ing any token. Usually, they convert the regular expression to an NFA with ϵ-transitions and then construct subsets of states to produce a DFA directly, rather than first eliminating ϵ-transitions. Each final state indicates the particular token found, so the automaton is really a Moore machine. The transition function of the FA is encoded in one of several ways to take less space than the transition table would take if represented as a two-dimensional array. The lexical analyzer produced by the generator is a fixed program that interprets coded tables, together with the particular table that represents the FA recognizing the tokens (specified to the generator in regular expression notation). This lexical analyzer may then be used as a module in a compiler. Examples of lexical analyzer generators that follow the above approach are found in Johnson *et al.* [1968] and Lesk [1975].

Text editors

Certain text editors and similar programs permit the substitution of a string for any string matching a given regular expression. For example, the UNIX text editor allows a command such as

$$s/\text{b}\text{b}\text{b}^*/\text{b}/$$

that substitutes a single blank for the first string of two or more blanks found in a given line. Let "any" denote the expression $a_1 + a_2 + \cdots + a_n$, where the a_i's are all of a computer's characters except the "newline" character. We could convert a regular expression r to a DFA that accepts any*r. Note that the presence of any* allows us to recognize a member of $L(r)$ beginning anywhere in the line. However, the conversion of a regular expression to a DFA takes far more time than it takes to scan a single short line using the DFA, and the DFA could have a number of states that is an exponential function of the length of the regular expression.

What actually happens in the UNIX text editor is that the regular expression any*r is converted to an NFA with ϵ-transitions, and the NFA is then simulated directly, as suggested in Fig. 2.6. However, once a column has been constructed listing all the states the NFA can enter on a particular prefix of the input, the previous column is no longer needed and is thrown away to save space. This approach to regular set recognition was first expressed in Thompson [1968].

EXERCISES

*S 2.1 Find a finite automaton whose behavior corresponds to the circuit in Fig. 2.26, in the sense that final states correspond to a 1-output. A circle with a dot represents an *AND-gate*, whose output is 1 only if both inputs have value 1. A circle with a + represents an *OR-gate*, whose output is 1 whenever either input has value 1. A circle with a ~ represents an inverter, whose output is 1 for input 0 and 0 for input 1. Assume there is sufficient time between changes in input values for signals to propagate and for the network to reach a stable configuration.

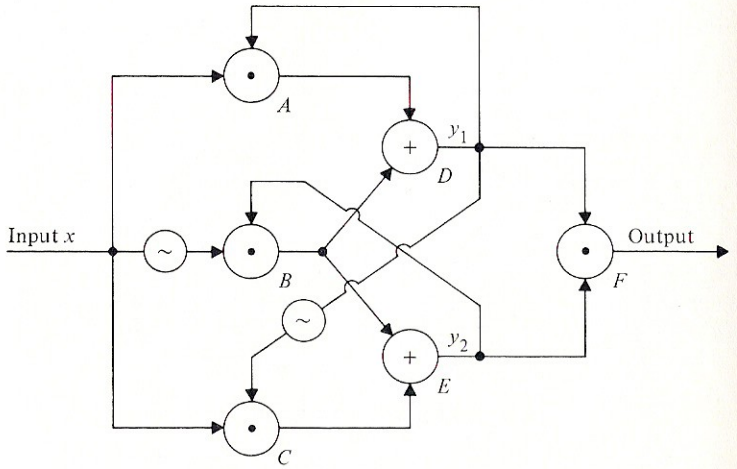

Fig. 2.26 A logic circuit.

2.2 Historically, finite automata were first used to model neuron nets. Find a finite automaton whose behavior is equivalent to the neuron net in Fig. 2.27. Final states of the automaton correspond to a 1-output of the network. Each neuron has excitatory (circles) and inhibitory (dots) synapses. A neuron produces a 1-output if the number of excitatory synapses with 1-inputs exceeds the number of inhibitory synapses with 1-inputs by at least the threshold of the neuron (number inside the triangle). Assume there is sufficient time between changes in input value for signals to propagate and for the network to reach a stable configuration. Further assume that initially the values of y_1, y_2, and y_3 are all 0.

