CSE 105 Homework 1

Due January 15

• You may (and are encouraged to) discuss the problems with other students, but what you hand in must be your own work.

• If you do not understand a problem or find a mistake, please email William at wmatthews@cs.ucsd.edu.

• In addition to these problems, we recommend doing Sipser problems 1.4, 1.5, 1.7, 1.11, 1.16, 1.31, 1.32, 1.40 for more practice. However, you should not hand in your solutions to these problems.

• If you give an NFA or DFA as an answer, remember to give a correctness argument explaining why the NFA or DFA is correct.

• Typed homework solutions will receive 5% extra credit.

1 Set Review

Consider the sets $A = \{1, 2, 3\}$, $B = \{2, 4, 6\}$, and $C = \{1, 3\}$.

a) What is $A \cap C$?
   
   $A \cap C = \{1, 3\}$

b) What is $B \cup C$?

   $B \cup C = \{1, 2, 3, 4, 6\}$

c) What is the power set of $B$?
   
   $\mathcal{P}(B) = \{\emptyset, \{2\}, \{4\}, \{6\}, \{2, 4\}, \{2, 6\}, \{4, 6\}, \{2, 4, 6\}\}$

d) What is $A \times C$?
   
   $A \times C = \{(1,1), (1,3), (2,1), (2,3), (3,1), (3,3)\}$

e) List all of the ways that we can correctly fill in the blanks in $\ldots \subseteq \ldots$ and $\ldots \subset \ldots$ with the sets $A$, $B$, and $C$.

   $C \subseteq A$
   $A \subseteq A$
   $B \subseteq B$
   $C \subseteq C$
2 Constructing a DFA

Let $\Sigma = \{0, 1\}$ and consider the language $L$ consisting of strings $x \in \Sigma^*$ which satisfy the following three properties:

- $x$ has no leading 0's.
- When viewed as a binary number, $x$ is even.
- When viewed as a binary number, $x$ is not divisible by 4.

For example, $10010 \in L$ since it has no leading 0’s and is the binary representation of 18 which is even and not divisible by 4. On the other hand, 00010, 101, and 100000 are not in $L$ because they violate the first, second, and third conditions respectively.

Give a DFA that recognizes $L$. This can be done with a DFA with 5 states, but you should make sure that yours has no more than 10 states.

2.1 Solution

For a string to satisfy the 3 properties it must begin with a 1 and end with 10. If it doesn’t begin with a 1 then the first property is violated. The second property says that the string must end with a zero. The third property says that the string cannot end with 00, thus we get from the second and third properties that the string must end with 10.

Note that the 1 which must begin the string may be the same as the 1 in the final 10, but it doesn’t have to be. A regular expression for $L$ is $10 \cup 1\Sigma^*10$.

2.1.1 DFA

![DFA Diagram]

2.1.2 Correctness

We will give an invariant for each state and then show that the invariants must hold based on the transitions in the DFA. The only accepting state is $d$, and the invariant for $d$ is exactly the condition needed for a string to be in $L$.

a) The DFA is only in state $a$ before reading any symbols. $a$ is the start state and there are no transitions going to $a$.

r) This the reject state for strings that begin with a 0. The only transitions to this state are from state $a$ (implying no symbols have been read) on a 0, and from this state on any symbol (implying that the string already started with a 0).

b) The DFA is in this state if the string begin with a 1 and if the last symbol read was a 1. If the string began with a 0, then the machine is in state $r$, otherwise reading a 1 from any state transitions the DFA to $b$. 


d) The DFA is in state $d$ if the string began with a 1 and the last two symbols read were 1 followed by 0. This is exactly the set of conditions needed to be in the language $L$, so $d$ is an accepting state. The only transition to $d$ is from $b$ on a 0, so the invariant for $d$ follows from the invariant on $b$ and this transition.

c) The DFA is in state $c$ if the string began with a 1 and the last symbols read were a 1 followed by at least two 0's. If $c$ is reached from $d$, then the invariant holds since the last 3 symbols read must be 1,0,0. If $c$ is reached from $c$ then the invariant holds because it must have previously held at $c$ and then another 0 was read.

3 Constructing an NFA

Let $\Sigma = \{a, b\}$ and consider the language $L = a^*b \cup b^*$

a) Give an NFA that recognizes $L$. Your NFA should have at most 4 states.

b) Give a DFA that recognizes $L$. Your DFA should have at most 16 states.

3.1 Solution

3.1.1 NFA

![NFA Diagram]

3.2 NFA Correctness

Just as we did for problem 2, we will give an invariant for each state. The invariants on accepting states imply that the string read thus far is in $L$.

w) The NFA is only in this state if no symbols have been read.

x) The NFA is in this state if the string thus far consists of zero or more $a$’s. If we come from state $w$ then 0 $a$’s have been read, if we come from $x$ then at least one $a$ has been read. In either case, no $b$’s have been read.

y) The NFA is in this state if the string thus far consists of zero or more $b$’s. This means that the string is in $b^*$ and thus in $L$ so we accept in this state. If we get to $y$ from $w$ then 0 $b$’s have been read, otherwise we come from $y$ on a $b$ and the invariant must have held previously.

z) The NFA is only in this state if the string thus far consists of zero or more $a$’s followed by a $b$. This means the string is in $a^*b$ and thus in $L$ so we accept in this state. The only way to get to $z$ is from $x$ on a $b$, so this invariant follows from the invariant on $x$. 
3.3 DFA

The correctness of the DFA follows from the correctness of the NFA and the correctness of the NFA to DFA construction given in Sipser Thm. 1.39. Unreachable states generated by the construction were removed.

