CS 273: Intro to Theory of Computation, Spring 2008

Problem Set 3 (due Monday, February 4th, 4pm)

This homework contains five problems. As usual, please submit each problem on a **separate sheet of paper**. Turn in your homework at Elaine Wilson's office (3229 Siebel).

1. Building and interpreting regular expressions.

   Give regular expressions for the following languages over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.

   (a) $L_a = \left\{ w \;\middle|\; |w| \text{ is a multiple of 3 and } w \text{ starts with } \mathtt{aa} \right\}$.

   (b) $L_b = \left\{ w \;\middle|\; w \text{ contains the substring } \mathtt{bbb} \text{ or } \mathtt{bab} \right\}$.

   (c) $L_c = \left\{ w \;\middle|\; w \text{ does not contain two consecutive } \mathtt{b}\text{'s} \right\}$.

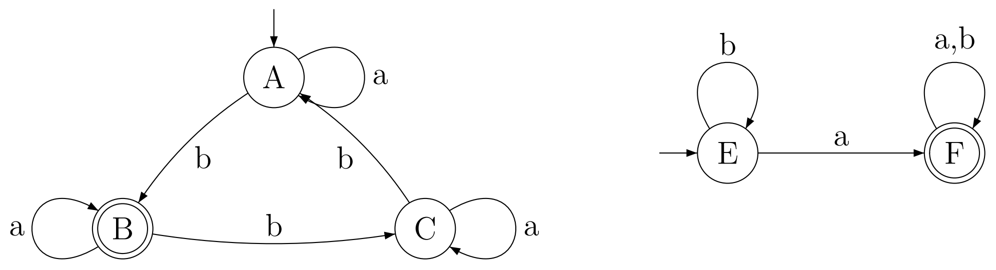
   Describe the languages represented by the following regular expressions:

   (d) $(a \cup b \cup \epsilon)((a \cup b)(a \cup b)(a \cup b))^*(a \cup b \cup \epsilon)$.

   (e) $(aa \cup bb \cup ab \cup ba \cup \epsilon)^+$

2. Product construction

   (a) Consider the following two DFAs. Use the product construction (pp. 45–47 in Sipser) to construct the state diagram for a DFA recognizing the intersection of the two languages.



   (b) When the product construction was presented in class (and in Sipser), we assumed that the two DFAs had the same alphabet. Suppose that we are given two DFAs M_1 and M_2 with different alphabets. E.g. $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$. To build a DFA M that recognizes $L(M_1) \cup L(M_2)$, we need to add two additional sink states s_1 and s_2. We send the first or the second element of each pair to the appropriate sink state if the incoming character is not in the alphabet for its DFA.

Write out the new equations for M's transition function δ and its set of final states F.

3. Algorithms for modifying DFAs.

Suppose that $M = (Q, \Sigma, \delta, q_0, F)$ and $N = (R, \Sigma, \gamma, r_0, G)$ are two DFAs sharing the common alphabet $\Sigma = \{a, b\}$.

(a) Define a new DFA $M' = (Q \cup \{q_X, q_R\}, \Sigma \cup \{\#\}, \delta', q_0, \{q_X\})$ whose transition function is defined as follows

$$\delta'(q,t) = \begin{cases} \delta(q,t) & q \in Q \text{ and } t \in \Sigma \\ q_X & q \in F, t = \# \\ q_X & q = q_X \\ q_R & \text{otherwise.} \end{cases}$$

Describe the language accepted by M' in terms of the language accepted by M.

(b) Show how to design a DFA N' which accepts the language

$$L' = \left\{ ttw \;\middle|\; (t = a \text{ and } w \in L(M)) \text{ or } (t = b \text{ and } w \in L(N)) \right\}.$$

Define your DFA using notation similar to the definition of M' in part (a).

4. Shared channel.

Funky Computer Systems, who have now gone out of business, submitted the lowest bid for wiring the DFAs supporting the Siebel center classrooms. These idiots wired two of the DFAs M and N so that their inputs come in on a shared input channel. When you try to submit a string w to M and a string y to N, this single channel receives the characters for the two strings interleaved. For example, if $w =$ `abba` and $y =$ `cccd`, then the channel will get a string like `abbcccad` or `acbccbad`.

Fortunately, these two DFAs have alphabets that do not overlap, so it's possible to sort this out. Your job is to design a DFA that accepts a string on the shared channel exactly when M and N would have accepted the two input strings separately.

Specifically, let $M = (Q, \Sigma, \delta, q_0, F)$ and $N = (R, \Gamma, \gamma, r_0, G)$, where $\Sigma \cap \Gamma = \emptyset$. Your new machine M' should read strings from $(\Sigma \cup \Gamma)^*$. It should be designed using a variation on the product construction, i.e. its state set should be $Q \times R$.

Give a formal definition of M' in terms of M and N. Also briefly explain the ideas behind how it works (very important especially if your formal notation is buggy).

5. Union in/out game.

A palindrome is a string that if you reverse it, remains the same. Thus `tattarrattat`[1] is a palindrome. You could consider the empty string ϵ to be a palindrome. But, for this problem, let's consider only strings containing at least one character.

(a) Let L_k be the language of all palindromes of length k, over the alphabet $\Sigma = \{a, b\}$. Show a DFA for L_4.

(b) For any fixed k, specify the DFA accepting L_k.

(c) Let L be the language of all palindromes over Σ. Argue,as precisely as you can, that L is not regular.

(d) We can express the language L as $\cup_{k=1}^{\infty} L_k$. It's tempting to reason as follows: the language L is the union of regular languages, and as such it is regular.

What is the flaw in this argument, and why is it *wrong* in our case?

[1]Which is the longest palindromic word in the Oxford English Dictionary is tattarrattat, coined by James Joyce in Ulysses for a knock on the door.