Solutions for Problem Set 4
CS 373: Theory of Computation

Assigned: February 7, 2013 Due on: February 14, 2013

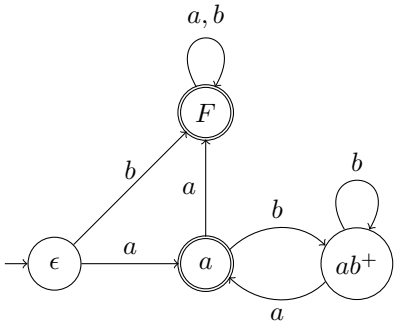
Problem 1. [Category: Comprehension+Design] Consider the language $L = \overline{\mathbf{L}((abb^*)^*)}$.

1. Construct a DFA recognizing L. You need not prove that your construction is correct. If you construct it using the algorithms described in class then you should show all your steps. If you construct the automaton directly then you should explain the intuition behind your construction clearly. **[5 points]**
2. Construct a regular expression for the language L. Again you don't need to prove your regular expression to be correct, but you should show all the steps in the construction. **[5 points]**

Solution:

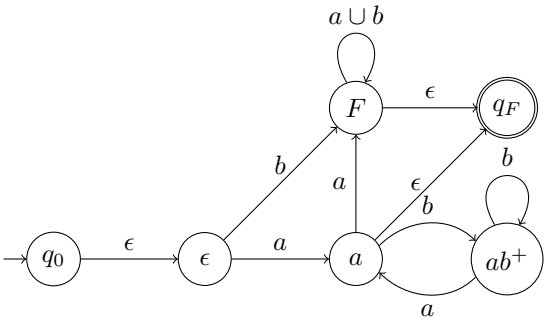
1. For this part, it is very useful to come up with the smallest DFA recognizing L, so that the solution in the next part is easy. One possibility is to start from the regular expression $(abb^*)^*$, construct an NFA for it, convert that NFA to a DFA, and then complement that and "minimize" it. However, we instead will find it convenient to construct a DFA for it directly.

   In order to construct the DFA for L, it is convenient to think about designing a DFA for $(abb^*)^*$ and then "complementing" it. To check if a string follows the pattern $(abb^*)^*$ we will have a state that remembers that we have seen the first a, and state that remembers that we have seen an a followed by at least one b. In addition, we will have an initial state, and a state we go to whenever we discover that the string does not have the pattern $(abb^*)^*$. We pick the final states based on the fact that we want to recognize L. Putting it all together we get the DFA shown below.

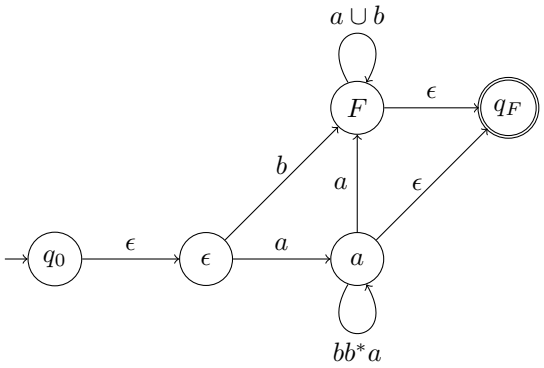


2. To construct a regular expression for the language L, we will use the algorithm described in class. We will convert the DFA above to a GNFA, and then remove one state at a time until we get a two state GNFA. The series of steps is shown below. In the pictures below, we do not draw the transitions labelled $\emptyset$ to avoid clutter.

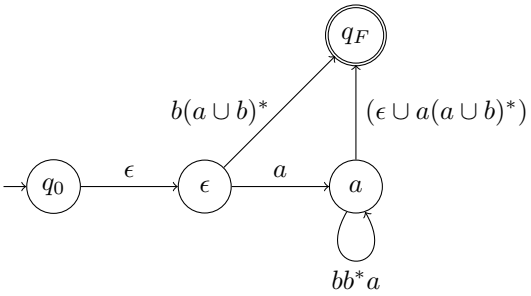
   The regular expression for L is $R = b(a \cup b)^* \cup a(bb^*a)^*(\epsilon \cup a(a \cup b)^*)$.



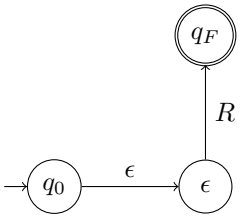
Step 1: Initial GNFA



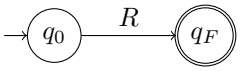
Step 2: GNFA after ripping ab^+



Step 3: GNFA after ripping F



Step 4: GNFA after ripping a



Step 5: GNFA after ripping ϵ

Figure 1: In step 4 and 5, $R = b(a \cup b)^* \cup a(bb^*a)^*(\epsilon \cup a(a \cup b)^*)$. R is the regular expression describing L.

