1.5 Each of the following languages is the complement of a simpler language. In each part, construct a DFA for the simpler language, then use it to give the state diagram of a DFA for the language given. In all parts, $\Sigma = \{\texttt{a}, \texttt{b}\}$.

$^{\mathsf{A}}$**a.** $\{w \mid w$ does not contain the substring ab$\}$
$^{\mathsf{A}}$**b.** $\{w \mid w$ does not contain the substring baba$\}$
c. $\{w \mid w$ contains neither the substrings ab nor ba$\}$
d. $\{w \mid w$ is any string not in $\texttt{a}^*\texttt{b}^*\}$
e. $\{w \mid w$ is any string not in $(\texttt{ab}^+)^*\}$
f. $\{w \mid w$ is any string not in $\texttt{a}^* \cup \texttt{b}^*\}$
g. $\{w \mid w$ is any string that doesn't contain exactly two a's$\}$
h. $\{w \mid w$ is any string except a and b$\}$

1.6 Give state diagrams of DFAs recognizing the following languages. In all parts, the alphabet is $\{0,1\}$.

a. $\{w \mid w$ begins with a 1 and ends with a 0$\}$
b. $\{w \mid w$ contains at least three 1s$\}$
c. $\{w \mid w$ contains the substring 0101 (i.e., $w = x0101y$ for some x and y)$\}$
d. $\{w \mid w$ has length at least 3 and its third symbol is a 0$\}$
e. $\{w \mid w$ starts with 0 and has odd length, or starts with 1 and has even length$\}$
f. $\{w \mid w$ doesn't contain the substring 110$\}$
g. $\{w \mid$ the length of w is at most 5$\}$
h. $\{w \mid w$ is any string except 11 and 111$\}$
i. $\{w \mid$ every odd position of w is a 1$\}$
j. $\{w \mid w$ contains at least two 0s and at most one 1$\}$
k. $\{\varepsilon, 0\}$
l. $\{w \mid w$ contains an even number of 0s, or contains exactly two 1s$\}$
m. The empty set
n. All strings except the empty string

1.7 Give state diagrams of NFAs with the specified number of states recognizing each of the following languages. In all parts, the alphabet is $\{0,1\}$.

$^{\mathsf{A}}$**a.** The language $\{w \mid w$ ends with 00$\}$ with three states
b. The language of Exercise 1.6c with five states
c. The language of Exercise 1.6l with six states
d. The language $\{0\}$ with two states
e. The language $0^*1^*0^+$ with three states
$^{\mathsf{A}}$**f.** The language $1^*(001^+)^*$ with three states
g. The language $\{\varepsilon\}$ with one state
h. The language 0^* with one state

1.8 Use the construction in the proof of Theorem 1.45 to give the state diagrams of NFAs recognizing the union of the languages described in

a. Exercises 1.6a and 1.6b.
b. Exercises 1.6c and 1.6f.

1.9 Use the construction in the proof of Theorem 1.47 to give the state diagrams of NFAs recognizing the concatenation of the languages described in

a. Exercises 1.6g and 1.6i.
b. Exercises 1.6b and 1.6m.

1.10 Use the construction in the proof of Theorem 1.49 to give the state diagrams of NFAs recognizing the star of the languages described in

a. Exercise 1.6b.
b. Exercise 1.6j.
c. Exercise 1.6m.

^{A}**1.11** Prove that every NFA can be converted to an equivalent one that has a single accept state.

1.12 Let $D = \{w|\ w$ contains an even number of a's and an odd number of b's and does not contain the substring ab$\}$. Give a DFA with five states that recognizes D and a regular expression that generates D. (Suggestion: Describe D more simply.)

1.13 Let F be the language of all strings over $\{0,1\}$ that do not contain a pair of 1s that are separated by an odd number of symbols. Give the state diagram of a DFA with five states that recognizes F. (You may find it helpful first to find a 4-state NFA for the complement of F.)

1.14 **a.** Show that if M is a DFA that recognizes language B, swapping the accept and nonaccept states in M yields a new DFA recognizing the complement of B. Conclude that the class of regular languages is closed under complement.
b. Show by giving an example that if M is an NFA that recognizes language C, swapping the accept and nonaccept states in M doesn't necessarily yield a new NFA that recognizes the complement of C. Is the class of languages recognized by NFAs closed under complement? Explain your answer.

1.15 Give a counterexample to show that the following construction fails to prove Theorem 1.49, the closure of the class of regular languages under the star operation.[7] Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1. Construct $N = (Q_1, \Sigma, \delta, q_1, F)$ as follows. N is supposed to recognize A_1^*.

