6.045J/18.400J: Automata, Computability and Complexity Prof. Nancy Lynch

Quiz 1: Solutions

Nati Srebro, Susan Hohenberger

Please write your name in the upper corner of each page. (2 Points)

Problem 1:

True or False (18 points) Full credit will be given for correct answers. If you include justification for your answers, you may obtain partial credit for incorrect answers.

In all parts of this question, the alphabet Σ is $\{0, 1\}$.

1. True or False: If L is a regular language and F is a finite language (i.e., a language with a finite number of words), then $L \cup F$ must be a regular language.

True; all finite languages are regular languages and regular languages are closed under union.

2. True or False: If L is a regular language, then $\{ww^R : w \in L\}$ must be a regular language. (Here, w^R denotes the reverse of string w.)

False; we can show this language in not regular using techniques similar to Example 1.40 on page 81 of Sipser. A common mistake is confusing the language above with LL^R.

3. True or False: Regular expressions that do not contain the star operator can represent only finite languages.

True; the star operator in regular expressions is the equivalent of a loop in DFAs. If a deterministic finite automata with n states does not contain a loop, then *at most* it can recognize strings of length less than n. The set of such strings is finite.

4. True or False: Define $EVEN(w)$, for a finite string w, to be the string consisting of the symbols of w in even-numbered positions. For example, $EVEN(1011010) = 011$.
   If L is a regular language, then $\{EVEN(w) : w \in L\}$ must be regular.

True; given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes L, we can build an NFA $M' = (Q, \Sigma, \delta', q_0, F)$ that recognizes $L' = \{EVEN(w) : w \in L\}$. Intuitively, the transition function δ' is set so that from every state A there are outgoing transitions to any state that is two hops (i.e., reachable in M by reading two input characters) away from A, where the new transition symbol in M' is the second of the two symbols in M.

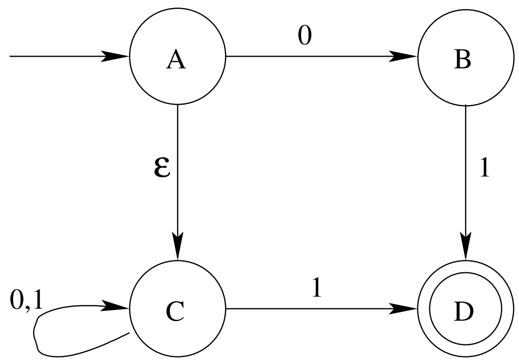
5. True or False: For every pair of regular expressions R and S, the languages denoted by $R(SR)^*$ and $(RS)^*R$ are the same.

True; intuitively, one can see that minimally both expressions are R and that when the star operator is exercised they both expand to expressions of alternating Rs and Ss, where both expressions begin and end with R. It is possible to prove that they describe equivalent sets using induction.

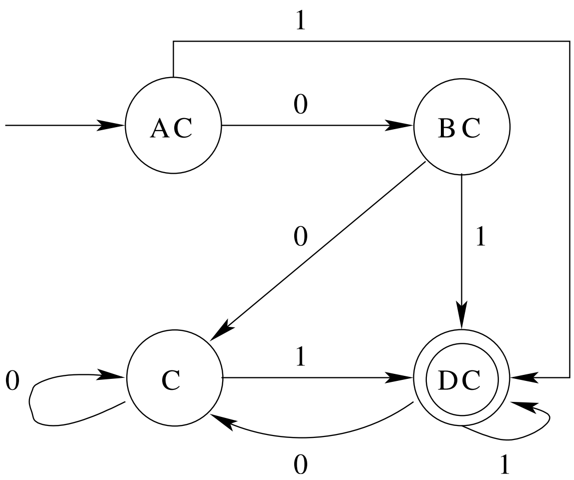
6. True or False: If L_1 and L_2 are languages such that L_2, L_1L_2, and L_2L_1 are all regular, then L_1 must be regular.

False; consider $L_1 = \{0^{2^i} : i > 0\}$ and $L_2 = \{0\}^*$.

