Chapter 3

Regular Expressions and Finite Automata

3.1. (a) 001 or 011 (b) 0101 or 1010 (c) 110 (d) 0110

3.2. (a) 00 (b) 01 (c) 0 (d) The shortest one is 010.

3.3. (a) $(r+s)^*$ (b) $r(r+s)^*$ (c) r^* (d) r^* (e) $(r+s)^*$

3.4. First observe that the only difference between $(111^*)^*$ and 111^* is that the first allows the string Λ. In other words, strings corresponding to $(111^*)^*$ are Λ and all strings of two or more 1's. Therefore, the formula follows from the fact that any string of two or more 1's can be formed by concatenating copies of 11 and/or 111—i.e., that any integer $n \geq 2$ can be expressed as $2i + 3j$ for some $i, j \geq 0$. The proof of this fact is similar to the argument in Exercise 2.50.

3.5. The string Λ corresponds to both expressions, and it is easy to see that any nonnull string corresponding to the first one must start with a and end in b. Therefore, it is sufficient to show that every string of the form $x = ayb$ corresponds to the first regular expression. Let us show this by induction on the number of a's in x that are not immediately preceded by a. If there is only one such a, then x does not contain the substring ba, and it must therefore match the regular expression aa^*bb^*. Suppose $k \geq 1$, and every string of the form ayb having k a's not immediately preceded by a matches the first regular expression. Now let $x = ayb$, and suppose x contains $k+1$ a's that are not immediately preceded by a. The first is the one at the beginning of x; consider the second such a, and let x_1 be the suffix of x beginning with this a. Then the prefix of x preceding x_1 must be of the form aa^*bb^*. The induction hypothesis tells us that x_1 matches the regular expression $(aa^*bb^*)^*$, and so the entire string x must match it also.

3.6. $\emptyset^* = \{\Lambda\}$.

3.7. (a) The set of languages that are subsets of $\Sigma \cup \{\Lambda\}$. This can also be described as the set of languages L over Σ for which every element of L is a string of length 0 or 1. If $|\Sigma| = k$, there are 2^{k+1} such languages.
(b) The set of all languages over Σ which either are empty or contain exactly one string. There are infinitely many such languages.
(c) The set of all languages that are $\emptyset$ or $\{\Lambda\}$ or of the form $\{a\}^*$, where $a \in \Sigma$. There are exactly $|\Sigma| + 2$ such languages.
(d) The set of all finite languages over Σ, of which there are infinitely many.
(e) The (finite) set of all languages of the form

$$\Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_k \cup \Sigma_{k+1}^* \cup \Sigma_{k+2}^* \cup \ldots \Sigma_{k+j}^*$$

where each Σ_i is a subset of $\Sigma \cup \{\Lambda\}$. (k and/or j may be 0; if both are 0, the language is $\emptyset$.) For $\Sigma = \{a, b\}$, this includes the languages $\emptyset$, $\{\Lambda\}$, $\{a\}$, $\{b\}$, $\{a, b\}$, $\{a, \Lambda\}$, $\{b, \Lambda\}$, $\{a, b, \Lambda\}$, $\{a\}^*$, $\{b\}^*$, $\{a\}^* \cup \{b\}^*$, $\{a, b\}^*$, $\{a\}^* \cup \{b\}$, and $\{b\}^* \cup \{a\}$.

3.8. (a) $(001)^*(11)^*$ (b) $(001)^*0(001 + 11)^*$ (c) $(001 + 11)^*(0 + \Lambda)$

3.9. (a) $1^*01^*01^*$
(b) The most obvious solutions are those of the form $A0B0C$, where each of A, B, C is either 1^* or $(0 + 1)^*$, and at least one of the three is $(0 + 1)^*$.
(c) $\Lambda + 1 + (0 + 1)^*0 + (0 + 1)^*11$
(d) $(00 + 11)(0 + 1)^* + (0 + 1)^*(00 + 11)$
(e) Two answers are $(1 + 01)^*(\Lambda + 0)$ and $(\Lambda + 0)(1 + 10)^*$.
(f) $1^*(01^*01^*)^*$
(g) The regular expression $r = (1 + 01)^*$ corresponds to the set of strings that don't end with 0 and don't contain 00, and $s = (1 + 10)^*$ to the set of strings that don't begin with 0 and don't contain 00. In a string with exactly one occurrence of 00, then the strings before and after 00 correspond to r and s, respectively. Therefore, one answer is $(1+01)^*(0+\Lambda)+(1+01)^*00(1+10)^*$. A more concise answer is $(1+01)^*(\Lambda+0+00)(1+10)^*$.
(h) $1^*(011^+)^*$
(i) $(0 + 1)^*(11(0 + 1)^*010 + 010(0 + 1)^*11)(0 + 1)^*$

3.10. (a) All strings with an odd number of 1's.
(b) All strings whose length is a multiple of 3, or 1 plus a multiple of 3.
(c) All strings not containing any substring of the form $00x11$.
(d) All strings containing both 10 and 01.

3.13. (b) We may define *rrev* recursively on the set of regular expressions as follows:

$$rrev(\emptyset) = \emptyset; \qquad rrev(\Lambda) = \Lambda; \qquad \text{for } a \in \Sigma,\ rrev(a) = a$$

and for arbitrary regular expressions r and s,

$$rrev((r+s)) = (rrev(r)+rrev(s)); \quad rrev((rs)) = (rrev(s)rrev(r)); \quad rrev((r^*)) = ((rrev(r))^*)$$

Now we can show by structural induction that for any regular expression r, if $L(r)$ is the corresponding language, $rrev(r)$ is the regular expression corresponding to $rev(L(r))$. This is clearly true for the regular expressions $\emptyset$, Λ, and a, for $a \in \Sigma$. Suppose it is true for the regular expressions r and s. We show it is true for (rs), and the argument is similar for the other two cases. $rrev((rs)) = (rrev(s)rrev(r))$, by definition of *rrev*, and by Exercise 2.59a, $rev(L(r)L(s)) = rev(L(s))rev(L(r))$. According to the induction hypothesis, $rrev(s)$ corresponds to the language $rev(L(s))$, and similarly for r. Since the language corresponding to the concatenation of two regular expressions is the concatenation of the two languages, we have the desired result in this case.

