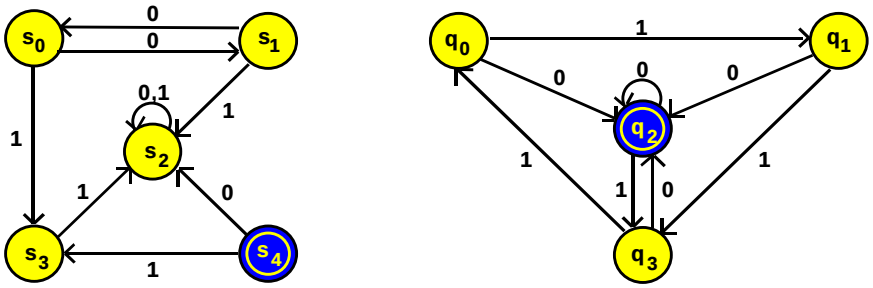
PROBLEMS

Finite Automata

1. Draw the state graphs for the finite automata which accept sets of strings composed of zeros and ones which:

   a) Are a multiple of three in length.
   b) End with the string 00.
   c) Possess runs of even numbers of 0's and odd numbers of 1's.

2. Describe the sets accepted by the finite automata pictured below.



3. Design a finite automaton that will control an elevator that serves three floors. Describe the states of the machine, intuitively present its operating strategy, and provide a state graph for your automaton.

4. Define formally and provide state tables for the finite automata that accept strings of zeros and ones which:

   a) Never contain three adjacent ones.
   b) Have a one as the next to last symbol.
   c) Contain an even number of zeros or an odd number of ones - not both!

5. String searching and pattern matching can be done easily by finite automata. Design a machine that accepts only strings containing 101 as a substring. Now do the same (design a machine) for the substring 00101.

6. Develop an algorithm to design finite automata for substring searching. The input should be the substring and the output is a finite automaton state table. Be sure to specify your data structures and intuitively describe the algorithm.

Closure Properties

1. Suppose that the finite automata M_i and M_k accept strings over the alphabet {0,1}. Design an automaton which accepts strings of the form x#y where x is accepted by M_i and y is accepted by M_k.

2. Prove that the class of sets accepted by finite automata is closed under intersection. In other words, given M_i and M_k construct the finite automaton M_m such that:

$$T(M_m) = T(M_i) \cap T(M_k)$$

3. Let x^R denote the *reversal* of the string x. (For example, if x = 010011 then x^R = 110010 and so forth.) Prove that the sets accepted by finite automata are closed under *string reversal* by constructing for any finite automaton, a new machine that accepts the reversals of the strings accepted by the original automaton.

4. Show that for each finite automaton, there is another machine that accepts only strings that are the front two thirds of the strings the first automaton accepted.

5. The *minus* operator on sets is usually defined as:

$$A - B = \{ x \mid x \in A \text{ and } x \notin B\}.$$

Prove that the class of sets accepted by finite automata is closed under minus.

6. Let x be a particular string. For arbitrary strings y and z such that z = yx, the *quotient* operator (/) can be defined:

$$z/x = yx/x = y.$$

(For example: 11010/10 = 110 and 11101/01 = 111.) This operator can be applied to sets as follows:

$$A/x = \{ y \mid yx \cup A\}.$$

(That is: $0^*110^*/10 = 0^*1$ and $0^*11(01)^*/11 = 0^*$.) Show that the class of sets accepted by finite automata is closed under quotient by constructing for any x and M_i, a machine M_k for which:

$$T(M_k) = T(M_i)/x.$$

7. *Set quotient* may be defined as:

$$A/B = \{ x \mid xy \in A \text{ for some } y \in B\}.$$

Show that the class of sets accepted by finite automata is closed under set quotient. That is, for M_i and M_k, design an M_m in such a way that:

$$T(M_m) = T(M_i)/T(M_k).$$

8. An *epsilon move* takes place when a finite automaton reads and changes state but does not move its tape head. (This is like a *stay* move for Turing machines.) Does this new operation add power to finite automata? Justify your answer.

Regular Sets and Expressions

1. What are the regular expressions for sets of strings composed of zeros and ones which:

   a) Are a multiple of three in length.
   b) End with the string 00.
   c) Possess runs (substrings) containing only even numbers of zeros and odd numbers of ones.

2. Derive the regular expressions for the sets accepted by the finite automata whose state graphs are pictured in the second problem of the first section.

3. Design finite automata that will accept the sets represented by the following regular expressions.

   a) $11(10 + 01)^*1^*01$
   b) $(0 + 11 + 01)^*0^*(01)^*$
   c) $(0 + 1)^*0^*101$

4. Show that the set of all binary integers that are the sum of exactly four (no more, no less!) positive squares is a regular set. (HINT: They are all found by substituting for m and n in the formula $4^n(8m + 7)$.)

