Theory of Computation: Assignment 3

Arjun Chandrasekhar

Due Thursday, 02/10/2022 at 11:59 pm (50 points)

1. (5 points) This problem is taken from exercise 1.8a in Sipser. Let

$$A = \{w | w \text{ begins with a 1 and ends with a 0}\}$$

Let

$$B = \{w | w \text{ contains at least three 1s}\}$$

First, design a 3-state NFA to recognize A. Then design a 4-state NFA to recognize B. Finally, design a 8-state NFA to recognize $A \cup B$. In all cases the alphabet is $\Sigma = \{0, 1\}$.

2. (5 points) This problem is taken from exercise 1.9a in Sipser. Let

$$A = \{w | \text{ the length of w is at most 5}\}$$

Let

$$B = \{w | \text{ every odd position of w is a 1}\}$$

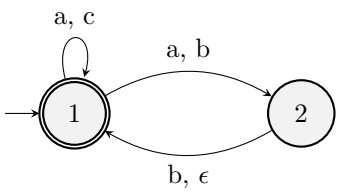
First, design a 6-state NFA to recognize A. Then, design a 2-state NFA to recognize B. Finally, design an 8-state NFA to recognize $A \circ B$. In all cases the alphabet is $\Sigma = \{0, 1\}$.

3. (5 points) This problem is taken from exercise 1.10b. Let

$$A = \{w | w \text{ contains at least two 0s and at most one 1}\}$$

First, design a 6-state NFA to recognize A. Then, design a 7-state NFA to recognize A^*. In both cases the alphabet is $\Sigma = \{0, 1\}$.

4. (5 points) This problem based from problem 1.16a in Sipser. Convert the following NFA to an DFA using the procedure described in class. The alphabet is $\{a, b, c\}$. For full credit your DFA must have all of the possible states and transitions.



5. (10 points) This problem is taken from problem 1.42 in Sipser. For languages A and B, the SHUFFLE operation is defined as follows:

$$\text{SHUFFLE}(A, B) = \{w = a_1 b_1 a_2 b_2 \dots a_n b_n | a_1 a_2 \dots a_n \in A, b_1 b_2 \dots b_b \in B, \text{ each } a_i, b_i \in \Sigma^*\}$$

At first glance this looks like PERFECT-SHUFFLE. The difference is that with SHUFFLE, each a_i, b_i is an entire word rather than a single letter. Put another way, with shuffle you don't have to alternate between A and B at each character; you can switch from one language to the other at any point in the String.

As an example, let

$$A = \{w | w \text{ is non-empty only contains 0s}\}$$

and let

$$B = \{w | w \text{ is non-empty and only contains 1s}\}$$

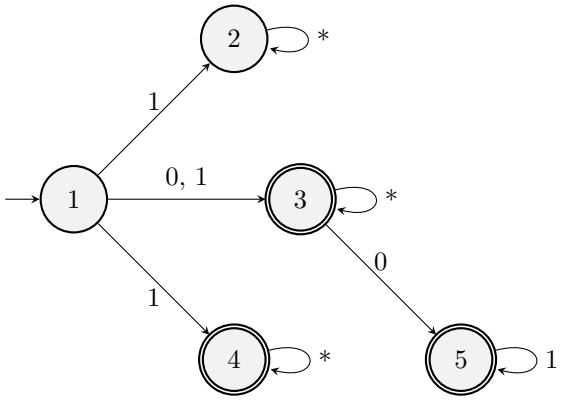
- 010101 is in both PERFECT-SHUFFLE(A, B) and SHUFFLE(A, B)
- 001011 $\in$ SHUFFLE(A, B), but 001011 $\notin$ PERFECT-SHUFFLE(A, B)
- 111111 is not in either PERFECT-SHUFFLE(A, B) or SHUFFLE(A, B) because no matter how you slice it up, the A part will contain 1s

Prove that regular languages are closed under the SHUFFLE operation.

