k=1}^{\infty} S_k)' = \Sigma^* - \{\Lambda\}$.

2.30. Suppose $A = \{a^{n+ip} \mid i \geq 0\}$. A finite automaton accepting A can be constructed as follows. Because the alphabet is $\{a\}$, only one transition is needed from each state. If the initial state is q_0, there are additional distinct states $q_1, q_2, \ldots, q_n$ and transitions from q_j to q_{j+1} for $0 \leq j < n$. If $n > 0$, the state q_n is the only one of these states that is accepting. If $p > 0$, there are nonaccepting states $q_{n+1}, q_{n+2}, \ldots, q_{n+p-1}$, a transition from q_j to q_{j+1} for $n \leq j < n + p - 1$, and a transition from q_{n+p-1} back to q_n. (In the case $p = 1$, this means a transition from q_n back to q_n. If $p = 0$, the transition from q_n goes to another nonaccepting state q, from which there is a transition back to q. It is easy to see that this FA accepts A.

2.31(a) If L is accepted by an FA, N is the integer in the pumping lemma, and f is not bounded, then for some n, $f(n) \geq N$. Let $x = a^{f(n)} b^n$, and suppose $x = uvw$ for some strings u, v, and w satisfying the conditions in the pumping lemma. The pumping lemma then says that $a^m b^n \in L$, for some $m > f(n)$, which contradicts the assumption about L.

(b) In this case, L is the union of the two languages L_1 and L_2, where $L_1 = \{b^{2i} \mid i \geq 0\}$ and L_2 is the language of all even-length strings with exactly one a. Both L_1 and L_2 can be accepted by FAs.

(c) We show the result in the case where f is periodic—i.e., where $f(n) = f(n + p)$ for every n. It is sufficient to show that the equivalence relation I_L has only a finite number of equivalence classes, and to do that it is sufficient to find a finite set S of strings so that any string is indistinguishable with respect to L from some element of S. Let M be the maximum value taken by the function f. For each i with $0 \leq i < p$ and each j with $0 \leq j \leq M$, let $x_{i,j} = b^i a^j$, and let $x_0 = a^{M+1}$. Now let x be any string. If $n_a(x) > M$, then xz cannot be in L for any z, and so $x I_L x_0$. Otherwise, let $n_b(x) = i$ and $n_a(x) = j$. Clearly $x I_L b^i a^j$, since the order of the symbols in a string is irrelevant as far as whether the string is in L. Moreover, since $f(n) = f(n + p)$ for every n, the only significant thing about the number of b's is the the value mod p; therefore, $x I_L x_{i_1,j}$, where $i_1 = i \bmod p$.

(d) Suppose L is accepted by an FA, and let n be the integer in the pumping lemma. Consider the string $x = b^m a^{f(m)}$, where m is any integer with $m \geq n$. By the pumping lemma, $x = uvw$, where $|uv| \leq n$, $|v| > 0$, and $uv^i w \in L$ for every $i \geq 0$. Since $m \geq n$,

$v = b^p$ for some $p > 0$. This means that each of the strings $b^{m+ip}a^{f(m}$ is in L. It follows from the definition of L that $f(m + ip) = f(m)$ for every $i \geq 0$. To summarize: for every $m \geq n$, there is a number p_m so that $0 < p_m \leq n$ and $f(m + ip) = f(m)$ for every $i \geq 0$. Now there are only a finite number of integers p that can be p_m for some $m \geq n$, since for any such p_m, $p_m \leq n$. Let P be the least common multiple of all of them—i.e., the smallest positive integer that is evenly divisible by all of them. Then for any $m \geq n$, $P = kp_m$ for some k. Therefore, for any $i \geq 0$, $f(m + iP) = f(m + (ik)p_m) = f(m)$. Therefore, f is eventually periodic.

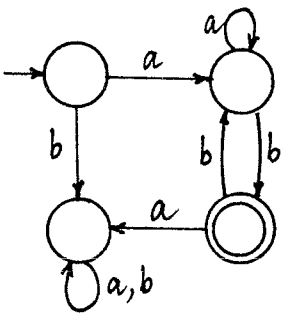
2.32. If there is only one equivalence class, then either $L = \emptyset$ or $L = \{a, b\}^*$.

2.33. There are $|x| + 2$ equivalence classes, one for each prefix of x and one containing all the nonprefixes (see Exercise 2.3(b)).