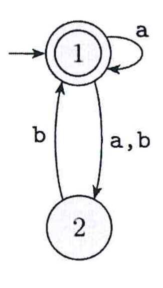
a. The states of N are the states of N_1.
b. The start state of N is the same as the start state of N_1.
c. $F = \{q_1\} \cup F_1$.
The accept states F are the old accept states plus its start state.
d. Define δ so that for any $q \in Q_1$ and any $a \in \Sigma_\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \notin F_1 \text{ or } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon. \end{cases}$$

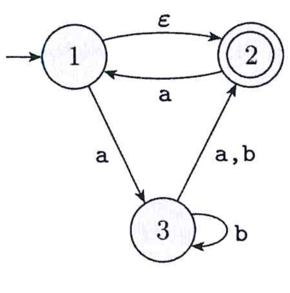
(Suggestion: Show this construction graphically, as in Figure 1.50.)

[7]In other words, you must present a finite automaton, N_1, for which the constructed automaton N does not recognize the star of N_1's language.

1.16 Use the construction given in Theorem 1.39 to convert the following two nondeterministic finite automata to equivalent deterministic finite automata.



(a)



(b)

1.17 **a.** Give an NFA recognizing the language (01 ∪ 001 ∪ 010)*.

b. Convert this NFA to an equivalent DFA. Give only the portion of the DFA that is reachable from the start state.

1.18 Give regular expressions generating the languages of Exercise 1.6.

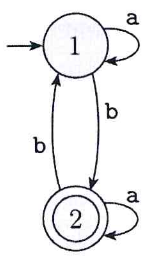
1.19 Use the procedure described in Lemma 1.55 to convert the following regular expressions to nondeterministic finite automata.

a. (0 ∪ 1)*000(0 ∪ 1)*
b. (((00)*(11)) ∪ 01)*
c. ∅*

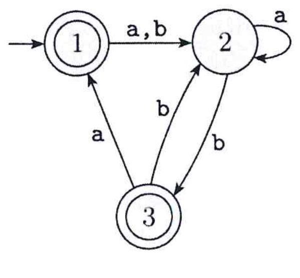
1.20 For each of the following languages, give two strings that are members and two strings that are *not* members—a total of four strings for each part. Assume the alphabet $\Sigma = \{a,b\}$ in all parts.

a. a*b*
b. a(ba)*b
c. a* ∪ b*
d. (aaa)*
e. $\Sigma^*a\Sigma^*b\Sigma^*a\Sigma^*$
f. aba ∪ bab
g. (ε ∪ a)b
h. (a ∪ ba ∪ bb)Σ^*

1.21 Use the procedure described in Lemma 1.60 to convert the following finite automata to regular expressions.



(a)



(b)

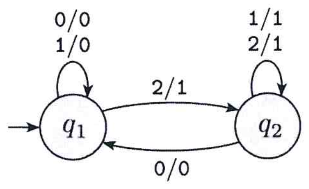
1.22 In certain programming languages, comments appear between delimiters such as `/#` and `#/`. Let C be the language of all valid delimited comment strings. A member of C must begin with `/#` and end with `#/` but have no intervening `#/`. For simplicity, assume that the alphabet for C is $\Sigma = \{\mathtt{a}, \mathtt{b}, /, \#\}$.

a. Give a DFA that recognizes C.

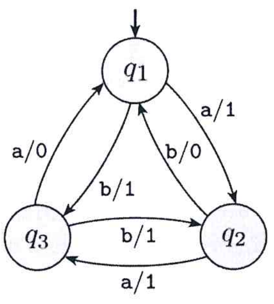
b. Give a regular expression that generates C.

^{A}**1.23** Let B be any language over the alphabet Σ. Prove that $B = B^{+}$ iff $BB \subseteq B$.

1.24 A ***finite state transducer*** (FST) is a type of deterministic finite automaton whose output is a string and not just *accept* or *reject*. The following are state diagrams of finite state transducers T_1 and T_2.



