Problem Set 3

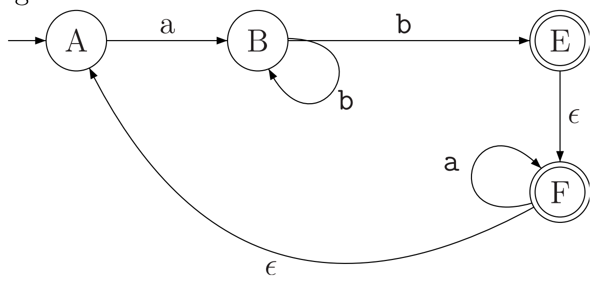
Due: Thursday, February 19, 2009 at 12:30 in class (i.e., SC 1105)
Version: **1.1**

Submission instructions: Submit each problem on a **separate sheet of paper**, put your name on each sheet, and write your discussion section time and day (e.g. Tuesday 10am) in the upper righthand corner. These details may sound picky, but they make the huge pile of homeworks much easier to grade quickly and more importantly, since we return them in the discussion sections, easier for you to get them back.

Also, write on each exercise the name/netid of your group members.

1. NFA interpretation.
[**Category**: Notations., **Points**: 10]

Consider the following NFA M.

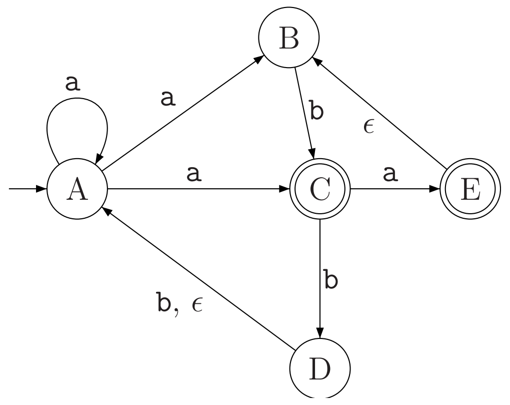


(a) Give a regular expression that represents the language of M. Explain briefly why it is correct.

Note: You needn't go through the process of converting this to a regular expression using GNFAs; you can guess (correctly) the regular expression.

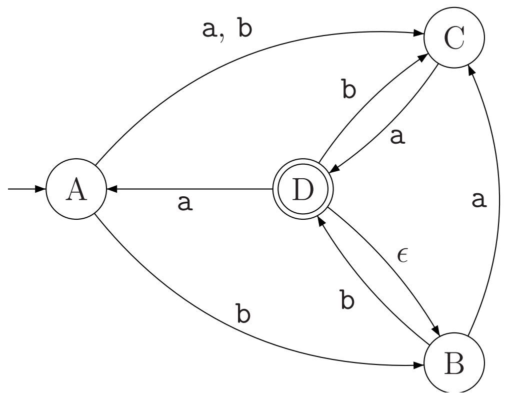
(b) Recall the definition of an NFA accepting a string w (Sipser p. 54). Show formally that M accepts the string $w =$ abba

(c) Let $\Sigma = \{a, b\}$. Give the formal definition of the following NFA N (in tuple notation). Make sure you describe the transition function completely (for every state and every letter).



2. NFA to DFA.
   [**Category**: Construction., **Points**: 10]

   Convert the following NFA to an equivalent DFA (with no more than 10 states). You must construct this DFA using the subset-construction as done in class. Draw the DFA diagram, and also write down the DFA in tuple notation (there is no need to include states that are unreachable from the initial state).



3. NFA Construction by Guessing.
   [**Category**: Construction, **Points**: 10]

   Let $\Sigma = \{0, 1\}$ be an alphabet. Let

$$L_n = \left\{ \alpha c x c \beta \;\middle|\; c \in \Sigma, \alpha, \beta \in \Sigma^*, x \in \Sigma^n, \text{ and } |\alpha| \text{ is divisible by 3 and } |\beta| \text{ is divisible by 3} \right\}.$$

   (a) **[3 Points]** Draw the NFA for L_2.

   (b) **[7 Points]** Formally define the NFA $M_n = (Q_n, \Sigma, \delta_n, q_{init,n}, F_n)$ for L_n for any n. In particular, you must define formally $Q_n, \delta_n, q_{init,n}, F_n$ using simple set notations.

