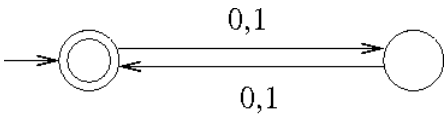
CSE 2001 First test - version 1
Fall 2012
Solutions

1. (3 points) Draw a DFA that accepts the language consisting of all words over the alphabet $\Sigma = \{0, 1\}$ of even length. Add one or two sentences to your drawing to explain your design.

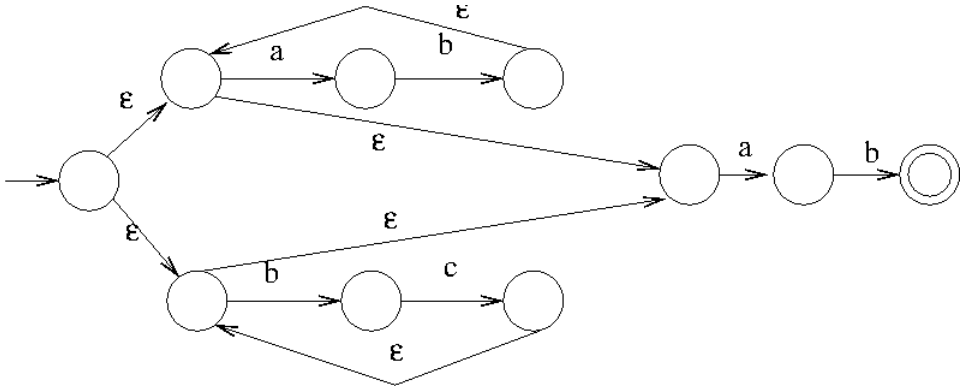
   Solution:



   As shown in the figure, the automaton has 2 states. The automaton is in the start state after seeing strings of even length (including zero length) and is in the other state after seeing an input of odd length. Thus the start state is the only accept state.

2. (3 points) Draw a NFA that accepts the language corresponding to the regular expression $((ab)^* \cup (bc)^*)ab$. You can do this by following the steps outlined in the book to convert regular expressions to finite automata. If you do not follow these steps then add one or two sentences to explain why your automaton is correct.

   Solution:



3. (3 points) Prove that for any 2 languages L_1, L_2, if $L_1 \subseteq L_2$ then $L_1^* \subseteq L_2^*$.

   Solution: Since $L_1 \subseteq L_2$, every word $w \in L_1$ also satisfied $w \in L_2$. By definition a word $v \in L_1^*$ is of the form $v = w_1 w_2 \ldots w_k$ where $k \geq 0$ and $w_i \in L_1$ for $0 \leq i \leq k$. Since all these $w_i \in L_2$, by definition of the Kleene-* operation, $v \in L_2$. Therefore $L_1^* \subseteq L_2^*$.

   Note: This question is about languages in general. It is not meaningful to talk about automata for this question. Further, the epsilon-closure of regular languages has no implications for this question.

   Some students wanted to express L^* in terms of concatenations of L. While that is a meaningful characterization, it makes the Math harder. Observe that $L^* = \cup_i^\infty L^k$. In order to prove this question you need to do an inductive proof that is far more complicated than my solution above.

4. (3 points) Suppose a language $L \subseteq \{a, b\}^*$ is defined recursively as follows.
   $\epsilon \in L$, and for every $x \in L$, both ax and axb are elements of L.
   Prove using induction that every string in L is in $L_0 = \{a^i b^j | i \geq j\}$.
   Note: Actually $L = L_0$ but you only have to prove that $L \subseteq L_0$ in this question.

   Solution: In any inductive proof, it is important to state the hypothesis precisely. We will carry out the induction on the length $|x|$ of strings x in the language L.

   Base case: $|x| = 0$. The hypothesis holds since the only string of length zero is ϵ and we are given that it is in L. It is also in L_0 by the definition of L_0 (at least as many a's as b's).

Inductive Step: We assume that all string of length less than or equal to n in L is also in L_0. Consider any string $s \in L$ such that $|s| = n + 1$. The only ways s could have been produced is as $s = ax$ for some $x \in L$ or $s = ayb$ for some $y \in L$. By the inductive hypothesis, any such x, y are in L and hence have at least as many a's as b's. Since ax increases the count of a's by 1 and ayb increases the count of both a's and b's by 1, s still has at least as many a's as b's and is therefore in L_0.

