

**CS321**  
**Theory of Computation**  
**Solution to Exam I, Fall 2009**

**Important:** Please show work and justify your answers. Otherwise you will not get full credit. Describe the idea informally first. Time: 50 minutes.

1. (a) [5pt] Convert the following regular expression to a right linear grammar:  
 $((aa + b)^*(bb + abab)(a + b)^*)^*$

The right linear grammar may be given as follows:

$$S \rightarrow aaS | bS | A | \lambda$$

$$A \rightarrow bbB | ababB$$

$$B \rightarrow aB | bB | S$$

The first rule generates any number  $aa$ 's or  $bs$ , and starts the second rule. The second rule generates exactly one  $bb$  or  $abab$  and starts the third rule. The third rule generates any number of  $a$ 's and  $bs$  and gives control back to the first rule. The first rule can also stop with generating  $\lambda$ .

- (b) [5pt] Suppose  $w, u, v$  represent any strings and  $a$  represents any single symbol. We can formally define the extended transition function for DFAs as

$$\delta^*(q, a) = \delta(q, a)$$

$$\delta^*(q, wa) = \delta(\delta^*(q, w), a). \text{ Use induction on the length of string } w \geq 1 \text{ to show that}$$

$$\delta^*(q, uv) = \delta^*(\delta^*(q, u), v).$$

This was apparently the most difficult, but it is also the easiest if you understand induction.

It helps to understand what the statement is saying. It is saying that if there is a walk labeled  $uv$  from  $q$  to some state  $r$ , then it can be split as a walk labeled  $u$  from state  $q$  to some state  $p = \delta^*(q, u)$ , from which there is another walk labeled  $v$  to  $r$ . It should be pretty obvious. The point is to show this more rigorously, keeping it straight what is known and what is not.

**Basis:**  $|v| = 1$ . So let  $v = a$ . We are told  $\delta^*(q, wa) = \delta(\delta^*(q, w), a) = \delta^*(\delta^*(q, w), a)$ , which proves the base case. (The last step follows from  $\delta^*(q, a) = \delta(q, a)$  since  $a$  is a single symbol.)

**Inductive hypothesis:** Assume  $\delta^*(q, uv) = \delta^*(\delta^*(q, u), v)$ , where the length of string  $v$  is  $n$ .

**Inductive Step:** We need to show the result holds for a string  $w = va$  which is of length  $n + 1$ . So we *need to* show that  $\delta^*(q, uva) = \delta^*(\delta^*(q, u), va)$ . Start from the left hand and rewrite it using what we know and arrive at the right hand side.

$$\delta^*(q, uva) = \delta(\delta^*(q, uv), a)$$

We just applied the definition of  $\delta^*$  here treating  $uv$  as the string  $w$  and  $a$  as a single character in the definition.

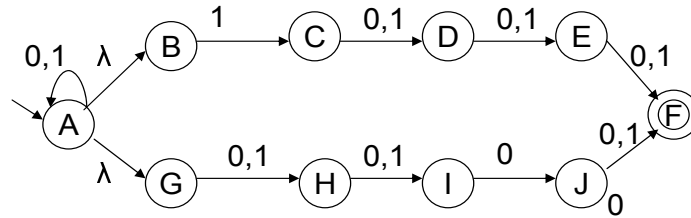
$$= \delta(\delta^*(\delta^*(q, u), v), a)$$

This was by applying inductive hypothesis to  $uv$ .

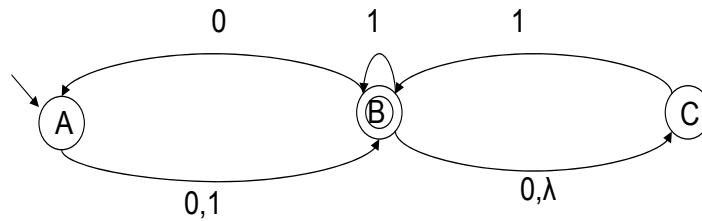
$$= \delta^*(\delta^*(q, u), va)$$

We again applied the definition of  $\delta^*$ , this time treating  $\delta^*(q, u)$  as the state  $q$  in our definition. You can verify that we showed what we needed to show.

2. (a) [4pt] Give an NFA for the language  $L$  over  $\{0, 1\}$ , where each string is at least 4 bits; more over the 4'th bit of string from the right end is a 1 or the 2'nd bit from the right end is a 0. So for example, 001010 and 0100 are in the language and 0111 and 00 are not.

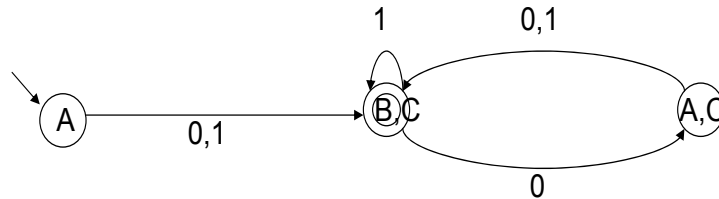


- (b) [2pt] Which of the strings 00, 01001, 10010, 000, 0000 are accepted by the following NFA?



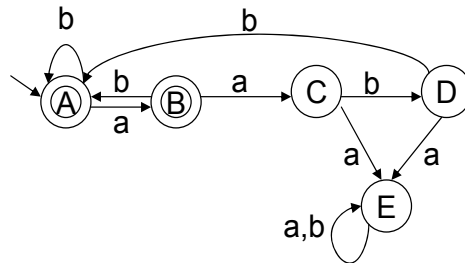
000 and 01001 are the only strings accepted.

- (c) [4pt] Convert the above NFA to a DFA.



3. Given the alphabet  $\Sigma = \{a, b\}$ , consider the set of strings wherein every occurrence of  $aa$  is followed by a  $bb$ . Note that the strings  $\lambda, b, ab, a, aabbb$  are in the language and  $aaab, aab, aabbaab, aa$  are not.

- (a) [5pt] Give a DFA for the above language. Make sure that your DFA works on all the above examples.



- (b) [4pt] Give a regular expression for the same language. You might convert the above DFA to a regular expression systematically, or give a regular expression directly.

There are many possible answers. The simplest one is  $(b + ab + aabb)^*(a + \lambda)$ . The  $(a + \lambda)$  part is needed to allow the strings to end with  $a$ , since that is permissible according to the rules. It cannot end with  $aa$ , however.

- (c) [1pt] Give a general method to convert a DFA that accepts any language  $L$  to one that accepts its complement  $\bar{L}$ . You might use the above DFA as an example, but must describe the idea in general terms.

All you need to do is to convert the final states into non-final states and vice versa.