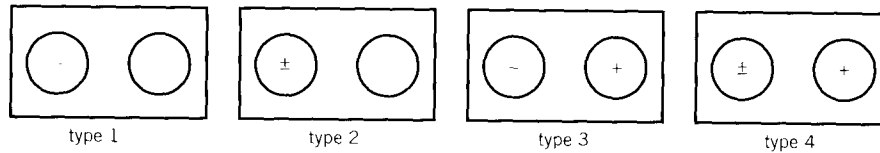


- (ii) Build an FA that accepts the language of all words with only a 's or only b 's in them. Give a regular expression for this language.
10. Consider all the possible FAs over the alphabet $\{a, b\}$ that have exactly two states. An FA must have a designated start state, but there are four possible ways to place the \pm 's:



Each FA needs four edges (two from each state), each of which can lead to either of the states. There are $2^4 = 16$ ways to arrange the labeled edges for each of the four types of FAs. Therefore, in total there are 64 different FAs of two states. However, they do not represent 64 nonequivalent FAs because they are not all associated with different languages. All type 1 FAs do not accept any words at all, whereas all FAs of type 4 accept all strings of a 's and b 's.

- (i) Draw the remaining FAs of type 2.
(ii) Draw the remaining FAs of type 3.
(iii) Recalculate the total number of two-state machines using the transition table definition.
11. Show that there are exactly 5832 different finite automata with three states x, y, z over the alphabet $\{a, b\}$, where x is always the start state.
12. Suppose a particular FA, called FIN, has the property that it had only one final state that was not the start state. During the night, vandals come and switch the $+$ sign with the $-$ sign and reverse the direction of all the edges.
- (i) Show that the picture that results might not actually be an FA at all by giving an example.
(ii) Suppose, however, that in a particular case what resulted was, in fact, a perfectly good FA. Let us call it NIF. Give an example of one such machine.
(iii) What is the relationship between the language accepted by FIN and the language accepted by NIF as described in part (ii)? Why?
(iv) One of the vandals told me that if in FIN the plus state and the minus state were the same state, then the language accepted by the machine could contain only palindromic words. Defeat this vandal by example.
13. We define a removable state as a state such that if we erase the state itself and the edges that come out of it, what results is a perfectly good-looking FA.
- (i) Give an example of an FA that contains a removable state.
(ii) Show that if we erase a removable state the language defined by the reduced FA is exactly the same as the language defined by the old FA.
14. (i) Build an FA that accepts the language of all strings of a 's and b 's such that the next-to-last letter is an a .
(ii) Build an FA that accepts the language of all strings of length 4 or more such that the next-to-last letter is equal to the second letter of the input string.

Problems

15. Build a FA that accepts all strings of length 6.
16. Build a FA that accepts all strings of length 4 or more.
17. Describe the language accepted by the FA in (i)

(ii)

(iii)

(iv) Wr

16. Let the language L be accepted by the transition graph T and let L not contain the word ba . We want to build a new TG that accepts exactly L and the word ba .
- One suggestion is to draw an edge from $-$ to $+$ and label it ba . Show that this does not always work.
 - Another suggestion is to draw a new $+$ state and draw an edge from any $-$ state to it labeled ba . Show that this does not always work.
 - What does work?

17. Let L be any language. Let us define the **transpose** of L to be the language of exactly those words that are the words in L spelled backward. If $w \in L$, then $\text{reverse}(w) \in L$. For example, if

$$L = \{a \quad abb \quad bbaab \quad bbbaa\}$$

then

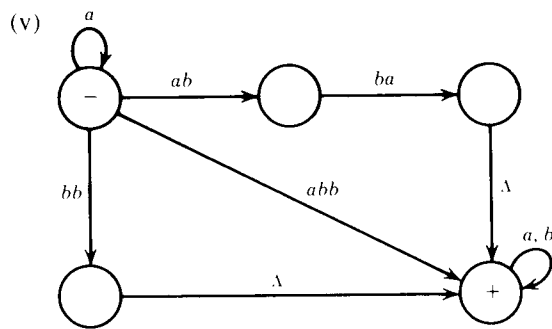
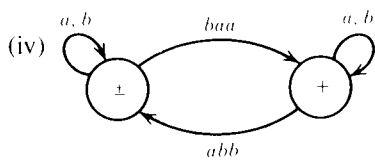
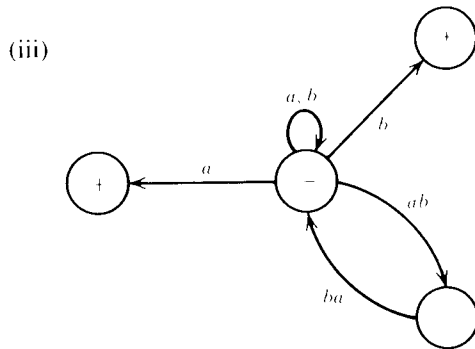
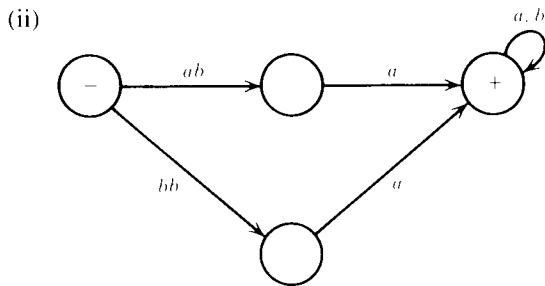
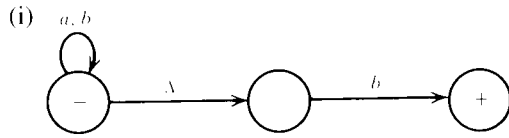
$$\text{transpose}(L) = \{a \quad bba \quad baabb \quad aabbb\}$$

