Introduction to Theory of Computing	**Problem Set 4**
CS 481 Fall 2004	**Due Friday, October 1, 2004**

Reading. To review the material covered by this problem set, read sections 7-10 of Kozen.

For each problem set please write both your name and your cornell.edu email address on top. **Please turn in Problems 1-2 and Problems 3-4 separately with your name written on both.**

(1) Give regular expressions for each of the following sets. For parts (a)-(c) assume that $\Sigma = \{a, b\}$.

(a) The number of a's is divisible by 3.

(b) The number of a's is divisible by 3 or the length is odd.

(c) $\Sigma^* - (a+b)^*ab(a+b)^*$

(d) For this problem let $\Sigma = \{a, b, c\}$. Now give a regular expressions for the set $\Sigma^* - (a+b)^*ab(a+b)^*$

(2) Are the following identities true for all regular expressions α and β? Prove your answer.

(a) $(\alpha + \beta)^* = \alpha^* + \beta^*$

(b) $(\alpha + \beta)^*\beta = (\alpha\beta^*)^+$

(c) $\alpha(\alpha\alpha + \alpha)^*\beta = (\alpha^+\beta)^+$

(d) $\alpha(\beta\alpha)^*\beta = (\alpha\beta)^+$

(3) Prove that if A is a regular language over an alphabet Σ that contains letters $a, b \in \Sigma$ $L(A) = \{c^n | \exists w \in A$ such that #a(w) + #b(w) = n$\}$ is also regular.

(4) We can think of a NFA as a way to *generate* words in a regular language L, rather than a machine that decides if given words are in L. To generate words in L you start at any start state, and when in a state $q \in Q$ you may continue on any arrow leaving q, and following an arrow for a letter σ, you append σ to your current word. In accepting states $q \in F$ you may terminate the word, in all other states you must continue. First note the following fact:

Fact. *For a given NFA M, the set of words that is possible to generate by the above method is exactly the language $L = L(M)$.*

You do not need to prove this fact, and you may use it in your solution. Now consider an extension of this idea, where arrows can have any regular expressions, not just letters (such as the ε symbol, or any other expression). See the Figure for two examples of such an *NFA with regular-expressions*. This NFA generates words the same way as a regular NFA, when following an arrow you append one word from the language corresponding to the regular expression. For example, the NFA on the left of the Figure defines the same language as the regular expression $0^*(01)^*$. The

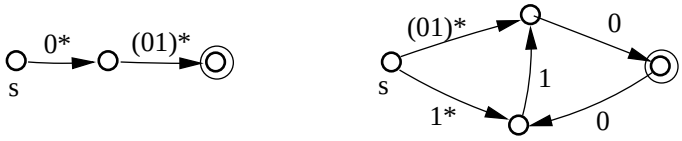

Figure 1: Two examples of NFA with regular expressions.

language generated by the NFA with regular expressions on the right, contains the words 11110 (by starting on the lower arrow out of s and using 111 as a word in $L(1^*)$), or the word 11110010 (by starting the same, and going around the circle twice), or the word 0101010 (by starting on the upper arrow).

(a) Give a regular expression for the language generated by the NFA on the right of the Figure.

(b) Given an NFA M with regular expressions let $L(M)$ be the set of words that can be generated by this method. Assume that M has a single start and a single accepting state. Show that $L(M)$ is regular.

Hints: I see two different options for solving this problem. (1) try to generate a regular expression α so that $L = L(\alpha)$. You can do this by the method of lecture 9, or by induction on the number of states: Given such an NFA consider a state q, that is not the start or the accepting state, and "reduce" the NFA by defining an equivalent NFA with fewer states.

(2) For an alternate option you can prove that there is a traditional NFA N that accepts the same language as generated by M with the regular expressions. You may use ε transitions in your regular NFA.