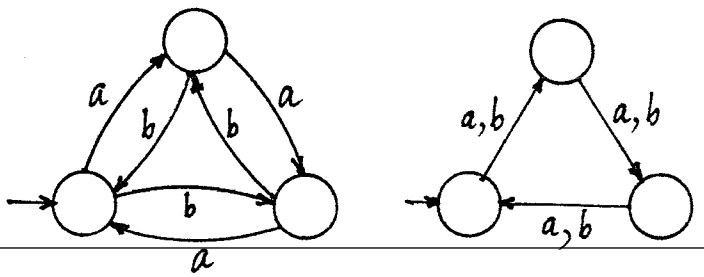
2.34. It is enough to show that for every x and every y in S, x and y are equivalent, and no element of S is equivalent to an element of S'. If $x, y \in S$, then neither is a prefix of any element of L, and therefore xI_Ly. If $x \in S$, $y \notin S$, and z is a string for which $yz \in L$, then z distinguishes x from y.

2.35. Suppose $xI_L\Lambda$ and $x \neq \Lambda$. Then for every $i \geq 1$, either both the strings x^iy and xx^iy are in L or neither is, which means that $x^iyI_Lx^{i+1}y$. Since $[\Lambda]$ contains all the strings x^i, it must be infinite. (Note: the same argument shows that if there are strings x and xy for which xI_Lxy and $y \neq \Lambda$, then $[x]$ is infinite.)

2.36.

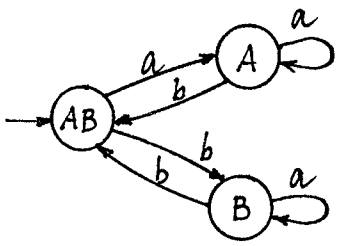


2.37. If the three equivalence classes are $[a]$, $[aa]$, and $[aaa]$, then $\Lambda \in [aaa]$, and if the three equivalence classes are also $[b]$, $[bb]$, and $[bbb]$, then $[aaa] = [bbb]$ for the same reason. It is easy to see that there are only two possible ways to draw the transitions of the FA:

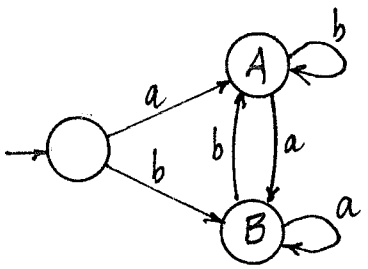


In each case, the only constraint involving the accepting states is that there must be either one or two. Therefore, there are six possible FAs for each of the two diagrams, for a total of twelve.

2.38(b) Denote the three sets by AB, A, and B, respectively. The transition diagram is shown below, with no states yet designated as accepting. By an argument similar to that in the solution to Exercise 2.38(a), there are four languages satisfying the required condition: B, A, $B \cup AB$, and $A \cup AB$.



2.38(c) The transition diagram is shown below. If A and B denote the equivalence classes containing a and b, respectively, the languages we want are A, B, $A \cup \{\Lambda\}$, and $B \cup \{\Lambda\}$.



2.39. The partition is the same. The sets $\{\Lambda\}$ and $\{a^i b^i \mid i > 0\}$ are both subsets in the partition, just as before. The only difference is that before these were $\{\Lambda\}$ and L, now they are $\{\Lambda\}$ and $L - \{\Lambda\}$.

2.40(a) and (b) Consider a string $x \in \{a, b\}^*$, and let $k = n_a(x) - n_b(x)$. Saying that $k = 0$ is the same as saying that $x \in L$. If $k > 0$, then x has an excess of k a's, so that for any z, $xz \in L$ if and only if z has an excess of k b's. Similarly, if $k < 0$, $xz \in L$ if and only if z has an excess of $-k$ a's. This means that if $n_a(x) - n_b(x) = n_a(y) - n_b(y)$, then x and y have the property that xz and yz will be in L for precisely the same strings z—i.e., x and y are equivalent. Conversely, if $n_a(x) - n_b(x) \neq n_a(y) - n_b(y)$, then any string z for which $n_b(z) - n_a(z) = n_a(x) - n_b(x)$ distinguishes x from y.

(c) Parts (a) and (b) say that an equivalence class containing a string x is determined by the difference $n_a(x) - n_b(x)$: another string y will be in this equivalence class precisely if $n_a(y) - n_b(y)$ is the same value as for x. This means that for every integer k, there is an equivalence class containing all those strings x for which the difference is k. Another way to say this is to say that the equivalence classes are

$$\ldots, [bbb], [bb], [b], [\Lambda] = L, [a], [aa], [aaa], \ldots$$

For each $k > 0$, $[b^k] = \{x \mid n_b(x) - n_a x) = k\}$, and $[a^k] = \{x \mid n_a(x) - n_b(x) = k\}$.