- Prove that if there is an FA that accepts L , then there is a TG that accepts the transpose of L .
 - Prove that if there is a TG that accepts L , then there is a TG that accepts the transpose of L .
Note: It is true, but much harder to prove, that if an FA accepts L , then some FA accepts the transpose of L . However, after Chapter 7 this will be trivial to prove.
 - Prove that $\text{transpose}(L_1L_2) = \text{transpose}(L_2) \cdot \text{transpose}(L_1)$.
18. Transition graph T accepts language L . Show that if L has a word of odd length, then T has an edge with a label with an odd number of letters.
19. A student walks into a classroom and sees on the blackboard a diagram of a TG with two states that accepts only the word \mathbf{A} . The student reverses the direction of exactly one edge, leaving all other edges and all labels and all $+$'s and $-$'s the same. But now the new TG accepts the language \mathbf{a}^* . What was the original machine?
20. Let us now consider an algorithm for determining whether a specific TG that has no \mathbf{A} -edges accepts a given word:
- Step 1 Number each edge in the TG in any order with the integers $1, 2, 3, \dots, x$, where x is the number of edges in the TG.
 - Step 2 Observe that if the word has y letters and is accepted at all by this machine, it can be accepted by tracing a path of not more than y edges.
 - Step 3 List all strings of y or fewer integers, each of which $\leq x$. This is a finite list.
 - Step 4 Check each string on the list in step 3 by concatenating the labels of the edges involved to see whether they make a path from a $-$ to a $+$ corresponding to the given word.
 - Step 5 If there is a string in step 4 that works, the word is accepted. If none work, the word is not in the language of the machine.
 - Prove that this algorithm does the job.
 - Why is it necessary to assume that the TG has no \mathbf{A} -edges.

Rules 3 and 4 of Part 3 of Kleene's theorem can also be proven using Theorem 7, but this we leave for the problems section.

PROBLEMS

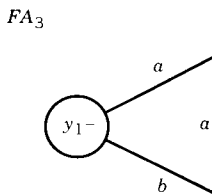
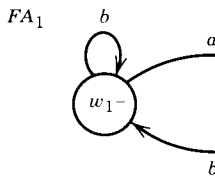
1. Using the bypass algorithm in the proof of Theorem 6, Part 2, convert each of the following TGs into regular expressions:



(vi)

2. In Chap...
Write a...
algorithm...
for rec...
cover h...

For Proble...



3. Using follow...

(i) F

(ii) F

(iii) F

4. Using follow...

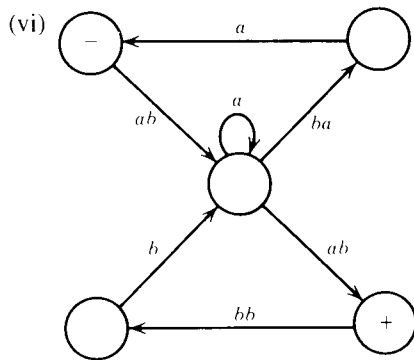
(i) F

(ii) F

(iii) F

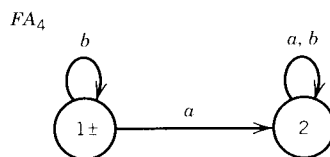
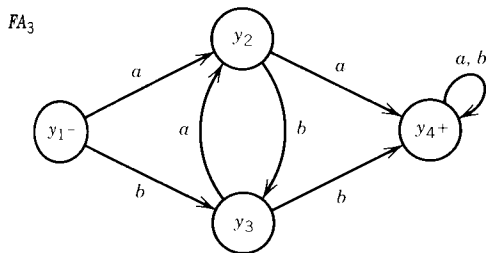
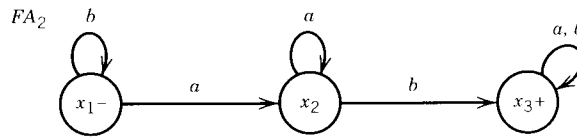
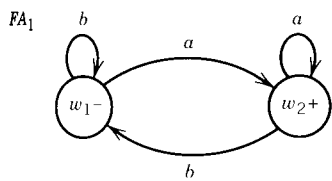
5. Using uct lar...