3.14. One expression is

$$(\Lambda + a + m)(d^+ + d^*pd^+ + d^+pd^*)(\Lambda + (E + e)(\Lambda + a + m)d^+)$$

3.15. (a) 2 (b) 3

3.16. (a) $\Lambda + aaa^*$ (b) $\Lambda + aaaa^*$

3.17. (a) The strings corresponding to each state are as follows. (It's a little easier to start from the end.)

V. All strings containing $aaba$.
IV. All strings ending in aab but not containing $aaba$.
III. All strings ending in aa but not containing $aaba$.
II. All strings ending in a but not ending in aa and not containing $aaba$.
I. All strings not ending in a or aab and not containing $aaba$.

(b)

V. All strings ending with $aaba$.
IV. All strings ending with aab.
III. All strings ending with aa.
II. All strings ending with a but not with either aa or $aaba$.
I. All strings ending with neither a nor aab.

(c)

V. All strings beginning with $aaba$.
IV. Only the string aab.
III. Only the string aa.
II. Only the string a.
I. Only the string Λ.
VI. All strings that don't begin with $aaba$, except for the strings Λ, a, aa, and aab.

(d)

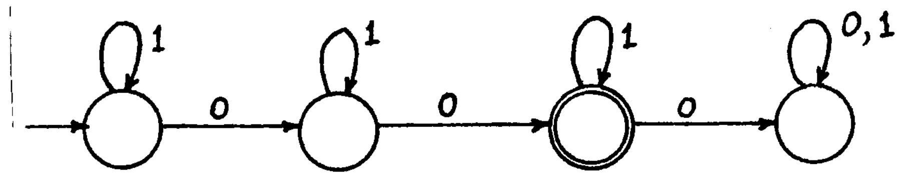
I. Λ only.
II. All strings that start and end with a.
III. All strings that start with a and end with b.
IV. All strings that start with b.

(d)

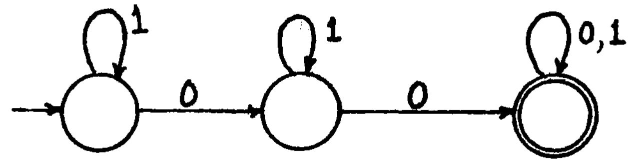
I. $(ab + ba)^*$
II. $(ab + ba)^*a$
III. $(ab + ba)^*b$
IV. $(ab + ba)^*(aa + bb)(a + b)^*$

3.18. If $|x| = n$, there is an FA with $|x| + 2$ 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.

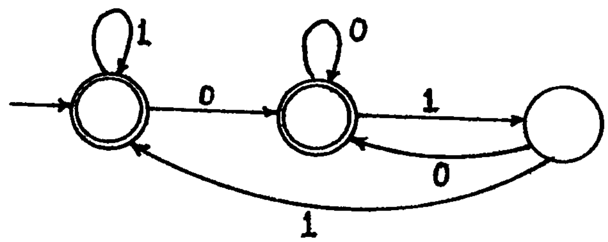
3.19 (a)



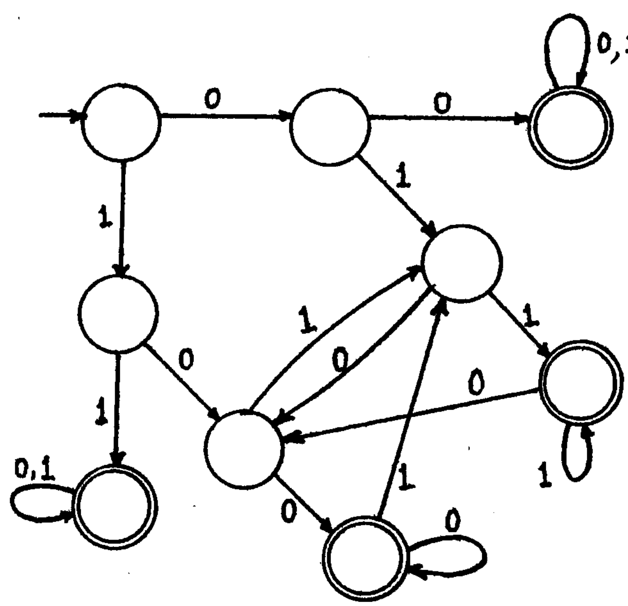
(b)



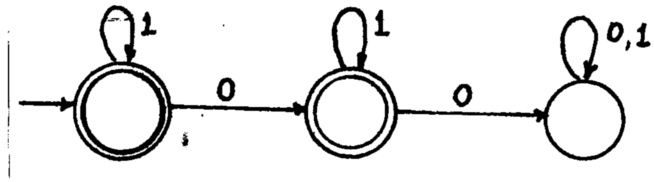
(c)



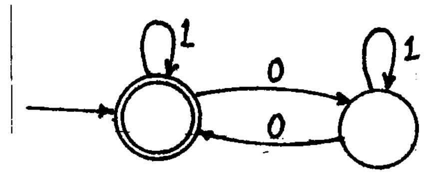
(d)



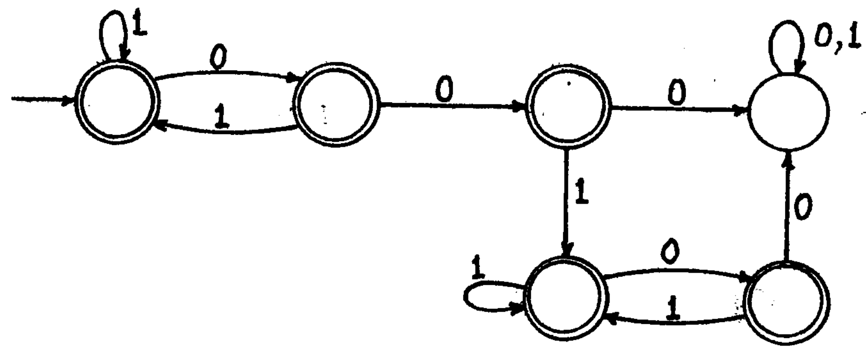
3.19 (e)