2.41. Suppose x and y are L_1-distinguishable, and specifically that for some z, $xz \in L_1$ and $yz \notin L_1$. Then for some w, $xzw \in L$, since xz is a prefix of an element of L; and $yzw \notin L$, since yz is not a prefix of an element of L. Therefore, if x and y are in different equivalence classes with respect to I_{L_1}, they are in different equivalence classes with respect to I_L. This means that the partition determined by I_L is at least as fine as that determined by I_{L_1}: Any equivalence class with respect to I_{L_1} is the union of equivalence classes with respect to I_L.

2.43. An argument similar to the one in Example 2.27 shows that the strings in $\{a, b\}^*$ are pairwise L-distinguishable, so that every equivalence class has exactly one element.

2.44. According to Exercise 2.34, one equivalence class is the set of all strings that are not prefixes of any element of L. These are the strings having a prefix with more right parentheses than left. Of the strings that are prefixes of balanced strings of parentheses, the equivalence class is determined by the number of excess left parentheses. (If x_1 and x_2 are both prefixes of balanced strings, and x_1 has k more left parentheses than right, and x_2 has j more left parentheses than right, then x_1 and x_2 are equivalent if $k = j$, and otherwise the string $)^k$ distinguishes x_1 and x_2 relative to L.)

This means that the equivalence classes are N, $L = [\Lambda] = [(^0]$, $[(]$, $[(^2]$, $[(^3]$, ..., where N is the set of nonprefixes of elements of L and $[(^k]$ is the set of all strings of parentheses that are prefixes of elements of L and have k more left parentheses than right.

2.45. $\{\Lambda\}$, L, and the set of strings that are not prefixes of any element of L are three of the equivalence classes. We can describe the general form of all the other equivalence classes by considering a specific example. Consider a string that is a prefix of an element of L, in which there are 3 unmatched left parentheses (parentheses without the matching right parentheses). Then there are sixteen "archetypal" pairwise inequivalent strings of this type: $(((, (((i, (((i+, (((i+i, ((i+(, ((i+(i, ((i+(i+, ((i+(i+i, (i+((, (i+((i, (i+((i+, (i+((i+i, (i+(i+(, (i+(i+(i, (i+(i+(i+, and $(i+(i+(i+i$. It is not hard to see that any two of these are distinguishable with respect to L. (For example, consider the two strings $x = ((i + (i$ and $y = (i + ((i + i$. Let $z = +i)) + i)$. Then $xz \in L$ and $yz \notin L$.) Furthermore, every other string that is a prefix of an element of L and that has exactly three unmatched left parentheses is equivalent to one of these, and looks like one of these except that one or more of the i's may be replaced by other elements of L. For example, a string in the same equivalence class as $((i + (i+$ is $(((i + i) + (((i + i) + i)+$. (The first i has been replaced by $(i + i)$ and the second by $((i + i) + i)$.)

We can enumerate these sixteen as follows. For each of the first two unmatched left parentheses, there are two possibilities: either it is followed immediately by another un-

matched left parenthesis, or it is followed immediately by an element of L and a $+$. For the third unmatched left parenthesis, there are four possibilities: there is nothing after it; it is followed by an element of L and nothing else; it is followed by an element of L and a $+$; or it is followed by an element of L, a $+$, and another element of L. Altogether, then, there are $2 * 2 * 4 = 16$ combinations.

In general, for each $n > 0$, there are $2^{n-1} * 4$ equivalence classes, which can be enumerated by considering the 2 possibilities for each of the first $n - 1$ unmatched parentheses and the 4 possibilities for the last.

2.46(a) For x and y to be distinguishable with respect to $\{a, b\}^*\{aba\}$, one of the two strings must end with a longer prefix of aba than the other. In this case, if the longer prefix is α, and $\alpha\beta = aba$, the string β distinguishes the two strings. Therefore, the longest string that might be necessary is one of length 2. (Another way to say it is that if a string of length 3 was required to distinguish x and y, they wouldn't be distinguishable at all, since adding a string of length 3 or more would have the same effect in both cases.)

(b) If x has i more left parentheses than right, and y has j more left than right, then saying x and y are L-distinguishable means that $i \neq j$. If k is the smaller of the two numbers i and j, a string of k right parentheses is the shortest string distinguishing x and y. Since i can be no larger than m, and j no larger than n, the smaller of i and j can be no larger than the smaller of m and n.

