

**CS321**  
**Theory of Computation**  
**Quiz 4, Fall 2008**

Name:

1. [5pt] Prove that all finite languages are regular.

Consider a finite language  $L = \{w_1, \dots, w_n\}$ , where the  $w_i$  are finite strings over some alphabet. Now consider the regular expression  $r = w_1 + w_2 + \dots + w_n$ . It is clear from the semantics of regular expressions that  $L = L(r)$ . Thus, since regular expressions define regular languages we have shown that  $L$  is regular.

2. [15pt] Let SkipFirst be an operation on languages defined as follows:

$$\text{SkipFirst}(L) = \{w : aw \in L, a \in \Sigma\}$$

That is,  $\text{SkipFirst}(L)$  is the set of strings that are formed by removing the first character from each string in  $L$ .

Show that the set of regular languages is closed under SkipFirst.

**Solution:** Consider a regular language  $L$  and let  $M$  be a DFA such that  $L = L(M)$ . We will now create an NFA  $N$  such that  $L(N) = \text{SkipFirst}(L)$ , which will show that  $\text{SkipFirst}(L)$  is regular and thus that regular languages are closed under the SkipFirst operator.

The state space of  $N$  will be identical to  $M$  except that we will add a new start state  $q'$  that has a  $\lambda$ -transition from  $q'$  to any state  $q$  that can be reached by a single transition from the initial state of  $M$ . Now consider any string  $w \in L(M)$ , where  $w = av$  for some character  $a$  and string  $v$ . We now argue that  $v$  will be accepted by  $N$ . To see this not that the original machine  $M$  must read in the leading  $a$  via a transition from the initial state of  $M$  to some state  $q$  and then from  $q$  read in the rest of the string  $v$  and end up at a final state. We can see that  $N$  will have a  $\lambda$ -transition from its initial state  $q'$  to  $q$  which means that it can then read in  $v$  from  $q$  and end up at a final state. Thus,  $N$  will accept  $v$ . Since the above argument holds for any string in  $L$ , this shows that  $L(N) = \text{SkipFirst}(L)$ .

3. [15pt] Use the Pumping Lemma to directly prove that the language

$$L = \{w_1cw_2 : w_1 \in \{a, b\}^*, w_2 \in \{a, b\}^*, |w_1| \leq |w_2|\}$$

is not regular.

Assume that  $L$  is regular. This means that the pumping lemma applies.

Let  $m$  be the pumping constant and consider the string  $w = a^mca^m$  which is in  $L$ . The pumping lemma tells us that  $w$  decomposes into  $xyz$  such that  $|xy| \leq m$  and  $|y| \geq 1$  and that  $w_i = xy^iz$  must be in  $L$ . Here we see that  $y = a^j$  for some  $j \geq 1$  which implies that  $w_i = a^{m+(i-1)j}ca^m$ . For  $i = 2$  we see that this gives  $w_2 = a^{m+j}ca^m$  which is not in the language. Since the pumping lemma tells us  $w_2$  is in  $L$  we have derived a contradiction showing that our original assumption is false. Thus,  $L$  is not regular.

4. [15pt] Use closure properties and the fact that  $\{a^ib^i : i \geq 0\}$  is not regular to prove that

$$L = \{w_1w_2 : w_1 \in \{a, b\}^*, w_2 \in \{a, b\}^*, n_a(w_1) = n_b(w_2)\}$$

is not regular. You may use any closure property discussed in class or in the book.

This problem does not have a solution. This was not intended to be a trick question, but rather an honest mistake by your professor. It turns out that  $L$  is equal to  $\{a, b\}^*$ , i.e. it contains all strings. This is not immediately obvious and was pointed out by one of your classmates. That is, it turns out that any string can be broken into two strings  $w_1w_2$  such that the number of a's in  $w_1$  is equal to the number of b's in  $w_2$ . This is an interesting property that can be proved via induction on the length of a string.

Needless to say I won't count this problem on the score of the quiz.