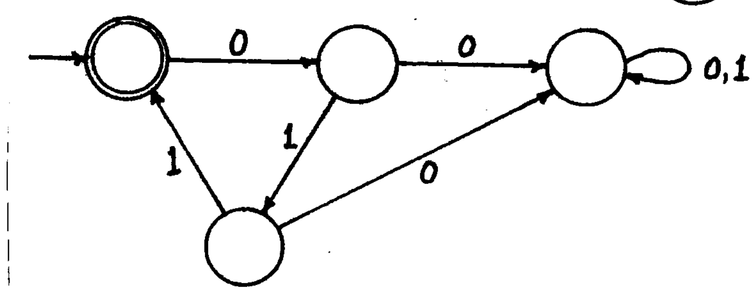
(f)



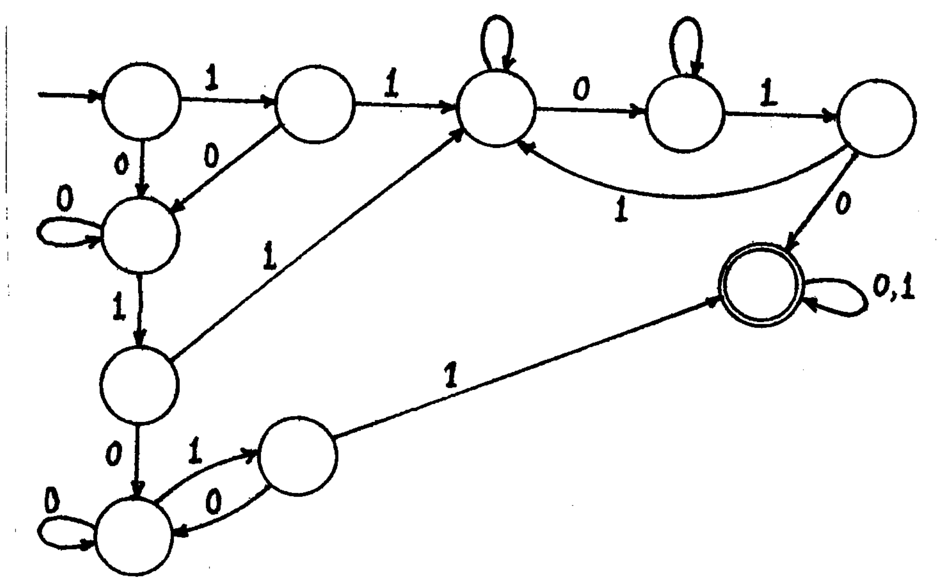
(g)



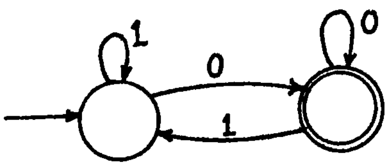
(h)



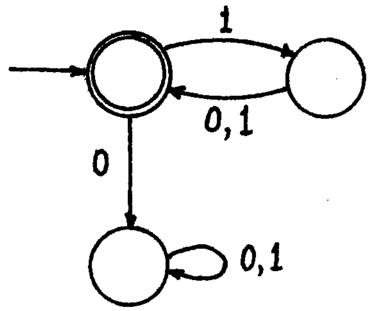
(i)



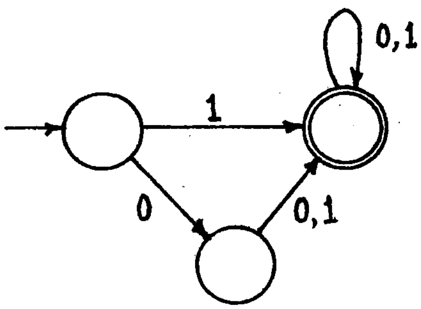
3.20 (a)



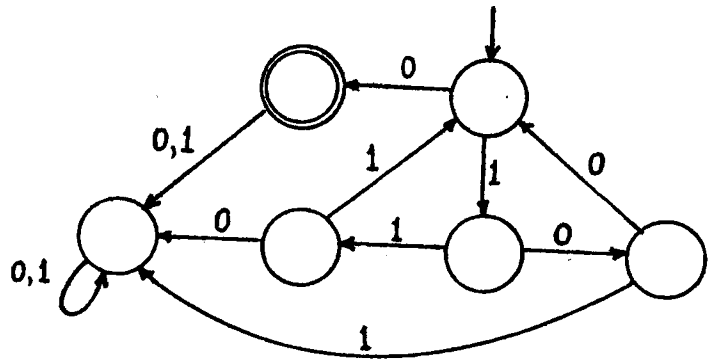
(b)



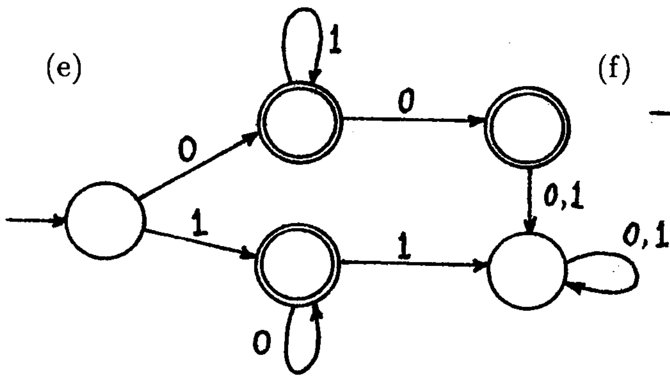
(c)



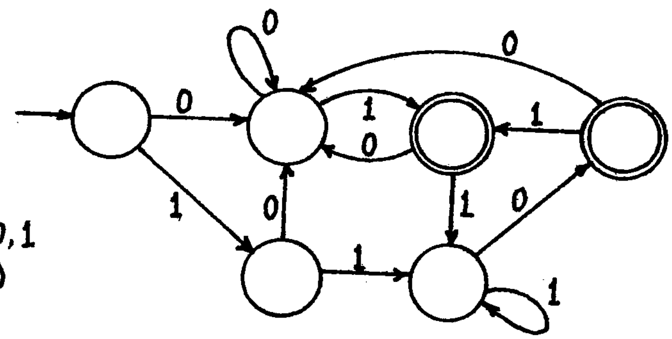
(d)



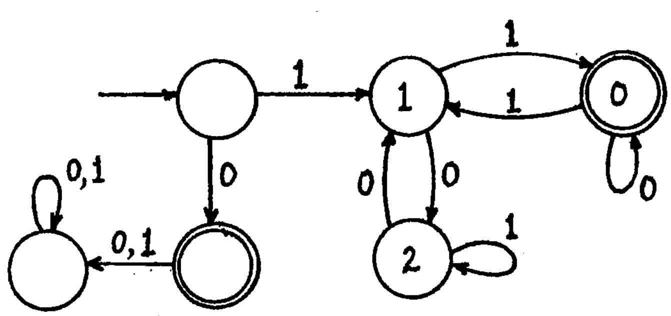
(e)



(f)



3.21. The numbered states correspond to remainders mod 3. Appending 0 or 1 to the string representing n yields $2n$ or $2n + 1$, respectively. The unnumbered accepting state represents the integer 0.