2.47(a) Every equivalence class has exactly one element.

(b) Every equivalence class has exactly two elements, except the one containing Λ, which has only one. For any $x \neq \Lambda$, x and $x^\sim$ are equivalent, but no other string is equivalent to either of these.

2.49. Suppose R is a right invariant equivalence relation on Σ^* so that the set of equivalence classes of R is finite and L is the union of some of the equivalence classes. If xRy, then for any z, $xzRyz$, since R is right invariant. This means that xz and yz are in the same equivalence class with respect to R. But since L is the union of some of the equivalence classes, every equivalence class that intersects L must be completely within L; therefore, for every z, $xz \in L$ if and only if $yz \in L$. Therefore, xI_Ly. This means that the partition of Σ^* determined by R is finer than that determined by I_L (every equivalence class with respect to I_L is the union of equivalence classes with respect to R). Since the number of equivalence classes with respect to R is finite, the number of equivalence classes with respect to I_L must be finite.

2.50(a) Suppose on the one hand that there is a right invariant partition of $\{a, b\}^*$, so that S is one of the subsets in the partition. Right invariant means that whenever x and y are in the same subset, then for any z, xz and yz are in the same subset. Therefore, if x and y are both in S, then for any z, xz and yz are in the same subset of the partition, so that if one of the two is in S, the other must be also.

On the other hand, suppose S has the property that for any $x, y \in S$ and any z, xz and yz are either both in S or both not in S. We consider the following sets: first, all the one-element sets $\{x\}$, where no prefix of x is in S; second, the set S; finally, all sets of the form $S\{\alpha\}$, where $S\{\alpha\} \not\subseteq S$ and $S\{\gamma\} \subseteq S$ for every proper prefix γ of α. (We summarize this last condition by saying that α is a *minimal* string so that $S\{\alpha\} \not\subseteq S$.)

A useful observation is that if $x \in S\{\alpha\}$ and $x \in S\{\beta\}$, where $S\{\alpha\} \not\subseteq S$, $S\{\beta\} \not\subseteq S$, and α and β are both minimal strings with this property, then $\alpha = \beta$. To see this, suppose $\alpha \neq \beta$ and $x = s_1\alpha = s_2\beta$, where $s_1, s_2 \in S$. The strings α and β can't be the same length; suppose $|\alpha| > |\beta|$. Then $\alpha = \gamma\beta$, for some γ, so that $s_1\gamma = s_2$. Because of the assumption on S, it follows that $S\{\gamma\} \subseteq S$, so that α can't be minimal.

Now we check that these subsets of Σ^* form a partition. Obviously, if x is a string such that no prefix of x is in S, then x cannot be in S or any of the sets $S\{\alpha\}$. If $S\{\alpha\} \not\subseteq S$, then $S\{\alpha\}$ and S must be disjoint, because of the assumption on S. Finally, as we observed in the previous paragraph, if $S\{\alpha\} \cap S\{\beta\} \neq \emptyset$, $S\{\alpha\} \not\subseteq S$, $S\{\beta\} \not\subseteq S$, and α and β are both minimal, then $\alpha = \beta$.

Finally, we check that this partition is right invariant. Because of the assumption on S, it is sufficient to show that if $x, y \in S\{\alpha\}$, and $xz \in S\{\beta\}$, where α and β are minimal as defined above, then yz is also in $S\{\beta\}$. We know that $x = s_1\alpha$ and $y = s_2\alpha$, for some $s_1, s_2 \in S$; therefore, xz and yz are both in $S\{\alpha z\}$. Let γ be the longest prefix of αz satisfying $S\{\gamma\} \subseteq S$. Then our previous discussion shows that $\alpha z = \gamma\beta$, so that $S\{\alpha z\} \subseteq S\{\beta\}$, and $y \in S\{\beta\}$.

(b) In the case where S is finite, the condition reduces to saying that no element of S is a prefix of any other element of S.

(c) If S is finite and satisfies this condition, let M be the maximum length of elements of S. We can consider the following partition of Σ^*: first, all the one-element sets $\{x\}$, where $|x| \leq M$ and no prefix of x is in S; second, the set S; third, the single set T containing all other strings. This is obviously a finite partition. It is right invariant, because of the assumption on S and the fact that for any $x \in T$ and any string y, $xy \in T$.