(i) F



2. In Chapter 5, Problem 10, we began the discussion of all possible FAs with two states. Write a regular expression for each machine of type 2 and type 3 by using the conversion algorithm described in the proof of Theorem 6, Part 2. Even though there is no algorithm for recognizing the languages, try to identify as many as possible in the attempt to discover how many different languages can be accepted by a two-state FA.

For Problems 3 through 12, use the following machines:



3. Using the algorithm of Kleene's theorem, Part 3, Rule 2, Proof 1, construct FAs for the following union languages:
- (i) $FA_1 + FA_2$
 - (ii) $FA_1 + FA_3$
 - (iii) $FA_2 + FA_3$
4. Using the algorithm of Kleene's theorem, Part 3, Rule 2, Proof 2, construct NFAs for the following languages:
- (i) $FA_1 + FA_2$
 - (ii) $FA_1 + FA_3$
 - (iii) $FA_2 + FA_3$
5. Using the algorithm of Theorem 6, Part 3, Rule 3, construct FAs for the following product languages:
- (i) $FA_1 FA_2$

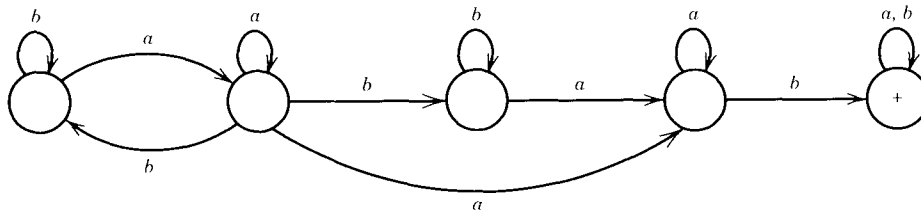
- (ii) $FA_1 FA_3$
- (iii) $FA_1 FA_1$
- (iv) $FA_2 FA_1$
- (v) $FA_2 FA_2$

6. Using the algorithm of Part 3, Rule 4, construct FAs for the following languages:

- (i) $(FA_1)^*$
- (ii) $(FA_2)^*$

7. We are now interested in proving Part 3, Rule 3, of Kleene's theorem by NFAs. The basic theory is that when we reach any + state in FA_1 , we could continue to FA_1 by following its a -edge and b -edge, or we could pretend that we have jumped to FA_2 by following the a -edge and b -edge coming out of the start state on FA_2 . We do not change any states or edges in either machine; we merely add some new (nondeterministic) edges from + states in FA_1 to the destination states of FA_2 's start state. Finally, we erase the + 's from FA_1 and the - sign from FA_2 and we have the desired NFA.

Let us illustrate this by multiplying FA_1 and FA_2 above.



Find NFAs for the following product languages:

- (i) $FA_1 FA_1$
- (ii) $FA_1 FA_3$
- (iii) $FA_2 FA_2$

8. Take the three NFAs in Problem 7 above and convert them into FAs by the algorithm of Theorem 7.

9. We can use NFAs to prove Theorem 6, Part 3, Rule 4, as well. The idea is to allow a nondeterministic jump from any + state to the states reachable from the - state by a - and b -edges.

- (i) Provide the details for this proof by constructive algorithm.
- (ii) Draw the resultant NFA for $(FA_1)^*$.
- (iii) Draw the resultant NFA for $(FA_2)^*$.
- (iv) Draw the resultant NFA for $(FA_3)^*$.

10. Convert the machines in Problem 9(ii) and (iii) above to FAs by the algorithm in the proof of Theorem 7.

11. Find FAs for the following languages:

- (i) $FA_4 FA_4$
- (ii) $(FA_4)^*$

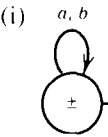
12. (i) Is the machine for $FA_1 FA_1$ (Problem 5) the same as the machine for $(FA_1)^*$ (Problem 6)? Are the languages the same?

(ii) Is the machine for $FA_4 FA_4$ the same as the machine for $(FA_4)^*$ (Problem 11)? Are the languages the same?

Problems

13. (i) For the ex...
chines with...
(ii) If some au...
what are t...
sponding to...
(a) By the...
(b) By bui...

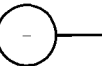
14. Convert each o...
in Proof 2 of T...



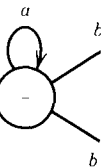
(ii)



(iii)



(iv)



(v)