3.22. (a) We use structural induction on y. The basis step is to show that for any x and any q, $\delta^*(q, x\Lambda) = \delta^*(\delta^*(q, x), \Lambda)$. This is true because $x\Lambda = x$ and $\delta^*(p, \Lambda) = p$ for any p (in particular, for $p = \delta^*(q, x)$). Suppose y has the property that for any x and any q, $\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$. Now consider ya, for any $a \in \Sigma$.

$$\delta^*(q, x(ya)) = \delta^*(q, (xy)a) = \delta(\delta^*(q, xy), a) = \delta(\delta^*(\delta^*(q, x), y), a) = \delta^*(\delta^*(q, x), ya)$$

The second equality uses the definition of δ^*; the next one uses the induction hypothesis; and the last one uses the definition of $\delta^*(p, ya)$, where $p = \delta^*(q, x)$.

(b) We use structural induction on x. 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 $\delta^*(q, xa) = \delta(\delta^*(q, x), a) = \delta(q, a) = q$.

(c) In the induction step, we consider $\delta^*(q, x^{k+1}) = \delta^*(q, x^k x) = \delta^*(\delta^*(q, x^k), x) = \delta^*(q, x) = q$. The second equality uses the formula in part (a), and the next equality uses the induction hypothesis.

3.23. For any FA, a trivial way to get another FA with one more state accepting the same language is to add a state, allow no transitions to that state, and make all the transitions from that state go to one of the states in the original FA.

3.24. The language $L = \{\Lambda, 0\} \subseteq \{0, 1\}^*$ has this property, as does any language for which there are two strings in the language that are distinguishable with respect to the language. (A special case is that in which some strings in L are prefixes of other strings in L and some strings in L are not; however, more than one accepting state may be required even if this condition does not hold.)

3.25. The language accepted by the FA is the set L of strings that don't contain the substring 00 and don't end in 01. The simplest strings corresponding to the four states are Λ, 0, 01, and 00. It is possible to show that any two of these strings are distinguishable with respect to L. For example, Λ and 0 are distinguished by the string 0; Λ and 01 are distinguished by the string Λ, as are 0 and 01, and 0 and 00, and Λ and 00; and 00 and 01 are distinguished by the string 0.

3.26. $n + 1$ states are required. There are $n + 1$ prefixes of z, whose lengths vary from 0 to n, and any two of these can be distinguished with respect to $L = \{0, 1\}^*\{z\}$. (If $x_1y_1 = x_2y_2 = z$ and $|x_1| < |x_2|$, then $x_1y_2 \notin L$ and $x_2y_2 \in L$.) No more states are necessary, because for any string x, if $x = x_1y$ and y is a prefix of z and x does not end with any longer prefix of z, then x is indistinguishable from y with respect to L. (Note: every x ends with a prefix of z, if only the prefix Λ.)

3.27. No. A simple example is provided by the language L of all even-length strings of 0's and the sequence $x_0, x_1, \ldots,$ where $x_i = 0^i$. For any n, x_n and x_{n_1} are distinguished with respect to L by the string Λ. (One of the two is in L and the other is not.)

3.28. 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 any $x \in L_n$, a set of $n+2$ strings containing each of the $n+1$ prefixes of x as well as a string y of length $n+1$ is pairwise distinguishable with respect to L_n. (If $x_1 y_1 = x_2 y_2 = x$, and $|x_1| < |x_2|$, then x_1 and x_2 are distinguished by the string y_1. Furthermore, any of the the prefixes is distinguishable from y with respect to L_n.) The language $L = \{0,1\}^*$ is one example. Any infinite language that does not contain two strings of the same length is another.

3.29. If $x \in L(M)$, then for some accepting state q, $\delta^*(q_0, x) = q$. For each prefix x_1 of x, the state $\delta^*(q_0, x_1)$ is obviously reachable from q_0, and therefore still present in M_1. Therefore, $\delta_1^*(q_0, x)$ is still q, and $x \in L(M_1)$. If $x \in L(M_1)$, since all the transitions of M_1 are present in M, x is also in $L(M)$.

3.30. 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 $L(M)$.

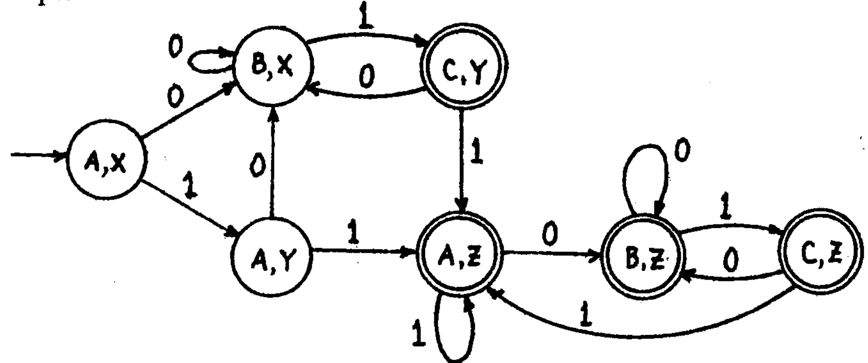
3.31. Every string is a prefix of an element of $L(M)$. (See the preceding exercise.)