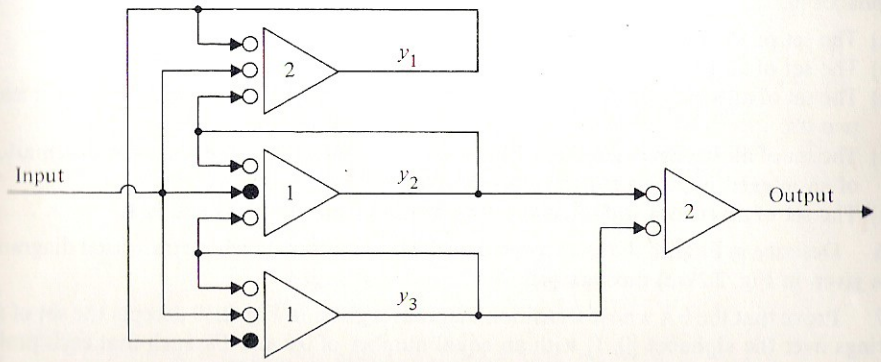

Fig. 2.27 A neuron net.

2.3 Consider the toy shown in Fig. 2.28. A marble is dropped in at A or B. Levers x_1, x_2, and x_3 cause the marble to fall either to the left or right. Whenever a marble encounters a

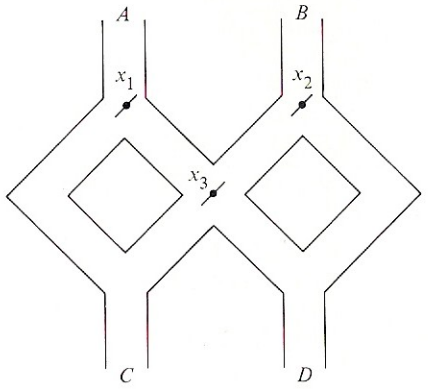

Fig. 2.28 A toy.

lever, it causes the lever to change state, so that the next marble to encounter the lever will take the opposite branch.

a) Model this toy by a finite automaton. Denote a marble in at A by a 0-input and a marble in at B by a 1-input. A sequence of inputs is accepted if the last marble comes out at D.
b) Describe the set accepted by the finite automaton.
c) Model the toy as a Mealy machine whose output is the sequence of C's and D's out of which successive marbles fall.

2.4 Suppose δ is the transition function of a DFA. Prove that for any input strings x and y, $\delta(q, xy) = \delta(\delta(q, x), y)$. [*Hint:* Use induction on $|y|$.]

2.5 Give deterministic finite automata accepting the following languages over the alphabet $\{0, 1\}$.

a) The set of all strings ending in 00.
b) The set of all strings with three consecutive 0's.
c) The set of all strings such that every block of five consecutive symbols contains at least two 0's.
d) The set of all strings beginning with a 1 which, interpreted as the binary representation of an integer, is congruent to zero modulo 5.
e) The set of all strings such that the 10th symbol from the right end is 1.

* **2.6** Describe in English the sets accepted by the finite automata whose transition diagrams are given in Fig. 2.29(a) through (c).

†S **2.7** Prove that the FA whose transition diagram is given in Fig. 2.30 accepts the set of all strings over the alphabet $\{0, 1\}$ with an equal number of 0's and 1's, such that each prefix has at most one more 0 than 1's and at most one more 1 than 0's.

2.8 Give nondeterministic finite automata accepting the following languages.

a) The set of strings in $(\mathbf{0} + \mathbf{1})^*$ such that some two 0's are separated by a string whose length is $4i$, for some $i \geq 0$.
b) The set of all strings over the alphabet $\{a, b, c\}$ that have the same value when evaluated left to right as right to left by multiplying according to the table in Fig. 2.31.

