1. (Sipser 1.18 f, i, and l) Give regular expressions generating the languages described in parts f, i, and l of Sipser’s exercise 1.6.

   f) \( \{ w \mid w \text{ does not contain the substring } 110 \} \)
   
   i) \( \{ w \mid \text{every odd position of } w \text{ is a } 1 \} \)
   
   l) \( \{ w \mid w \text{ contains an even number of } 0\text{s, or contains exactly two } 1\text{s } \} \)

2. Produce a regular expression describing the language accepted by the NFA shown below using the algorithm based on generalized nondeterministic finite automata presented in Sipser (but without showing transitions labeled by the regular expression ∅). As you execute the algorithm, remove states from the original machine in the following order: first C, then A, and finally B. Show each of the steps in the process by drawing a diagram for each of the GNFA’s produced in the process of decreasing the number of states to 2. Show the final regular expression you obtain.

3. At this point, we have seen three notations for describing regular languages: deterministic finite automata, nondeterministic finite automata and regular expressions. Before you learned about regular expressions, we used finite automata to prove in class that if \( L \) was a regular language, then the language \( L^R \) formed by reversing all the strings in \( L \) must also be a regular language.

   Imagine for a moment that instead of presenting automata first and then introducing regular expressions, we had instead first presented regular expressions and then described finite automata. If we had taken this approach then at some point all you would know is that regular languages are the languages described by regular expressions. Suppose that in this situation, you still wanted to prove that for all regular languages \( L \), \( L^R \) must be regular.

   Show how you could prove this result by providing a procedure that given any regular expression \( e \) produces a new regular expression \( e' \) in such a way that if \( L = L(e) \), then \( L^R = L(e') \). Justify the correctness of your transformation.

4. For any language \( L \) over alphabet \( \Sigma \), consider the derived languages

\[
L_{−\frac{1}{2}−} = \{ w \mid \text{for some } x, y \in \Sigma^*, |x| = |y| = |w| \text{ and } xyw \in L \}
\]

\[
L_{−\frac{1}{2}−} = \{ w \mid \text{for some } x, y \in \Sigma^*, |x| = |y| = |w| \text{ and } xwy \in L \}
\]
$L_{\frac{1}{3}-\frac{1}{3}} = \{ w \mid \text{for some } x, y, z \in \Sigma^*, |x| = |y| = |z|, w = xz \text{ and } xyz \in L \}$

Regular languages are closed under at least one of these three transformations. At the same time, regular languages are not closed under at least one of these transformations. Identify one of the transformations that preserves regularity and justify your claim that regular languages are closed under this transformation by providing a formal description of how to transform a DFA, $M$, such that $L = L(M)$ into a new automaton (probably a NFA) $M'$ that recognized the transformed language ($L_{\frac{1}{3}-\frac{1}{3}}$, $L_{\frac{1}{3} - \frac{1}{3}}$, or $L_{\frac{1}{3}-\frac{1}{3}}$). Provide a brief but complete explanation of the intuition behind your construction.

5. Complete problem 1.66 from Sipser:

A homomorphism is a function $f : \Sigma \to \Gamma^*$ from one alphabet to strings over another alphabet. We can extend $f$ to operate on strings by defining $f(w) = f(w_1)f(w_2)\ldots f(w_n)$, where $w = w_1w_1\ldots w_n$ and each $w_i \in \Sigma$. We further extend $f$ to operate on languages by defining $f(A) = \{ f(w) \mid w \in A \}$, for any language $A$.

a. Show, by giving a formal construction, that the class of regular languages is closed under homomorphism. In other words, given a DFA $M$ that recognizes $B$ and a homomorphism $f$, construct a finite automaton $M'$ that recognizes $f(B)$. Consider the machine $M'$ that you have constructed. Is it a DFA in every case?

b. Show, by giving an example, that the class of non-regular languages is not closed under homomorphism.

I am assigning this problem with two goals in mind. First, it asks you to prove a very handy result. With this result in hand, you will have a tool that lets you conclude that languages that “look the same” like $\{a^n b^n \mid n \geq 0 \}$ and $\{0^n 1^n \mid n \geq 0 \}$ must either both be regular or both be non-regular with justification and confidence.

Second, it is a problem where the focus is on finding a way to convert a simple intuitive construction into a clearly presented formal description in a situation where finding the right formalism can be challenging. To make sure your focus is on the formalism, I want to give you the intuition.

The basic idea you need to solve this problem is that $M'$ will have all of the states included in $M$ plus a pile of extra states that will be used to form chains of transitions that will replace the direct state to state transitions included in $M$. In particular, if there is a transition from $q_a$ to $q_b$ on input $x$ in $M$ and the homomorphism $f$ maps $x$ to $w_1w_2\ldots w_n$, you “might” specify in the transition function for $M'$ that there is a transition on $w_1$ from $q_a$ to some new state and that from that new state the only transition allowed is on $w_2$ to yet another new state and so on until you
end with a transition on $w_n$ from yet another new state to $q_f$. This will involve quite a lot of new states. If $f(x) = w_1w_2 \ldots w_n$ you will be adding “approximately”\(^1\) \(n\) new states for every transition on $x$ in the original machine $M$.

In case it is not clear, the preceding description of how to build $M'$ would not be sufficient to serve as the “formal construction” that Sipser requests in the statement of his problem. Providing that formalism is your job. Your construction should include precise, formal definitions of the set of states and final states of $M'$ and its transition function $\delta'$.

---

\(^1\)I have quoted “might” and “approximately” because there are small variations on the chains of transitions I have described informally that are perfectly reasonable but might or might not $\epsilon$-transitions and might involve anywhere from $n - 1$ to $n + 1$ new states to build a chain of transitions for $w_1w_2 \ldots w_n$. 

Due: 12:00. September 30, 2016