3.32. 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 $a \in \Sigma$. Then

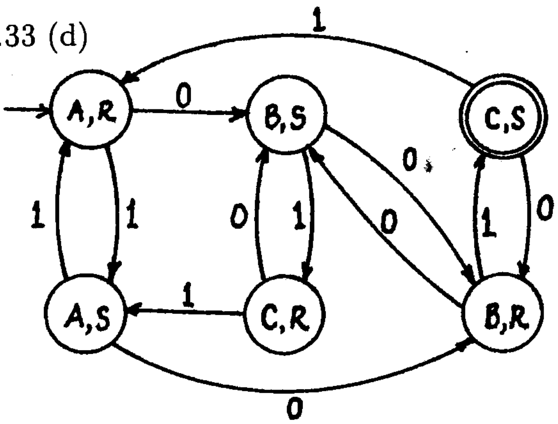
$$\begin{aligned}
\delta^*((p,q),xa) &= \delta(\delta^*((p,q),x),a) \\
&= \delta((\delta_1^*(p,x), \delta_2^*(q,x)),a) \\
&= (\delta_1(\delta_1^*(p,x),a), \delta_2(\delta_2^*(q,x),a)) \\
&= (\delta_1^*(p,xa), \delta_2^*(q,xa))
\end{aligned}$$

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^*.

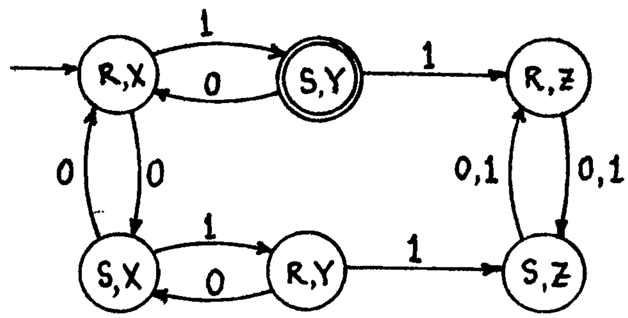
3.33 (a) The picture below is also the one for parts (b) and (c), except that in (b) the only accepting state is (C, Z) and in (c) the only accepting state is (C, Y). In both (a) and (c), the picture can be simplified.



3.33 (d)



(e)



3.34. On the one hand, any string matching $r + s$ matches r^*s^*, and therefore any string matching $(r + s)^*$ matches $(r^*s^*)^*$. On the other hand, if x is a string matching r^*s^*, then $x = x_1x_2$, where x_1 matches r^* and x_2 matches s^*. Then x_1 matches $(r + s)^*$ and so does x_2. Therefore, $x = x_1x_2$ also matches $(r + s)^*$.

3.35. We first show that for any $n \geq 0$, if x corresponds to the regular expression $(00^*1)^n1$, then x corresponds to the regular expression $1 + 0(0 + 10)^*11$. The proof for $n = 0$ is easy. Suppose that $k \geq 0$ and any string corresponding to $(00^*1)^k1$ corresponds to $1+0(0+10)^*11$. We must show that if x corresponds to $(00^*1)^{k+1}1$, then x corresponds to $1 + 0(0 + 10)^*11$. We know that $x = 00^j1y$, where y corresponds to $(00^*1)^k1$. By the induction hypothesis, y corresponds to $1 + 0(0 + 10)^*11$. If $y = 1$ the result is clear. Otherwise $y = 0z11$, where z corresponds to $(0 + 10)^*$. In this case $x = 00^j10z11$. Since 0^j10 corresponds to $(0 + 10)^*$, 0^j10z does also, and this implies the result.

Now we show the opposite direction, that if x corresponds to $1 + 0(0 + 10)^n11$, then x corresponds to $(00^*1)^*1$. Again the statement is clear when $n = 0$. Suppose $k \geq 0$ and any string corresponding to $1 + 0(0 + 10)^k11$ corresponds to $(00^*1)^*1$. Let x be a string corresponding to $1 + 0(0 + 10)^{k+1}11$. If $x = 1$ then x clearly corresponds to $(00^*1)^*1$. Otherwise, $x = 0z11$, where z corresponds to $(0 + 10)^{k+1}$. This implies that for some z' corresponding to $(0+10)^k$, either $x = 00z'11$ or $x = 010z'11$. By the induction hypothesis, $0z'11$ corresponds to $(00^*1)^*1$, and therefore to $(00^*1)^j1$ for some j. If $j = 1$, then $z' = 0^i$ for some i, and it is easy to see in this case that both $00z'11$ and $010z'11$ correspond to $(00^*1)^*1$, so that x is of the desired form. If $j \geq 2$ then $z' = 0^p1z''00^q$, where z'' corresponds to $(00^*1)^*$; in this case $00z'11 = 0^{p+2}1z''0^{q+1}11$, and $010z'11 = 010^{p+1}1z''0^{q+1}11$, so that again x has the right form.

3.36. (a) $(\Lambda + 0 + 00)(1 + 10 + 100)^*$
(b) Saying that a string doesn't contain 110 means that if 11 appears, then 0 cannot appear anywhere later. Therefore, a string not containing 110 consists of an initial portion not containing 11, possibly followed by a string of 1's. By including in the string of 1's *all* the trailing 1's, we may require that the initial portion doesn't end with 1. Since $(0 + 10)^*$ corresponds to strings not containing 11 and not ending with 1, one answer is $(0 + 10)^*1^*$.
(c) $(0 + 1)^*(101(0 + 1)^*010 + 010(0 + 1)^*101 + 1010 + 0101)(0 + 1)^*$

3.36 (d) $(00 + 11 + (01 + 10)(00 + 11)^*(01 + 10))^*$
(e) $e(01 + 10)e$, where e is the expression in (g).

3.37. We say a regular expression r over Σ has property P if, when each alphabet symbol of r is replaced by another regular expression over Σ, the result is also a regular expression over Σ. We use structural induction to show that every regular expression over Σ has property P.

The basis step is to show that the regular expressions $\emptyset$, Λ, and a have property P, for every $a \in \Sigma$. This is clear, because the only one of these that is changed by substituting regular expressions for elements of Σ is a, and in that case the result is a regular expression.
Induction hypothesis. The regular expressions r_1 and r_2 over Σ have property P.
Statement to be shown in induction step. The regular expressions $(r_1 + r_2)$, (r_1r_2), and (r_1^*) all have property P.
Proof. We take care of the first case, and the argument in the other two is similar. When a regular expression is substituted for each element of Σ in $(r_1 + r_2)$, the result is $(s_1 + s_2)$, where each s_i is obtained from the corresponding r_i by substituting regular expressions for alphabet symbols. By the induction hypothesis, each s_i is a regular expression. It follows by definition that $(s_1 + s_2)$ is also.