(d) If S satisfies the condition in (a), then S is one of the subsets of a finite right invariant partition if and only if S can be accepted by an FA. On the one hand, if P is a finite right invariant partition, then there is an FA M so that the states of M correspond to the subsets in P. On the other hand, if M is a minimum-state FA accepting S, then the subsets L_p, for the states p of M, form a finite right invariant partition, and L is the union of all the L_p's for which p is accepting. However, the condition in A implies (see the discussion on page 74) that $p \equiv q$ for any two accepting states p and q; therefore, since M is minimal, there is only one accepting state.

2.51. Suppose $M = (Q, \Sigma, q_0, A, \delta)$ is an FA accepting L_1. Then for a string $x \in \Sigma^*$, $x \in L_1/L_2$ if and only if there is a string $y \in L_2$ with $xy \in L$; and this is true if and only if there is a string $y \in L_2$ so that $\delta^*(q_0, xy) = \delta^*(\delta^*(q_0, x), y) \in A$. With this in mind, we define $B = \{q \in Q \mid \text{ for some } y \in L_2, \delta^*(q, y) \in A\}$. Then we have observed that for any x, $x \in L_1/L_2$ if and only if $\delta^*(q_0, x) \in B$. In other words, the FA $M' = (Q, \Sigma, q_0, B, \delta)$ accepts the language L_1/L_2.

Notice that the argument does not really use the fact that L_2 is accepted by an FA. However, if this is not the case there may be no way to determine exactly which states are in the set B. If it is, on the other hand, there is an algorithm to do this. We do not present all the details, but the idea is that for $q \in Q$, we can determine for each $r \in A$ the language $L(q,r) = \{x \in \Sigma^* \mid \delta^*(q,x) = r\}$, and we can then determine whether the intersection $L(q,r) \cap L_2$ is nonempty.

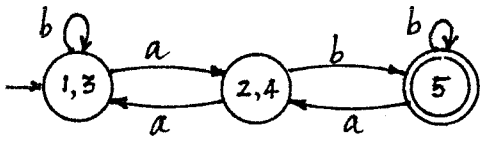
2.52. Using the paintcan analogy, we observe that with p distinct primary colors, there are 2^p possible combinations that can be used to create distinct colors, since there are 2^p subsets of a set of p elements. If we want the number of distinct colors to be at least n, then 2^p must be at least n, so that $p \geq \log_2 n$. We conclude that in order to distinguish all possible pairs from among n strings, at least $\log_2 n$ strings are required.

2.53. For $z = a_1 a_2 \ldots a_n$, let $p_i = \delta^*(p, a_1 \ldots a_i)$ and $q_i = \delta^*(q, a_1 \ldots a_i)$, for each i with $1 \leq i \leq n$. We observe that if z has the property we want (that is, exactly one of the two states $\delta^*(p,z)$ and $\delta^*(q,z)$ is in A), and if the pairs (p_i, q_i) and $(p_{i+d} q_{i+d})$ are identical, then we can shorten z by omitting the substring $a_{i+1} a_{i+2} a_{i+d}$, and the shortened string will still have the desired property. Therefore, we may assume that all the pairs (p_i, q_i) are distinct, for $1 \leq i \leq n$, and thus that n is no larger than the maximum number of distinct ordered pairs of states, which is s^2 (where $s = |Q|$).

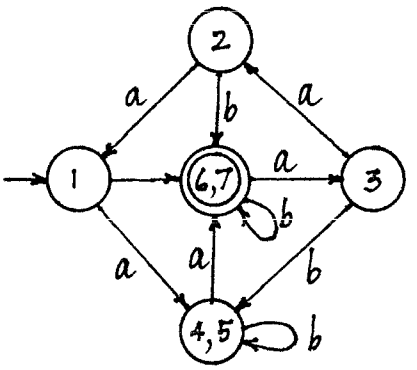
2.54. If L is accepted by an s-state FA M, then any two L-distinguishable strings x and y can be distinguished by a string of length s^2 or less. (We let $p = \delta^*(q_0, x)$ and $q = \delta^*(q_0, y)$ and use the result of Exercise 2.53.) On the other hand, if there is an n so that any two L-distinguishable strings can be distinguished by a string of length n or less, then L can be accepted by an FA. (According to Exercise 2.52, using only the finite set S of strings of length $\leq n$, it is impossible to have an infinite set of strings, any two of which can be distinguished by an element of S.)