5. Review the encodings of Turing machines from chapter one. Are these encodings a regular set? Discuss this in terms of nondeterministic Turing machines.

6. Derive and present the rules for determining *LAST* sets for regular expressions. Argue that they are correct.

7. Develop an algorithm for determining *FOLLOW* sets for any symbol in a regular expression. (You may assume that procedures for computing *FIRST* and *LAST* sets are available.)

Decision Problems for Finite Automata

1. Can finite automata accept sets of strings of the form:

   a) $0^n1^*[(0 + 11)^*(1 + 00)^*]^*0^*1^n$
   b) ww where w is a string of zeros and ones
   c) ww where w is a string of zeros

2. Can the following sets of strings be accepted by finite automata? Justify your answers!

   a) $\{ 1^n \mid n \text{ is a prime number} \}$
   b) $\{ 0^{2n}1^{2m} \mid n \text{ and } m \text{ are integers} \}$
   c) $\{ x \mid x \text{ is a binary power of two} \}$
   d) $\{ x \mid \text{the center symbol of } x \text{ is a } 1 \}$

3. Show that the regular sets are not closed under infinite union by producing an infinite family of regular sets whose union is not regular.

4. Consider a programming language in which only the following instructions occur.

```
x = 0
x = y
x = y + 1
repeat x
end
```

The symbols x and y stand for strings from a specified alphabet. A *correct program* is one which contains only the above instructions and in which an *end* eventually follows each *repeat*. *Nesting* is said to occur whenever two or more *repeat* instructions are encountered before reaching an *end*. The *depth of nesting* for a program is the number of consecutive *repeat* instructions.

Can the following sets of correct programs be accepted by finite automata?

a) Programs with depth of nesting no greater than two.
b) All correct programs.

5. Prove that every infinite regular set has an infinite regular subset.

6. Are all subsets of a regular set regular? Why?

7. Two states of a finite automaton are said not to be *equivalent* if there is a string which takes one into an accepting state and the other into a rejecting state. How many strings must be checked in order to determine whether two states are equivalent? Develop an algorithm for this.

8. Design an algorithm to determine whether a finite automaton accepts an infinite set. Prove that your algorithm is correct.

9. Exhibit an algorithm that detects whether one finite automaton accepts a subset of the set accepted by another machine. Show that this procedure works.

10. Examine the emptiness problem algorithm for finite automata. How much time does it require to analyze an automaton that has n states and uses m symbols?

Pushdown Automata

1. Design a pushdown automaton which accept strings of the form $1^*0^n1^n$ and one which accepts strings which contain twice as many zeros as ones.

2. Can pushdown automata accept sets of strings of the form:

   a) $0^n1^*[\,(0 + 11)^*(1 + 00)^*\,]^*0^*1^n$
   b) ww where w is a string of zeros and ones
   c) ww where w is a string of zeros

3. Prove that acceptance by empty stack is equivalent to accepting by final state for pushdown automata.

4. Provide pushdown machines that accept sets of strings composed of zeros and ones which are:

   a) of the form $1^n\,0^n$ or $1^n\,0^{2n}$.
   b) not of the form ww.

5. Consider pushdown automata that write output on separate one directional tapes (that is, they never go back to change any of what they have written). This basically means that they may write a string as part of each instruction. Design a machine that changes infix arithmetic expressions to postfix expressions.

6. Design a pushdown machine that generates output which will change postfix expressions into assembly language code.

7. Define pushdown automata with two stacks. Prove that they can simulate Turing machines.

8. When a pushdown machine executes an instruction and does not move its reading head, we say that it has made an *epsilon move*. Does this new capability add power to these automata? Why?

Unsolvable Problems for Pushdown Automata

1. Prove that the equivalence problem (whether two arbitrary machines accept the same set) for pushdown automata is unsolvable. (HINT: relate it to a pushdown machine problem you know is unsolvable.) Do the same for the set inclusion problem.

2. Show that whether or not a pushdown machine accepts everything but a finite set is unsolvable.

3. Design a pushdown automaton that accepts strings of the form x#y where x is a Turing machine configuration and y is the reversal of the configuration yielded by x. From this, develop two machines that accept sets whose intersection is a set of valid Turing machine computations.

4. Show that the problem of whether two pushdown automata accept sets with no elements in common is unsolvable.

Linear Bounded Automata

1. Demonstrate that multiheaded linear bounded automata are equivalent to those we defined.

2. Explain why multitrack linear bounded automata are equivalent to ordinary one track machines.

3. Design a linear bounded automaton which accepts strings of the form $0^n1^n0^n$.

4. Analyze the space and time complexity for the linear bounded automaton that accepted the set of squares.

5. Prove that linear bounded automata can accept sets of valid Turing machine computations.

6. Show that emptiness and finiteness are unsolvable for linear bounded automata.

7. Prove that equivalence is unsolvable for linear bounded automata.