3.38. (b) Suppose r satisfies $r = c + rd$ and Λ does not correspond to d. Let x be a string corresponding to r. To show that x corresponds to cd^*, assume for the sake of contradiction that x does not correspond to cd^j for any j. We can then show using mathematical induction that for *every* $n \geq 0$, x corresponds to the regular expression rd^n.

The basis step $n = 0$ is clear, since x corresponds to r. Suppose that $k \geq 0$ and x corresponds to rd^k. $rd^k = (c + rd)d^k = cd^k + rd^{k+1}$; since x does not correspond to cd^k, it must correspond to rd^{k+1}.

Now it remains to derive a contradiction. Since Λ does not correspond to d, every string corresponding to d has length at least 1. It follows that every string corresponding to d^n must have length at least n, and therefore that x must have length at least n for every n. This is clearly impossible. We conclude that x must correspond to cd^j for some j.

3.39. Make a sequence of passes through the expression. In each pass, replace any regular expression of the form $(\emptyset + r)$ or $(r + \emptyset)$ by r (where r is any regular expression), any regular expression of the form $(\emptyset r)$ or $(r\emptyset)$ by $\emptyset$, and any occurrence of $(\emptyset^*)$ by Λ. Stop after any pass in which no changes are made. If $\emptyset$ remains in the expression, then the expression must actually be $\emptyset$, in which case the corresponding language is empty.

3.40. Make a sequence of passes through the expression. In each pass, replace any regular expression of the form (Λr) or $(r\Lambda)$ by r (where r is any regular expression); replace any occurrence of Λ^* by Λ; replace any regular expression of the form $(r + \Lambda)^*$ by (r^*) (where r is any regular expression); and replace any regular expression of the form $(\Lambda + r)s$ by $s + rs$ (where r and s are any regular expressions). Stop after any pass in which no changes are made. If Λ remains in the expression, then the expression corresponds to the language $\{\Lambda\}$.

3.41. (a) It is clear that if $L^k = L^*$, then $L^k = L^{k+1}$. On the other hand, suppose that $L^k = L^{k+1}$. Let m be the length of the shortest element of L. Then the shortest elements in L^k and L^{k+1} have lengths km and $(k+1)m$, respectively, which implies that m must be 0. Therefore, $\Lambda \in L$. It follows that $L^i \subseteq L^{i+1}$ for every i, and therefore that $L^* = \cup_{i=0}^{\infty} L^i \subseteq \cup_{i=k}^{\infty} L^i$. But in addition, $L^{k+i} \subseteq L^k$ for every i, so that $\cup_{i=k}^{\infty} L^i \subset L^k$. We conclude that $L^k = L^{k+1}$ if and only if $L^k = L^*$.
(b) The order is 3, because L^2 contains no string of length 6, and L^3 contains strings of all lengths ≥ 4.
(c) ∞, because the language does not contain Λ.
(d) It is not hard to see that this language contains every string in which the number of a's is either a multiple of 3, or 1 plus a multiple of 3. It follows from this that the order is 2.

3.42. (a) The language $\{aba\}^*$ can be described as the union of $\{\Lambda\}$ and the set of strings that start with ab and end with ba and contain none of the substrings bb, bab, and aaa. So one generalized regular expression describing this language is

$$\Lambda + ab\emptyset' \cap \emptyset' ba \cap (\emptyset' bb \emptyset')' \cap (\emptyset' bab \emptyset')' \cap (\emptyset' aaa \emptyset')'$$

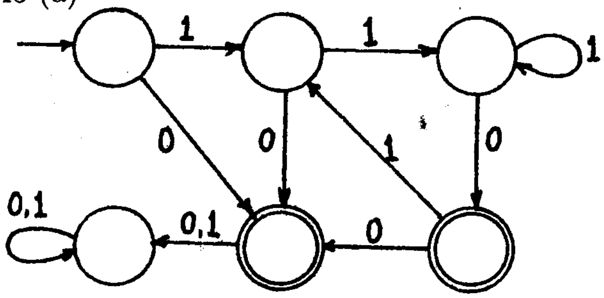
(b) $\{aa\}^*$ cannot be described this way. The intuitive reason is that without using *, there is no way to generate all even-length strings without also including some odd-length strings. A more rigorous proof follows.

Assume that the alphabet is $\{a\}$. (It's not hard to see that if we can show the result for this case, then it will still be impossible to describe $\{aa\}^*$ this way, even if larger alphabets are allowed.) For a language $L \subseteq \{a\}^*$, we let $e(L) = \{n \geq 0 \mid n \text{ is even and } a^n \in L\}$ and $e'(L) = \{n \geq 0 \mid n \text{ is even and } a^n \notin L\}$; similarly, $o(L) = \{n \geq 0 \mid n \text{ is odd and } a^n \in L\}$ and $o'(L) = \{n \geq 0 \mid n \text{ is odd and } a^n \notin L\}$. Obviously, $e(L) \cup e'(L)$ is the set of even integers, and $o(L) \cup o'(L)$ is the set of odd integers. We say L is *finitary* if either $e(L)$ or $e'(L)$ is finite and either $o(L)$ or $o'(L)$ is finite. The language $\{aa\}^*$ is nonfinitary, since both $e(\{aa\}^*)$ and $e'(\{aa\}^*)$ are infinite.

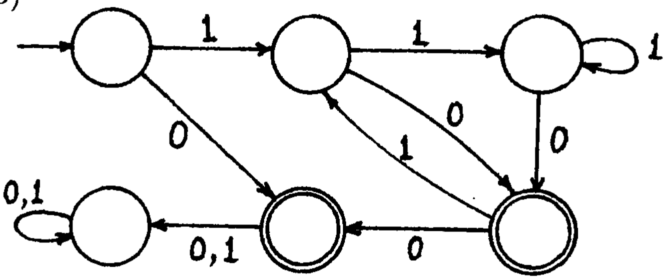
The fact that $\{aa\}^*$ cannot be described by a generalized regular expression not involving * follows from a more general result: no nonfinitary language can be described this way. This result follows from the fact that for languages $L_1, L_2 \subseteq \{a\}^*$, if L_1 and L_2 are finitary, then so are $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 L_2$, and L_1'. We will not verify all these statements, but we check one representative case.