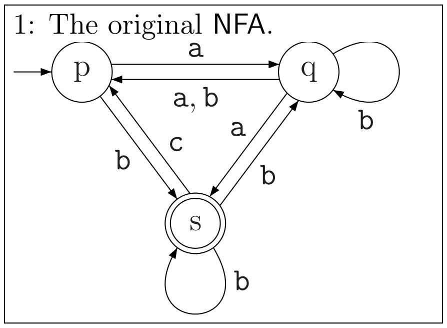
4. NFA to Regex.
   [**Category**: Construction, **Points**: 10]

   In lecture 8 (Sipser pp. 69–76), we saw a procedure for converting a DFA to a regular expression. This algorithm also works even if the input is an NFA.

   For the following NFA, use this procedure to compute an equivalent regular expression. So that everyone does the same thing (and we don't create a grading nightmare), you should do this by:

   – Adding new start and end states, and then
   – removing states p, q, s **in that order**.

   Provide detailed drawing of the GNFA after each step in this process.

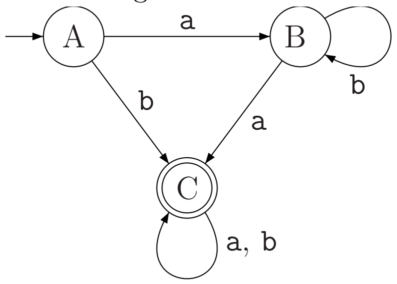


Note, that in this problem you will get interesting self-loops. For example, one can travel to from q to p and then back q. This creates a self-loop at q when p is removed.

5. NFA to Regex by other means.
[**Category**: Proof., **Points**: 10]

(**Extra credit.**)

There is another technique one can use to compute a regular expression from an NFA. As a concrete example, consider the following NFA M seen in class (lecture 8, the examples section).



Consider, for each state in the above automata, the language that this automata would accept if we set this state to be the initial state. The language accepted from state A, denoted by $L(A)$, which we will write as $\mathcal{A}$ to make the notations cleaner. Clearly, a word in $\mathcal{A}$ is either a followed by a word that is in $\mathcal{B}$ (the language the automaton accepts when B is the initial state), or its b followed by a word in $\mathcal{C}$ (the language the automaton accepts with C as initial state). We can write this observation as an equation over the three languages:

$$\mathcal{A} = \mathtt{a}\mathcal{B} \cup \mathtt{b}\mathcal{C}.$$

As a concrete example, in the above automata, C is a final state, which implies that $\epsilon \in \mathcal{C}$. As such, by the above equation, $\mathcal{A}$ must contain the word $\mathtt{b} = \mathtt{b}\epsilon \in \mathtt{b}\mathcal{C}$. Now, since A is the initial state of M, it follows that $L(M) = \mathcal{A}$. This implies that $\mathtt{b} \in L(M)$. (Thats a very indirect way to see that, but it would be useful for us shortly.)

(A) Write down the system of three equations for the languages in the above automata. (Note, that one gets one equation for each state of the NFA.)

(B) Let r, s be two regular expression over some alphabet Σ, and consider an equation of the form

$$\mathcal{D} = r\mathcal{D} \cup s.$$

let $\mathcal{D}$ be the minimal language that satisfies this equation. Give a regular expression for the language $\mathcal{D}$. Prove your answer.

(C) For a character $\mathtt{a} \in \Sigma$, what is the smallest language $\mathcal{E}$ satisfying the equation $\mathcal{E} = \mathtt{a}\mathcal{E}$? Prove your answer.

(D) The above suggests a way to get the regular expression for a language. Take the system of equations from (A), and eliminate from it (by substitution) the variable $\mathcal{B}$. Then, eliminate from it the variable $\mathcal{C}$. After further manipulation, you remain with an equation of the form

$$\mathcal{A} = \textit{something},$$

where "something" does not contain any of our variables (i.e., $\mathcal{B}$, and $\mathcal{C}$). Now, convert the right side into a regular expression describing all the words in $\mathcal{A}$.

Carry out this procedure in detail for the above NFA M (specifying all the details in your solution). What is the regular expression of the language of M that results from your solution?