Hint: for PERFECT-SHUFFLE, we created a DFA that alternated between each of the two machines after each character. For this problem you'll want to create an NFA that does something similar, but "guesses" when to alternate between machines.

6. This problem is based on problem 1.38 in Sipser. An **all-NFA** is a new type of automaton. Just like a regular NFA, an all-NFA is defined by a 5-tuple $(Q, \Sigma, \delta, q_s, F)$. It performs computation in the same way. The only difference is the acceptance criterion: the all-NFA accepts if *all* possible computation paths lead to an accept state (we ignore computations that "die"). This is in contrast to a regular NFA, which just needs a single accepting computation to accept the string.

   As an example, consider the following state macine:



   - If this were an NFA, the string 111 would be accepted, because there exists a computation path that leads to an accept state. However, if this were an all-NFA the string 111 would be rejected. The computation can end in states 2, 3, 4, or 5 depending on what choices the all-NFA makes. Because there exists a computation path that leads to a reject state, the all-NFA would reject 111.
   - The string 001 is accepted, even if this is an all-NFA. The computation can end in states 3 or 5 depending on what choices the all-NFA makes. Each of these states is an accept state, so the all-NFA accepts 011. Note that the all-NFA cannot reach state 4, which is an accept state. However, this does not cause the all-NFA to reject. We don't require the all-NFA to be able to reach every accept state - we simply require that all of the potential final states are accept states.

- The string 000 is accepted even if this is an all-NFA. The computation can end only end in states 3 or 5, both of which are accept states. Note that it is possible for the all-NFA to move to state 5 and then die on the third character. However, we ignore computation paths that die.

An All-NFA is similar to a regualar NFA, however it is not exactly the same; its definition is not identical. Thus, you cannot assume that it inherits every single property of a regular NFA.

(a) (10 points) Prove that a language L is regular if and only if L is recognized by an all-NFA. Remember, there are two directions. Give a formal description for any machine you construct.

(b) (10 points) Use all-NFAs to prove that regular languages are closed under intersection - that is, prove that if A and B are regular, then

$$A \cap B = \{w | w \in A, w \in B\}$$

is regular. For full credit your proof must make use of an all-NFA. You will not receive any credit if you attempt to use De Morgan's laws. You will also not receive any credit if you attempt to use the Cartesian product construction to run two DFAs in parallel. Your proof must take advantage of the unique properties of an all-NFA.

7. (10 points, extra credit) This problem is borrowed from Dr. John Glick at the University of San Diego. Imagine Snoopy and Charlie are sitting at a table. On the table is a square tray with four glasses at the corners. Charlie's. goal is to turn all the glasses either right-side up or upside down. However, Charlie is blindfolded and he is wearing mittens. He does not know the initial state of the glasses. If they are initially all turned the same way, then Charlie automatically wins. In his turn, Charlie may grab one or two glasses and turn them over; however, because of the blindfold and the mittens he cannot see or feel whether the glasses he grabbed are right-side up or upside down. He can, however, choose whether to grab adjacent glasses or diagonally opposite glasses (or just one glass). If the glasses are all turned the same direction, Snoopy announces that Charlie has won. Otherwise, Snoopy may rotate the tray, just to make Charlie's goal harder.

   Find the shortest sequence of actions by Charlie that is guaranteed to win the game, no matter how Snoopy plays. Prove that your solution is correct and is the shortest possible.

   First construct an all-NFA to represent the game. Then covert it to a DFA that is equivalent. Use this to find the shortest sequence of moves, and to prove that your sequence is optimal. If you just write down a sequence of moves, you will not get credit.

   (**Hint:** Your all-NFA states should represent different "configurations" of the cups. Some configurations are equivalent to each other, because Charlie can't tell the difference between up/down and because Snoopy can rotate the glasses. These equivalent configurations can be combined into the same state, to reduce the size of your all-NFA.)