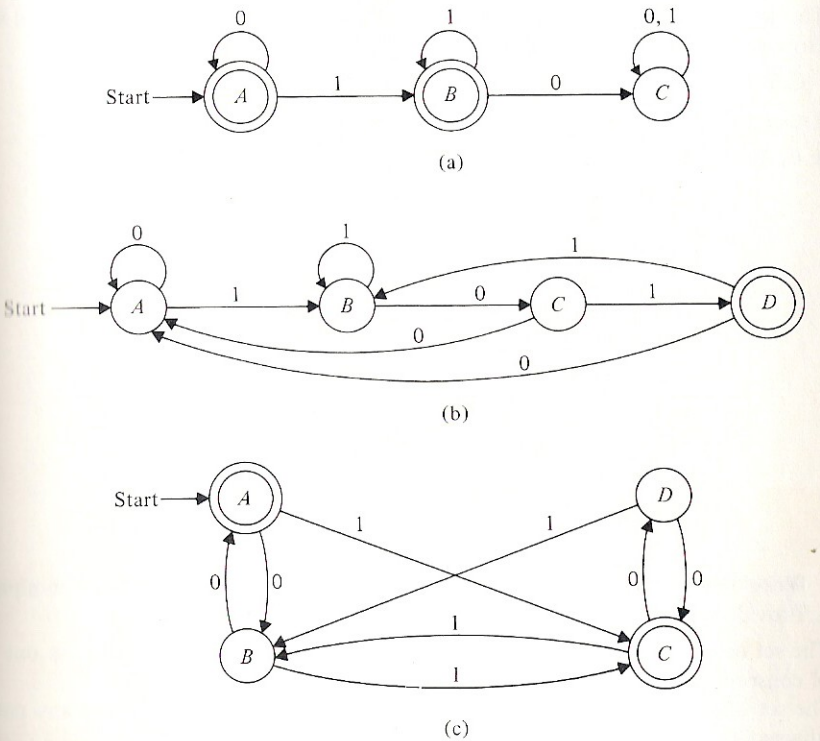

Fig. 2.29 Transition diagrams for finite automata.

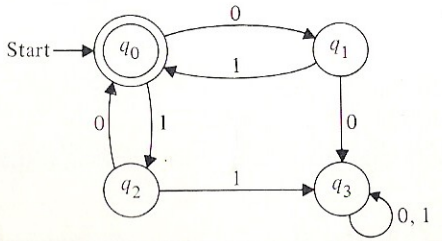

Fig. 2.30 Transition diagram.

	a	b	c
a	a	a	c
b	c	a	b
c	b	c	a

Fig. 2.31 Nonassociative multiplication table.

c) The set of all strings of 0's and 1's such that the 10th symbol from the right end is a 1. How does your answer compare with the DFA of Problem 2.5(e)?

2.9 Construct DFA's equivalent to the NFA's.

a) $(\{p, q, r, s\}, \{0, 1\}, \delta_1, p, \{s\})$, b) $(\{p, q, r, s\}, \{0, 1\}, \delta_2, p, \{q, s\})$

where δ_1 and δ_2 are given in Fig. 2.32.

	0	1
p	p, q	p
q	r	r
r	s	—
s	s	s

δ_1

	0	1
p	q, s	q
q	r	q, r
r	s	p
s	—	p

δ_2

Fig. 2.32 Two transition functions.

2.10 Write regular expressions for each of the following languages over the alphabet $\{0, 1\}$. Provide justification that your regular expression is correct.

* a) The set of all strings with at most one pair of consecutive 0's and at most one pair of consecutive 1's.

b) The set of all strings in which every pair of adjacent 0's appears before any pair of adjacent 1's.

c) The set of all strings not containing 101 as a substring.

* d) The set of all strings with an equal number of 0's and 1's such that no prefix has two more 0's than 1's nor two more 1's than 0's.

2.11 Describe in English the sets denoted by the following regular expressions.

a) $(11 + 0)^*(00 + 1)^*$

b) $(1 + 01 + 001)^*(\epsilon + 0 + 00)$

c) $[00 + 11 + (01 + 10)(00 + 11)^*(01 + 10)]^*$

2.12 Construct finite automata equivalent to the following regular expressions.

a) $10 + (0 + 11)0^*1$

b) $01[((10)^* + 111)^* + 0]^*1$

c) $((0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*$

2.13 Construct regular expressions corresponding to the state diagrams given in Fig. 2.33.

2.14 Use the ideas in the proof of Theorem 2.4 to construct algorithms for the following problems.

a) Find the lowest-cost path between two vertices in a directed graph where each edge is labeled with a nonnegative cost.

b) Determine the number of strings of length n accepted by an FA.

2.15 Construct an NFA equivalent to the 2DFA $(\{q_0, \ldots, q_5\}, \{0, 1\}, \delta, q_0, \{q_2\})$, where δ is given by Fig. 2.34.

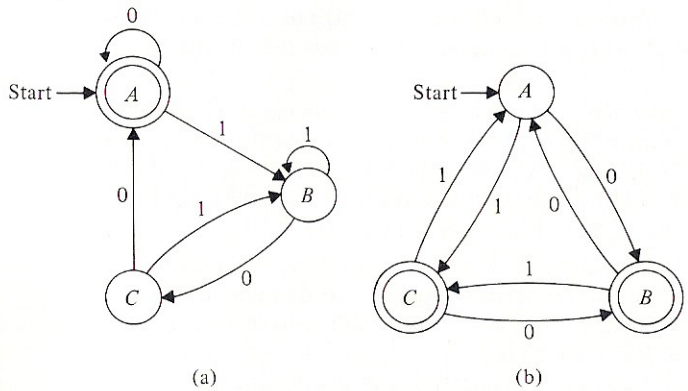

Fig. 2.33 Transition diagrams.