2.55. The FAs in parts (b) and (e) are already minimal.

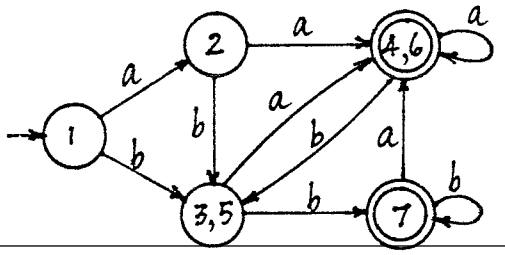
(a)



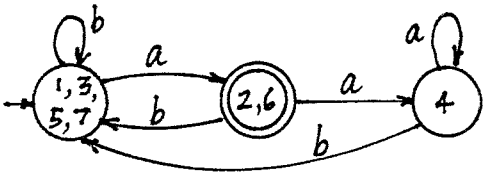
(c)



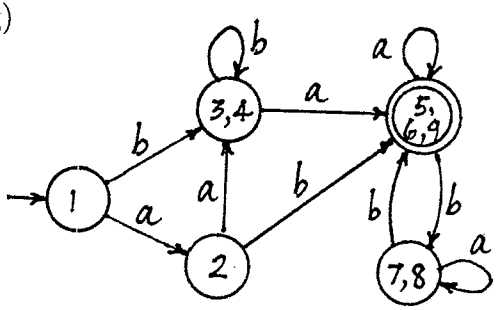
(d)



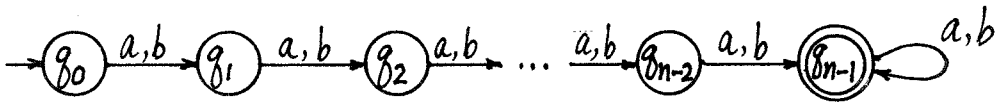
(f)



(g)



2.56(a) Suppose the FA has n states. For the FA shown below, the only pair marked on the first pass is (q_{n-2}, q_{n-1}). If starting with pass 2 the pairs are considered in the order (q_0, q_1), (q_1, q_2), $\ldots$, (q_{n-3}, q_{n-2}), then one pair is marked on each pass. The total number of passes is $n - 1$, and no order would require more.



(b) For a pair (p, q) that is ultimately marked, let $n_{(p,q)}$ be the length of the shortest string z for which $\delta^*(p, z) \in A$ and $\delta^*(q, z) \notin A$ (or vice versa). If we process the pairs in nondecreasing order of $n_{(p,q)}$ (i.e., first all the pairs (p, q) with $n_{(p,q)} = 1$, then all the pairs (p, q) with $n_{(p,q)} = 2$, and so forth), then every pair that will ever be marked will be marked on the second pass.

2.57(a) No. For $i \geq 0$, let $x_i = 01^i$. Then for any i and j with $i < j$, the string ab^i distinguishes x_i and x_j, since ab^jab^i has no nonnull prefix of the form ww.

(b) Yes. For any x, $x \in L$ if and only if x contains one of the substrings aa, bb, $abab$, or $baba$. (Reason: if x contains neither aa nor bb, then x must consist of alternating b's and a's; therefore, if x contains a nonnull substring ww, it must contain either $abab$ or $baba$.)

(c) No. We use the pumping lemma to show L' is nonregular, and it will follow that L is also. Suppose L' is regular, let n be the integer in the pumping lemma, and let x be a string in L' with $|x| \geq n$. (The fact that there is such an x follows from the parenthetical statement in the exercise.) Then $x = uvw$ for some strings u, v, w such that $|v| > 0$ and $uv^iw \in L'$ for every $i \geq 0$. In particular, $uv^3w \in L'$. However, this string contains three consecutive occurrences of v and is therefore an element of L by definition.

(d) No. This can easily be proved using the pumping lemma, starting with a string of the form b^nab^n.

(e) No. Let $S = \{a^{2i} \mid i \geq 0\}$. Then if $0 \leq i < j$, the strings a^{2i} and a^{2j} are distinguished by b^{2i}, because the middle two symbols of $a^{2i}b^{2i}$ are unequal and the middle two symbols of $a^{2j}b^{2i}$ are both a. S is therefore an infinite set of L-distinguishable strings.

(f) No. Suppose L is accepted by an FA, and let n be the integer in the pumping lemma. Let $x = a^nb^na^nb^n = (a^nb^n)\Lambda(a^nb^n)$. Then $x = uvw$, where u, v, and w satisfy the usual three conditions. As usual, v must consist of one or more a's from the first group of a's,

say $v = a^j$. Then $uv^2w = a^{n+j}b^n a^n b^n$, and it is easy to see that this string is not of the form xyx for any x with $|x| \geq 1$.

(g) No, because the set of palindromes cannot be accepted by an FA.