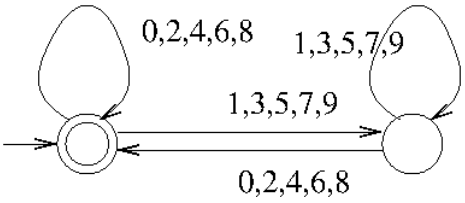
Therefore, by the principle of mathematical induction every string in L is also in L_0, i.e., $L \subseteq L_0$

5. (3 points) Consider the set of all even numbers. Assume that the numbers are represented in the normal decimal format (i.e. base 10). Design an automaton to accept this language.

   Note that the decimal format does not imply that there is a decimal point; we are dealing with integers.

   Solution: In this question, the alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Let us first assume that the language includes ϵ - the question does not explicitly specify if it does.

   We can choose whether to feed the input in from left to right or right to left. Let us assume that we will input the number starting with the most significant digit. In that case, the automaton should accept if and only if the last digit input was even (i.e., 0,2,4,6 or 8).



   As shown in the figure, the automaton has two states, one to remember that the last digit seen was odd and the other to remember that the last digit shown was even.

   You would get full credit for this. If you want to remove ϵ from the accepted language, you would add a new start state with transitions to the accept state on even digit inputs and transitions to the non-accept state on odd digit inputs.

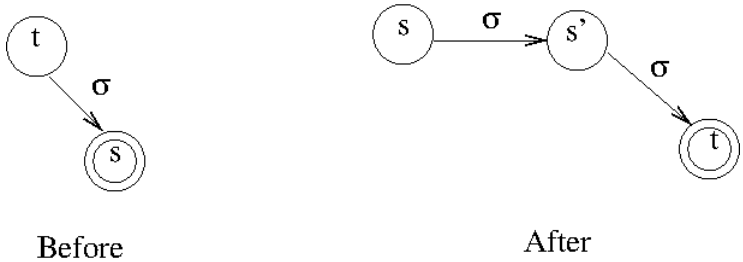
6. (3 points) Suppose L is a regular language. Design a DFA or NFA (not GNFA) for the language

$$L' = \{w\sigma^2 | w\sigma \in L, \sigma \in \Sigma\},$$

   which is the language obtained from L by repeating the last letter of each word w in L with non-zero length, i.e., $|w| \geq 1$.

   Solution:

   We will modify the DFA M that accepts the regular language L. Consider each transition to an accepting state $t \in F$ from another state $s \in Q$. Let $\sigma \in \Sigma$ be the label on this arrow. Then we replace this transition with two transitions and a new state s' as shown in the figure.



   Note that this creates a NFA. This would get you full credit but it is still not complete. What if strings $w\sigma$ and $w\sigma\alpha$ are accepted by the original automaton? The new automaton should accept $w\sigma\alpha\alpha$ and the above contruction would not accept that string. So the above idea has to be carried out iteratively. Whenever you add a new state s' it shoul inherit the outgoing transitions of t in addition to the σ transition to t. Then the procedure has to be repeated to check if the new transitions created are to accept states.

7. (3 points) Consider a regular language L. Design a finite automaton that accepts the set of subsequences of all words in L. The word w_2 is a subsequence of a word w_1 if $w_1 = \sigma_1\sigma_2 \ldots \sigma_k$ for some $\sigma_1, \sigma_2, \ldots, \sigma_k \in \Sigma$ and $w_2 = \sigma_{1_1}\sigma_{i_2} \ldots \sigma_{i_l}$ where $1 \leq i_1 < i_2 < \ldots < i_l \leq k$.

   Solution:

   Since a subsequence is formed by leaving out zero or more characters from a string, we see that some or all edges on an accepting path can be removed and the automaton should still reach an accept state. The easiest way to achieve this is by adding an epsilon transition between every pair of nodes that are connected by an edge in the given automaton.

   Thus any substring of a string s would take the automaton through the same sequence of states as s, except that it would use the epsilon transitions to handle the skipped characters.