4 Closure

Let $A$ be a language. Define the operation $splittable$ as follows:

$$A^{splittable} = \{ w_1w_2 \mid w_1 \in A, w_2 \in A, \text{ and } w_1w_2 \in A \}$$

Show that the class of regular languages is closed under the $splittable$ operation.

4.1 Solution

If $A^{splittable}$ were defined as $\{ w_1w_2 \mid w_1 \in A, w_2 \in A \}$ then it would be equivalent to $A \circ A$. Instead, we have the additional condition that every string in $A^{splittable}$ is also in $A$. Thus, $A^{splittable}$ is exactly the set of string that are both in $A \circ A$ and in $A$, or equivalently $A^{splittable} = (A \circ A) \cap A$.

If $A$ is regular then $A \circ A$ is regular (the concatenation of two regular languages is regular, Sipser Thm. 1.26). It then follows that $(A \circ A) \cap A$ is regular (the intersection of two regular languages is regular, Sipser p. 46, footnote 3).

5 More Regular Languages

Let $\Sigma = \{0, 1\}$. Consider the following language:

$$C = \{ a_1b_1a_2b_2 \ldots a_kb_k \mid a_i, b_i \in \Sigma \text{ and when viewed as binary numbers, } |a_1a_2\ldots a_k - b_1b_2\ldots b_k| = 1 \}$$

For example, $10010101 \in C$ since $|1000 - 0111| = 1$, but $11000100 \not\in C$ since $|1000 - 1010| = 2$.

Show that $C$ is regular. Hint: This problem is similar to problems 1.32-1.34 in Sipser.

5.1 Solution

We will prove that $C$ is regular in 2 stages. First we will prove that $C = (00 \cup 11)^*10(01)^* \cup (00 \cup 11)^*01(10)^* = (00 \cup 11)^*(10(01)^* \cup 01(10)^*)$, and then we will give an NFA for this regular expression. (The existence of a regular expression for $C$ proves that $C$ is regular, but we give an NFA since this result may not have been covered yet.)
5.1.1 Regular Expression

The regular expression \((00 \cup 11)^*10(01)^*\) corresponds to the case where \(a_1a_2\ldots a_k - b_1b_2\ldots b_k = 1\) and \((00 \cup 11)^*01(10)^*\) corresponds to \(a_1a_2\ldots a_k - b_1b_2\ldots b_k = -1\). We will only prove the first case \(((00 \cup 11)^*10(01)^*)\), the second case follows from a nearly identical argument.

Strings in \(C\) consist of pairs of bits \(a_i, b_i\). These strings may begin with pairs 00 or 11 since in both of these cases \(a_i\) and \(b_i\) cancel out when subtracted. Strings in \(C\) must have a pair of bits that differ, otherwise the difference in binary values would be 0. Let \(j\) designate the index of that pair. Since we are just considering the case where \(a_1a_2\ldots a_k - b_1b_2\ldots b_k = 1\), \(a_j\) must be 1, and \(b_j\) must be 0 (otherwise the \(b\) number in binary would be larger than the \(a\) number). Pairs after index \(j\) must be of the form 01. There may be 0 or more such pairs. If they are of this form, then the difference remains 1. If a pair after index \(j\) is not of the form 01, then the binary difference will be strictly greater than 1 (either an \(a\) bit goes from 0 to 1, or a \(b\) bit goes from 1 to 0.

5.1.2 NFA

![NFA Diagram]

States \(a\), \(b\), and \(c\) form the part of the NFA to match \((00 \cup 11)^*\). The NFA will be in state \(a\) after matching such a string.

Pairs of states \(e\) and \(f\) and \(h\) and \(i\) match the strings 10 and 01 respectively. The \(\epsilon\) transition from \(a\) to \(d\) concatenates the part of the NFA which matches 10 or 01 after the part that matches \((00 \cup 11)^*\). When the NFA reaches state \(f\), it will must have matched \((00 \cup 11)^*10\), and when it reaches state \(i\), it must have matched \((00 \cup 11)^*01\).

State \(g\) allows strings \((01)^*\) to follow the strings matching \((00 \cup 11)^*10\) which reached state \(f\). Combining the two, strings which reach state \(f\) are exactly those which match \((00 \cup 11)^*10(01)^*\).

An identical argument holds for states \(i\) and \(j\), establishing that strings which reach state \(i\) are exactly those which match \((00 \cup 11)^*01(10)^*\).

To combine everything, the regular expression is equivalent to \(C\), and the NFA is equivalent to the regular expression. Thus \(C\) must be regular.