Problem 2: **(20 points)** Consider the following NFA:



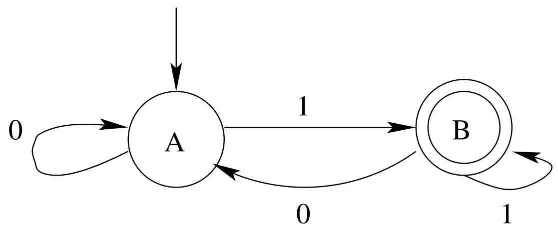
1. **(16 points)** Convert this NFA into an equivalent DFA using the procedure we studied in class. **Your answer should be the state diagram of a DFA. Your diagram should include only the states that are reachable from the start state.** (Note: There are not more than a half-dozen states in the resulting DFA). Please label your states in some meaningful way. You may explain your work, to receive more credit for an incorrect answer.



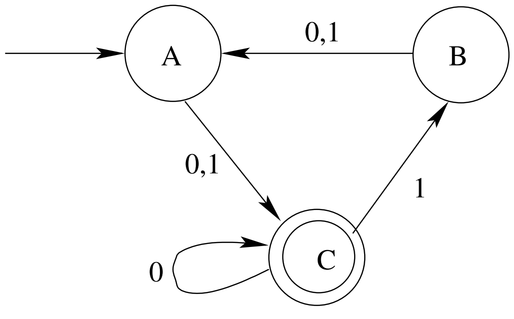
2. **(3 points)** What language is recognized by your DFA? Your answer may be either a regular expression or an explicit description of the set.

"any string in $\{0,1\}^*$ ending in a 1": $(0 \cup 1)^*1$.

3. **(3 points)** Give a DFA with two states that recognizes the same language.

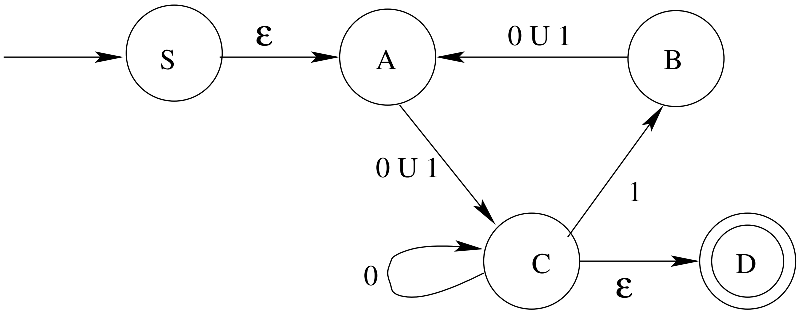


Problem 3: **(20 points)** Find a regular expression for the language recognized by this machine, using the procedure we have studied in class: Show all your work, in particular, the state diagrams after the removal

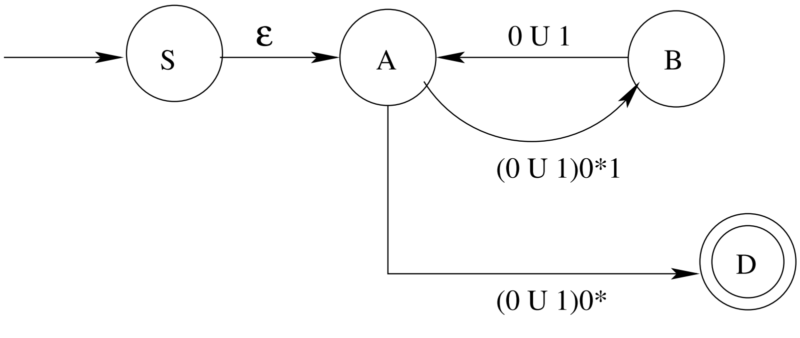


of each successive state. You may omit $\emptyset$-transitions from your diagrams. Please start by giving the GNFA before state removal and then remove states in the order C, B, A.