■

Problem 2. [Category: Comprehension+Design+Proof] For any string $w = w_1 w_2 \cdots w_n \in \Sigma^*$ (with $w_i \in \Sigma$), the reverse of w, denoted as w^R, is the string w in reverse order, i.e., $w^R = w_n w_{n-1} \cdots w_1$. For a language $A \subseteq \Sigma^*$, let $A^R = \{w^R \mid w \in A\}$.

1. Given a DFA M, construct an NFA that recognizes $(\mathbf{L}(M))^R$. **[5 points]**
2. Prove that the NFA constructed in the previous part is correct. **[5 points]**

Solution:

1. We will construct an NFA M^R that will recognize $(\mathbf{L}(M))^R$. Essentially, the NFA M^R "reverses" the direction of the transitions of M and has a new initial state that has ϵ-transitions to the final states of M. Complete the formal definition of M^R based on this intuition.

   Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing A. The NFA $M^R = (Q^R, \Sigma, \delta^R, q_0^R, F^R)$ where

   - $Q^R = Q \cup \{q_0^R\}$ where $q_0^R \notin Q$,

- $F^R = \{q_0\}$
- And the transition function δ^R is described as

$$\delta^R(q,a) = \begin{cases} F & \text{if } q = q_0^R \text{ and } a = \epsilon \\ \{q' \mid \delta(q',a) = q\} & \text{if } q \in Q \text{ and } a \neq \epsilon \\ \emptyset & \text{in all other cases} \end{cases}$$

2. The correctness can be established by capturing the relationship between computations of M and computation of M^R. The statement to be proved by induction is

$$\forall w \in \Sigma^*.\ \forall q \in Q.\ q_0 \xrightarrow{w}_M q \text{ iff } q \xrightarrow{w^R}_{M^R} q_0$$

The proof of this statement by induction on $|w|$ is as follows.

- **Base Case:** Let w be such that $|w| = 0$. Then $w = \epsilon$. Now since M is deterministic, $q_0 \xrightarrow{\epsilon}_M q$ iff $q = q_0$ because a DFA does not take any steps without reading a symbol. Now, M^R has no ϵ-transitions except from the new initial state s, and q is a state of M (i.e., $q \in Q$). Thus, we have $q \xrightarrow{\epsilon}_{M^R} q_0$ (for $q \in Q$) iff $q = q_0$. Putting it all together we have the base case.
- **Induction Hypothesis:** For all w such that $|w| < n$, for all $q \in Q$, $q_0 \xrightarrow{w}_M q$ iff $q \xrightarrow{w^R}_{M^R} q_0$.
- **Induction Step:** Let $w = ua$, where $u \in \Sigma^{n-1}$ and $a \in \Sigma$. The induction step can be established by the following reasoning.

$$\begin{aligned} q_0 \xrightarrow{ua}_M q \quad & \text{iff } \exists q' \in Q.\ q_0 \xrightarrow{u}_M q' \xrightarrow{a}_M q \text{ where } \delta(q',a) = q \\ & \text{iff } \exists q' \in Q.\ q \xrightarrow{a}_{M^R} q' \xrightarrow{u^R}_{M^R} q_0 \text{ because of ind. hyp. and defn. of } \delta^R \\ & \text{iff } q \xrightarrow{w^R}_{M^R} q_0 \end{aligned}$$

We will now use the established claim to prove the correctness of the construction. Consider the following sequence of reasoning steps: $w \in \mathbf{L}(M)$ iff there is $q \in F$, s.t. $q_0 \xrightarrow{w}_M q$ (definition of M accepting) iff $q \xrightarrow{w^R}_{M^R} q_0$ (by statement just proved) iff $q_0^R \xrightarrow{\epsilon}_{M^R} q \xrightarrow{w^R}_{M^R} q_0$ iff $w^R \in \mathbf{L}(M^R)$. This completes the proof.

■

Problem 3. [Category: Comprehension+Design+Proof] For languages $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, define the *avoids* operation as follows.

$$A \text{ avoids } B = \{w \in A \mid w \text{ doesn't contain any string in } B \text{ as substring}\}$$

1. Let $A = \{1001, 1111\}$ and $B = \{01, 10\}$. What is A avoids B? **[1 points]**
2. Let $A = \mathbf{L}((0 \cup 1)^*)$ and $B = \mathbf{L}(1^*)$. What is A avoids B? **[1 points]**
3. Prove that if A and B are regular then $A\text{avoids}B$ is regular. You can either construct a DFA/NFA/regular expression for A avoids B (and then you don't have to prove that your construction is correct) or use previously established closure properties to prove this result. **[8 points]**

Solution:

1. A avoids $B = \{1111\}$

2. A avoids $B = \emptyset$

3. Observe that $L_1 = \Sigma^* B \Sigma^*$ is the set of all strings that have a substring in B. Moreover, L_1 is regular because Σ^* is regular, B is regular, and regular languages are closed under concatenation. We can see that

$$A \text{ avoids } B = A \setminus L_1$$

Since A is regular and regular languages are closed under set difference, we can conclude that A avoids B is regular.

■