Suppose, for example, that $e(L_1)$ is finite, $o(L_1)$ is finite, $e(L_2)$ is finite, and $o'(L_2)$ is finite. (Assume also that both L_1 and L_2 are nonempty.) Then $e(L_1 \cup L_2)$ is finite, and $o'(L_1 \cup L_2)$ is finite. Both $e(L_1 \cap L_2)$ and $o(L_1 \cap L_2)$ are finite. Both $e'(L_1')$ and $o'(L_1')$ are finite. Consider the number $e(L_1 L_2)$. If L_1 has no strings of odd length, then the strings in $L_1 L_2$ of even length are obtained by concatenating even-length strings in L_1 with even-length strings in L_2, and therefore $e(L_1 L_2)$ is finite. If there is an odd-length string in L_1, however, then since L_2 contains strings of almost all odd lengths, $L_1 L_2$ contains strings of almost all even lengths—i.e., $e'(L_1 L_2)$ is finite. Similarly, either $o(L_1 L_2)$ is finite (which will happen if L_1 has no strings of even length), or $o'(L_1 L_2)$ is finite (which is true if L_1 has a string of even length).

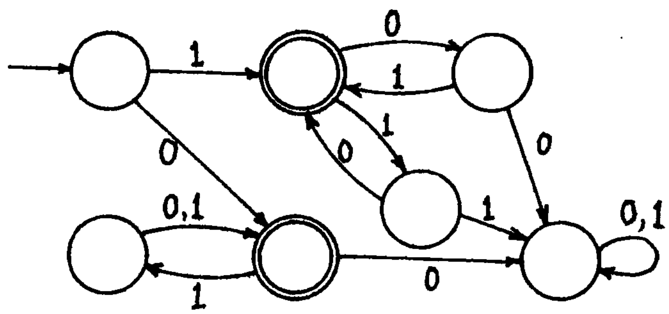
3.43 (a)


1
1
1
0
0
1
0
0,1
0,1
0

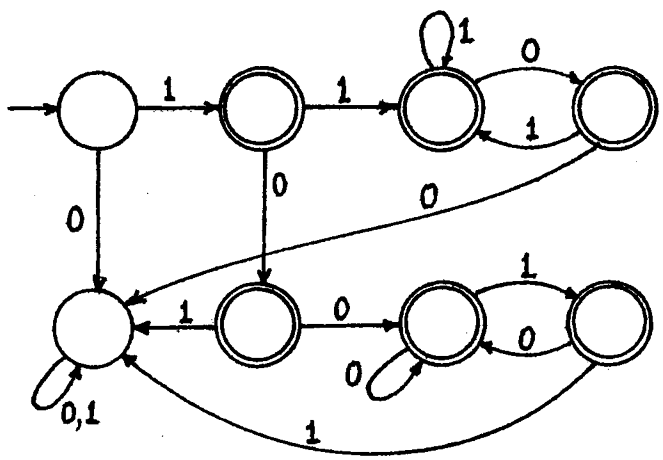
(b)


1
1
1
0
0
1
0
0,1
0,1
0

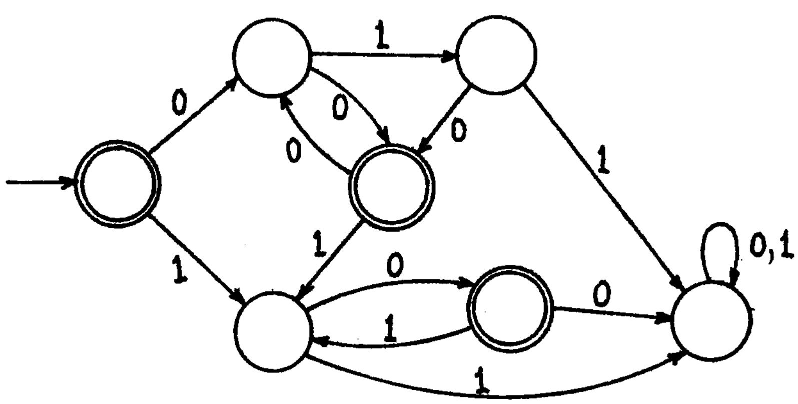
(c)


1
0
1
0
1
0
0
1
0,1
0,1
1
0

(d)


1
1
1
0
1
0
0
0
1
0
1
0
0
0,1
1

(e)


1
0
0
0
0
1
1
1
0
0
0,1
1
1

3.44. (a) Not valid. There is no way to use the definition to determine what $\delta^*(q, a)$ is, for $a \in \Sigma$.

(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 3.25(a).

(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.

3.45. Let $x, y \in \{0, 1\}^*$ with $x \neq y$. We consider three cases. If $|x| = |y|$, then $xx \in L$ and $yx \notin L$, so that x and y are distinguishable. Otherwise relabel the strings if necessary so that $|x| < |y| = |x| + k$. If k is odd, then $xx \in L$, and $xy \notin L$ because $|xy| = 2|x| + k$, which is odd. Finally, if $k = 2j > 0$, let $w = w_1 w_2$, where w_1 and w_2 are any strings for which $|w_1| = |w_2| = j$ and the first symbol of w_2 is different from the first symbol of y. Then $xwxw \in L$. However, $ywxw = (yw_1)(w_2 x w_1 w_2)$, where the two parenthesized strings are of equal lengths and start with different symbols, so that $ywxw \notin L$.

3.46. (a) One answer is any two strings, neither of which is a prefix of an element of L: 1 and 10, for example. Another answer is any two distinct nonnull elements of L. 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 001 and 00011. Here an element of L is obtained in both cases if 1 is added to the end, and an element of L' is obtained in both cases if anything else is added to the end.