	0	1
q_0	(q_0, R)	(q_1, R)
q_1	(q_1, R)	(q_2, R)
q_2	(q_2, R)	(q_3, L)
q_3	(q_4, L)	(q_3, L)
q_4	(q_0, R)	(q_4, L)

Fig. 2.34 A transition function for a 2DFA.

2.16 Prove the following identities for regular expressions r, s, and t. Here $r = s$ means $L(r) = L(s)$.

a) $r + s = s + r$ b) $(r + s) + t = r + (s + t)$

c) $(rs)t = r(st)$ d) $r(s + t) = rs + rt$

e) $(r + s)t = rt + st$ f) $\varnothing^* = \epsilon$

g) $(r^*)^* = r^*$ h) $(\epsilon + r)^* = r^*$ i) $(r^*s^*)^* = (r + s)^*$

2.17 Prove or disprove the following for regular expressions r, s, and t.

a) $(rs + r)^*r = r(sr + r)^*$ b) $s(rs + s)^*r = rr^*s(rr^*s)^*$

c) $(r + s)^* = r^* + s^*$

2.18 A *two-way nondeterministic finite automaton* (2NFA) is defined in the same manner as the 2DFA, except that the 2NFA has a set of possible moves for each state and input symbol. Prove that the set accepted by any 2NFA is regular. [*Hint:* The observation in the proof of Theorem 2.5 that no state may repeat with the same direction in a valid crossing sequence is no longer true. However, for each accepted input we may consider a shortest computation leading to acceptance.]

2.19 Show that adding the capability of the 2NFA to keep its head stationary (and change state) on a move does not increase the class of languages accepted by 2NFA.

2.20 A 2NFA *with endmarkers* is a 2NFA with special symbols ¢ and $ marking the left and right ends of the input. We say that input x, which contains no ¢ or $ symbols, is

accepted if the 2NFA started with ¢x$ on its tape and with the tape head scanning ¢ enters an accepting state anywhere on its input. Show that the 2NFA with endmarkers accepts only regular sets.

2.21 Consider a 2DFA $M = (Q, \Sigma, \delta, q_0, F)$. For each string x construct a mapping f from Q to $Q \cup \{\not\in\}$, where $f(q) = p$ if the 2DFA started on the rightmost symbol of x eventually moves off x to the right, in state p. $f(q) = \not\in$ means that the 2DFA when started on the rightmost symbol of x either never leaves x or moves off the left end. Construct a DFA which simulates M by storing in its finite control a table f instead of a crossing sequence.

** **2.22** Let r and s be regular expressions. Consider the equation $X = rX + s$, where rX denotes the concatenation of r and X, and $+$ denotes union. Under the assumption that the set denoted by r does not contain ϵ, find the solution for X and prove that it is unique. What is the solution if $L(r)$ contains ϵ?

** **2.23** One can construct a regular expression from a finite automaton by solving a set of linear equations of the form

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix},$$

where a_{ij} and c_i are sets of strings denoted by regular expressions, $+$ denotes set union, and multiplication denotes concatenation. Give an algorithm for solving such equations.

2.24 Give Mealy and Moore machines for the following processes:

a) For input from $(0 + 1)^*$, if the input ends in 101, output A; if the input ends in 110, output B; otherwise output C.

b) For input from $(0 + 1 + 2)^*$ print the residue modulo 5 of the input treated as a ternary (base 3, with digits 0, 1, and 2) number.

Solutions to Sample Exercises

2.1 Note that the gate output at y_1 affects the gate output at y_2 and conversely. We shall assume values for y_1 and y_2 and use these assumed values to compute new values. Then we repeat the process with the new values until we reach a stable state of the system. In Fig. 2.35 we have tabulated the stable values of y_1 and y_2 for each possible assumed values for y_1 and y_2 and for input values 0 and 1.

$y_1 y_2$	Input 0	Input 1
00	00	01
01	11	01
11	11	10
10	00	10

(a)

	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_2	q_3
q_3	q_0	q_3

(b)

Fig. 2.35 Transitions of switching circuit.