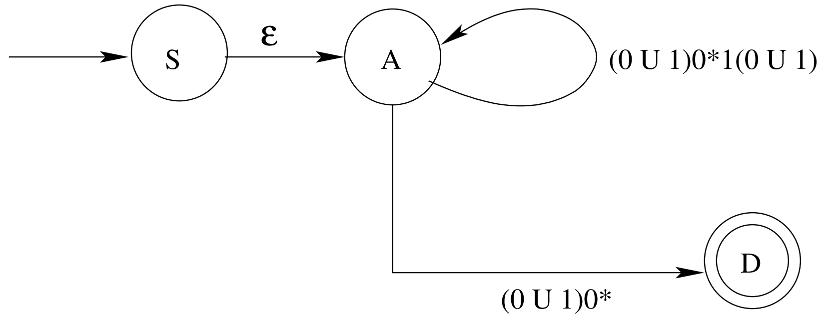
GNFA before state removal:



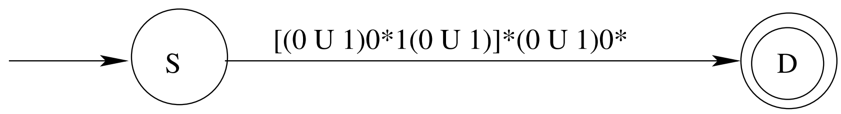
After removing state C:



After removing state B:



After removing state A:



Regular expression representing the language recognized by the original DFA:
$[(0 \cup 1)0^*1(0 \cup 1)]^*(0 \cup 1)0^*$

Problem 4: **(20 points)** Regular expressions are defined using three operators: union, concatenation, and star. Suppose we define "Extended Regular Expressions" in the same way as regular expressions, with the addition of the *set intersection* operator. For example, $((0 \cup 1)^*1) \cap (1(0 \cup 1)^*)$ is an extended regular expression, which denotes the set of words that both begin and end with 1.

Show that adding the intersection operator does not extend the power of ordinary regular expressions. Do this by describing a *procedure* that, given an extended regular expression α, produces an ordinary regular expression β that represents the same language. You may use procedures described in class and in Sipser's book without saying how they work, e.g., you may say things like "convert the NFA to a DFA". The description of your procedure should be concise, but the procedure need not be the most efficient one possible.

Input: Extended Regular Expression α

Procedure:

Given α, we work recursively to build regular expressions for all its subexpressions, until we get β, a regular expression equivalent to α. The interesting recursive step is the one where we use the intersection operator (i.e., add this recursive step to the recursive definition of a regular expression that we covered in class).

Then, the problem reduces to:
Given two ordinary regular expressions S_1 and S_2, describe a procedure for producing a regular expression for $L(S_1) \cap L(S_2)$.

Now, doing this involves several steps:
Convert S_1 and S_2 to NFAs, then to DFAs, using the procedures given in class. Then construct a DFA for the intersection $L(S_1) \cap L(S_2)$, using the product machine construction. Then convert the DFA to a GNFA and then to a regular expression.

Output: Regular Expression β, which is equivalent to α.

Problem 5: **(20 points)** Prove that the following language L over the alphabet $\{a, b, c\}$ is not regular:

$$L = \{wcx : w, x \in \{a, b\}^* \text{ and the number of } a's \text{ in } w \text{ is equal to the number of } b's \text{ in } x\}.$$

For example, the word $abababcbbb$ is in L.

Claim: L is not regular.
Proof:
Assume to the contrary that L is regular. Let p be the pumping length given by the pumping lemma. Let s be the string $a^p cb^p$. Because s is a member of L and s has length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, satisfying the three conditions of the lemma. We show that this is impossible.

Since, by the pumping lemma, we know that $|xy| \leq p$ and $|y| > 0$, then for this s, we see that y is made up of one or more a's that all come *before* the 'c' symbol. If we pump down, by letting $i = 0$, then the resulting string is $s' = xy^0 z$, where the number of a's before 'c' is at least one less than the number of b's after it. This new string s' is not in L. Thus, since s can not be pumped, L is not a regular language.