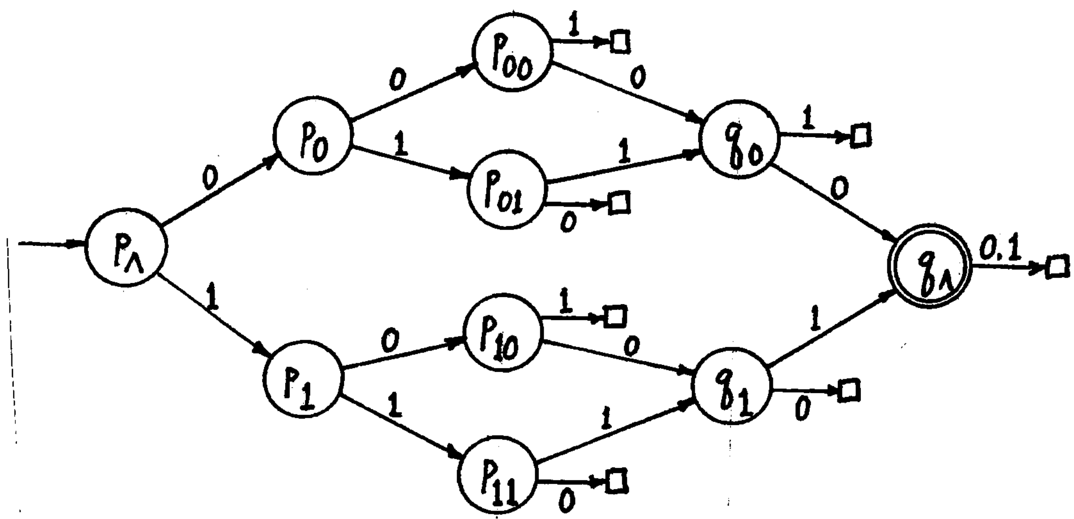
(b) Any two elements of $\{0^n \mid n \geq 0\}$ are distinguishable with respect to L.

3.47. In general, the 2^n states correspond to the 2^n possible strings of length n. We can see how to draw the transitions by thinking of the state corresponding to string x as representing the last n symbols of the input string: $\delta(a_1 a_2 \ldots a_n, a) = a_2 \ldots a_n a$. A string x of length $k < n$ corresponds to the string $0^{n-k}x$ with leading 0's. The initial state, therefore, is the one corresponding to the string 0^n. The accepting states are the ones corresponding to strings beginning with 1.

3.48. 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 $0 \le i \le m$ and $0 \le j \le m$, and i and j represent the numbers of 0's and 1's, respectively, in the input string. For such a pair (i, j), each string with i 0's and j 1'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 0's and j 1's, and $i < m$, then $\delta((i,j),0) = (i+1, j)$; similarly for $\delta((i,j),0)$ if $j < m$. For each $i < m$, $\delta((m,i),0) = \delta((i,m),1) = N$.

This is the minimum possible number of states. It is straightforward to show that any set of $(m+1)^2 + 1$ strings consisting of one for each state is pairwise distinguishable with respect to L.

3.50. 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. There is a state q_D, which is not an accepting state, and $\delta(q_D, 0) = \delta(q_D, 1) = q_D$. Every arrow in the picture that ends in a square actually goes to q_D.

It is straightforward to generalize this to arbitrary n, so as to obtain an FA M_n. For each string x of length $\le n$, 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 other strings are taken care of by the single state q_D. Since there are $2^{n+1} - 1$ strings of length $\le n$ 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 $|x| \ne |y|$, then x and y are distinguishable. It is therefore sufficient to show: (i) if $|x_1| = |x_2| \le n$ and $x_1 \ne x_2$, then x_1 and x_2 are distinguishable; and (ii) if $|y_1| = |y_2|$, $y_1 \ne y_2$, $x_1 y_1 \in L$, and $x_2 y_2 \in L$, then x_1 and x_2 are distinguishable. Statement (i) is easy:

$$x_1 0^{2n-2|x_1|} x_1^r \in L \text{ and } x_2 0^{2n-2|x_1|} x_1^r \notin L$$

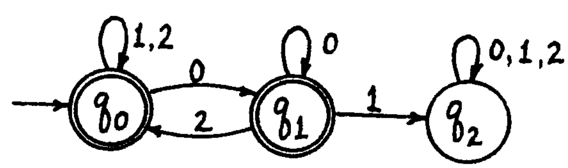
Statement (ii) is just as easy: if $x_1 y_1 \in L$ and $y_1 \neq y_2$, then since $x_2 y_2 \in L$, we must have $x_2 y_1 \notin L$. Therefore, y_1 distinguishes x_1 and x_2.

3.51. Suppose there an n and a set S of strings of length n so that for any x, $x \in L$ if and only if $x = x_1 x_2$ for some $x_2 \in S$. Then $\{x \in L \mid |x| \geq n\} = \Sigma^* S$, which is the concatenation of regular languages and therefore regular. L is the union of this language and a finite language, so that L is also regular.

3.52. A finite language L satisfies the property in a trivial way. We may choose n bigger than the length of the longest string in L, and S to be the empty set. Then for any string x with length n or greater, $x \in L$ if and only if x ends with some element of S, because neither condition is true for x.

3.53. Let $L = \{0x \mid x \in \{0,1\}^*\}$. 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.

3.54. The FA is pictured here.



Let a_i, b_i, and c_i be the number of strings x of length i for which $\delta^*(q_0, x)$ is q_0, q_1, and q_2, respectively. Then for each i, we may write these equations:

$$a_{i+1} = 2a_i + b_i \qquad b_{i+1} = a_i + b_i$$

(The first equation, for example, follows from the fact that there are two arrows to state q_0 from state q_0 and one to state q_0 from state q_1.) If we let n_i denote the number of strings of length i that don't contain the substring 01, then $n_i = a_i + b_i$. We may write

$$\begin{aligned} n_{i+1} &= a_{i+1} + b_{i+1} &&= 2a_i + b_i + a_i + b_i \\ &= 3(a_i + b_i) - b_i &&= 3n_i - (a_{i-1} + b_{i-1}) \\ &= 3n_i - n_{i-1} \end{aligned}$$

It is easy to verify that the sequence $m_i = f(2i+2)$ satisfies the same recursive relationship: $m_{i+1} = 3m_i - m_{i-1}$ for every $i \geq 1$. Furthermore, $n_0 = m_0$ and $n_1 = m_1$. Therefore, $n_i = m_i$ for every $i \geq 0$.

3.55. (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 $\{0,1\}^*$. 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

$\delta_1(q_0, 0) \neq \delta_2(q_0, 0)$, then 0 is in one of the languages but not the other, and similarly if $\delta_1(q_0, 1) \neq \delta_2(q_0, 1)$. Finally, if the transitions from the accepting state are the same in M_1 and M_2, and $\delta_1(q_1, 0) \neq \delta_2(q_1, 0)$, then there is a string with second symbol 0 that is in one but not both of the languages, and similarly if $\delta_1(q_1, 1) \neq \delta_2(q_1, 1)$. The conclusion is that the 36 nonisomorphic FAs in (e) accept 25 distinct languages.