T_1



T_2

Each transition of an FST is labeled with two symbols, one designating the input symbol for that transition and the other designating the output symbol. The two symbols are written with a slash, /, separating them. In T_1, the transition from q_1 to q_2 has input symbol 2 and output symbol 1. Some transitions may have multiple input–output pairs, such as the transition in T_1 from q_1 to itself. When an FST computes on an input string w, it takes the input symbols $w_1 \cdots w_n$ one by one and, starting at the start state, follows the transitions by matching the input labels with the sequence of symbols $w_1 \cdots w_n = w$. Every time it goes along a transition, it outputs the corresponding output symbol. For example, on input 2212011, machine T_1 enters the sequence of states $q_1, q_2, q_2, q_2, q_2, q_1, q_1, q_1$ and produces output 1111000. On input abbb, T_2 outputs 1011. Give the sequence of states entered and the output produced in each of the following parts.

a. T_1 on input 011

b. T_1 on input 211

c. T_1 on input 121

d. T_1 on input 0202

e. T_2 on input b

f. T_2 on input bbab

g. T_2 on input bbbbbb

h. T_2 on input ε

1.25 Read the informal definition of the finite state transducer given in Exercise 1.24. Give a formal definition of this model, following the pattern in Definition 1.5 (page 35). Assume that an FST has an input alphabet Σ and an output alphabet Γ but not a set of accept states. Include a formal definition of the computation of an FST. (Hint: An FST is a 5-tuple. Its transition function is of the form $\delta\colon Q \times \Sigma \longrightarrow Q \times \Gamma$.)

1.26 Using the solution you gave to Exercise 1.25, give a formal description of the machines T_1 and T_2 depicted in Exercise 1.24.

1.27 Read the informal definition of the finite state transducer given in Exercise 1.24. Give the state diagram of an FST with the following behavior. Its input and output alphabets are $\{0,1\}$. Its output string is identical to the input string on the even positions but inverted on the odd positions. For example, on input 0000111 it should output 1010010.

1.28 Convert the following regular expressions to NFAs using the procedure given in Theorem 1.54. In all parts, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

a. $\mathtt{a(abb)^* \cup b}$
b. $\mathtt{a^+ \cup (ab)^+}$
c. $\mathtt{(a \cup b^+)a^+b^+}$

1.29 Use the pumping lemma to show that the following languages are not regular.

^{A}**a.** $A_1 = \{\mathtt{0}^n\mathtt{1}^n\mathtt{2}^n \mid n \geq 0\}$
b. $A_2 = \{www \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$
^{A}**c.** $A_3 = \{\mathtt{a}^{2^n} \mid n \geq 0\}$ (Here, $\mathtt{a}^{2^n}$ means a string of 2^n a's.)

1.30 Describe the error in the following "proof" that $\mathtt{0^*1^*}$ is not a regular language. (An error must exist because $\mathtt{0^*1^*}$ *is* regular.) The proof is by contradiction. Assume that $\mathtt{0^*1^*}$ is regular. Let p be the pumping length for $\mathtt{0^*1^*}$ given by the pumping lemma. Choose s to be the string $\mathtt{0}^p\mathtt{1}^p$. You know that s is a member of $\mathtt{0^*1^*}$, but Example 1.73 shows that s cannot be pumped. Thus you have a contradiction. So $\mathtt{0^*1^*}$ is not regular.

PROBLEMS

1.31 For languages A and B, let the ***perfect shuffle*** of A and B be the language

$$\{w \mid w = a_1b_1 \cdots a_kb_k, \text{ where } a_1 \cdots a_k \in A \text{ and } b_1 \cdots b_k \in B, \text{ each } a_i, b_i \in \Sigma\}.$$

Show that the class of regular languages is closed under perfect shuffle.

1.32 For languages A and B, let the ***shuffle*** of A and B be the language

$$\{w \mid w = a_1b_1 \cdots a_kb_k, \text{ where } a_1 \cdots a_k \in A \text{ and } b_1 \cdots b_k \in B, \text{ each } a_i, b_i \in \Sigma^*\}.$$

Show that the class of regular languages is closed under shuffle.

1.33 Let A be any language. Define *DROP-OUT*(A) to be the language containing all strings that can be obtained by removing one symbol from a string in A. Thus, *DROP-OUT*$(A) = \{xz \mid xyz \in A \text{ where } x, z \in \Sigma^*, y \in \Sigma\}$. Show that the class of regular languages is closed under the *DROP-OUT* operation. Give both a proof by picture and a more formal proof by construction as in Theorem 1.47.

^{A}**1.34** Let B and C be languages over $\Sigma = \{\mathtt{0}, \mathtt{1}\}$. Define

$$B \overset{1}{\leftarrow} C = \{w \in B \mid \text{ for some } y \in C, \text{ strings } w \text{ and } y \text{ contain equal numbers of 1s}\}.$$

Show that the class of regular languages is closed under the $\overset{1}{\leftarrow}$ operation.