(h) No. If L could be accepted by an FA, then $\{a^{n^2} \mid n \geq 0\}$ could also, but we know from Example 2.32 that it cannot.

(i) Yes. There is an FA with 6 states accepting L. Five of the states correspond to the five possible values of $n_a(x) - n_b(x)$ from -2 to 2, and the sixth is the "dead" state for strings not in L.

(j) No. Any two of the strings b, b^2, b^3, ..., say b^i and b^j (where $i < j$), can be distinguished with respect to L by the string a^{i+3}.

(k) Yes. One way to construct an FA is to have states corresponding to the possible pairs ($n_a(x)$ mod 5, $n_b(x)$ mod 5), 25 states in all. The initial state, which is the only accepting state, is $(0, 0)$.

(l) No. For each $i \geq 1$, let $x_i = a^{p_i}$, where p_i is the ith prime ($p_1 = 2$, $p_2 = 3$, ...). If $i \neq j$, then x_i and x_j are distinguished by the string b^{p_i}.

(m) Yes. Let $M = (Q, \{a, b\}, q_0, A, \delta)$ be an FA accepting L, and let $M_1 = (Q, \{a, b\}, q_0, A_1, \delta)$, where A_1 is the set of states q in A for which there is no string z satisfying $\delta^*(q, z) \in A$. Then M_1 accepts Max(L).

(n) Yes. Let $M = (Q, \{a, b\}, q_0, A, \delta)$ be an FA accepting L, and let $M_1 = (Q, \{a, b\}, q_0, A_1, \delta)$, where A_1 is the set of states q in A for which there do not exist strings w and z satisfying $|z| > 0$, $\delta^*(q_0, w) \in A$, and $\delta^*(q_0, wz) = q$. In other words, A_1 is the set of states q in A for which no path from q_0 to q reaches an accepting state until the last step. Then M_1 accepts Min(L).

2.58. One example is the language $L = AEqB$. L cannot be accepted by an FA and $L^* = L$.

2.59. One example is *Pal*, the set of palindromes over $\{a, b\}$. For this L, $L^* = \{a, b\}^*$, since Λ and all strings of length 1 are palindromes.

2.62(a) The relation is reflexive, since for any FA $M = (Q, \Sigma, q_0, A, \delta)$, the identity function $f : Q \to Q$ defined by $f(q) = q$ is an isomorphism from M to itself. Suppose that for $1 \leq k \leq 3$, $M_k = (Q_k, \Sigma, q_k, A_k, \delta_k)$, and $i : Q_1 \to Q_2$ and $j : Q_2 \to Q_3$ are isomorphisms from M_1 to M_2 and from M_2 to M_3, respectively. We construct isomorphisms from M_2 to M_1 and from M_1 to M_3. It will follow that the relation is both symmetric and transitive.

Since i and j are bijections, $i^{-1} : Q_2 \to Q_1$ and $j \circ i : Q_1 \to Q_3$ are also bijections. For any $p \in Q_2$ and $a \in \Sigma$, $i(i^{-1}(\delta_2(p, a))) = \delta_2(p, a)$, by definition of the function i^{-1}. But since i is an isomorphism from M_1 to M_2, we also have $i(\delta_1(i^{-1}(p), a)) = \delta_2(i(i^{-1}(p)), a) = \delta_2(p, a)$. Since i is one-to-one, $i^{-1}(\delta_2(p, a)) = \delta_1(i^{-1}(p), a)$. This is the formula i^{-1} needs to satisfy in order for it to be an isomorphism from M_2 to M_1. In addition, since for any $q \in Q_1$, q is an accepting state if and only if $i(q)$ is, it is also true that for any $p \in Q_2$, p is an accepting state if and only if $i^{-1}(p)$ is. Finally, since $i(q_1) = q_2$, $i^{-1}(q_2) = q_1$. Therefore, i^{-1} is an isomorphism from M_2 to M_1.

Now we show that $j \circ i$ is an isomorphism from M_1 to M_3.

$$j \circ i(\delta_1(q,a)) = j(i(\delta_1(q,a))) = j(\delta_2(i(q),a)) = \delta_3(j(i(q)),a) = \delta_3(j \circ i(q),a)$$

The second equality holds because i is an isomorphism, the third because j is. It is easy to check that for any $q \in Q_1$, q is an accepting state if and only if $j \circ i(q)$ is, and clearly $j \circ i(q_1) = q_3$.

(b) The proof is by structural induction on x. First, $i(\delta_1^*(q,\Lambda)) = i(q) = \delta_2^*(i(q),\Lambda)$. Now suppose y is a string for which $i(\delta_1^*(q,y)) = \delta^*(i(q),y)$ for every q. Then for any $a \in \Sigma$,

$$i(\delta_1^*(q,ya)) = i(\delta_1(\delta_1^*(q,y),a)) = \delta_2(i(\delta_1^*(q,y)),a) = \delta_2(\delta_2^*(i(q),y),a) = \delta_2^*(i(q),ya)$$

The first equality holds by the definition of δ_1^*; the second holds because i is an isomorphism; the third follows from the induction hypothesis; and the last uses the definition of δ_2^*.

(c) Suppose $i : M_1 \to M_2$ is an isomorphism. A string x is accepted by M_1 if and only if $\delta_1^*(q_1,x) \in A_1$. This is true if and only if $i(\delta_1^*(q_1,x)) \in A_2$, and by (b), this is true if and only if $\delta_2^*(i(q_1),x)) = \delta_2^*(q_2,x) \in A_2$. Therefore, x is accepted by M_1 if and only if x is accepted by M_2.

(d) Two. There is only one way to draw the transitions. Two nonisomorphic FAs are obtained by making the state an accepting state and a nonaccepting state, respectively.

(e) Suppose the two states are q_0 and q_1, with q_0 the initial state. In order to complete the transition diagram, we must decide three things: how to draw the transitions from q_0, how to draw the transitions from q_1, and which states to designate as accepting states. Since we require both states to be reachable from q_0, there are three ways to draw the transitions from q_0. There are four ways to draw the transitions from q_1; and since there is to be at least one accepting state, there are three ways of designating accepting states. The total number of transition diagrams is $3*4*3 = 36$, and it is easy to see that any two of these represent nonisomorphic FAs.

(f) All the FA's in which both states are accepting accept the language $\{a,b\}^*$. However, any two of the remaining 24 accept different languages. This can be seen as follows. Let M_1 and M_2 be the two FAs, with transition functions δ_1 and δ_2, respectively. If q_0 is an accepting state in one and not the other, then Λ is in one but not both of the two languages; thus we assume that M_1 and M_2 have exactly the same accepting states. If $\delta_1(q_0,a) \neq \delta_2(q_0,a)$, then a is in one of the languages but not the other, and similarly if $\delta_1(q_0,b) \neq \delta_2(q_0,b)$. Finally, if the transitions from the accepting state are the same in M_1 and M_2, and $\delta_1(q_1,a) \neq \delta_2(q_1,a)$, then there is a string with second symbol a that is in one but not both of the languages, and similarly if $\delta_1(q_1,b) \neq \delta_2(q_1,b)$. The conclusion is that the 36 nonisomorphic FAs in (e) accept 25 distinct languages.

2.63. As is pointed out in the statement of the problem, each state in either FA corresponds to one of the equivalence classes of I_L. This means that the L_q partition determined by one of the FA's is the same as that determined by the other. So the bijection $i : Q_1 \to Q_2$ is described as follows: if $q \in Q_1$, and $\delta_1^*(q_1,x) = q$, then $i(q) = \delta_2^*(q_2,x)$.

We must show these three things:
(i) $i(q_1) = q_2$; (ii) For any $q \in Q_1$, $q \in A_1$ if and only if $i(q) \in A_2$; and (iii) For any $q \in Q_1$ and any $a \in \Sigma$, $i(\delta_1(q,a)) = \delta_2(i(q),a)$.

Statement (i) follows from our definition of i and the fact that $\delta_1^*(q_1, \Lambda) = q_1 = \delta_2^*(q_2, \Lambda)$. To show (ii), we take a state $q \in A_1$ and a string x such that $\delta_1^*(q_1, x) = q$. Then $\delta_2^*(q_2, x) = i(q) \in A_2$, because the string x is in L (since M_1 accepts L). The argument is also reversible: If $i(q) \in A_2$, then $x \in L$ because M_2 accepts L, so that $q \in A_1$. Finally, for $q \in Q_1$ and $a \in \Sigma$, if $\delta_1^*(q_1, x) = q$, then

$$i(\delta_1(q, a)) = i(\delta_1^*(q_1, xa)) = \delta_2^*(q_2, xa) = \delta_2(\delta_2^*(q_2, x), a) = \delta_2(i(q), a)$$

2.64. If the unreachable states are deleted from both FAs, and the minimization algorithm is applied to both, the two original FAs accept the same languages if and only if the resulting minimum-state FAs are isomorphic.