

Ph.D. Qualifying Exam (Fall 2006)  
Automata and Formal Languages  
Answer ALL questions  
Closed Book Examination

Question 1. (10% + 10% + 10% + 20% + 20%)

(a) Give an inductive definition for regular expressions over the alphabet  $\{a, b\}$ .

(b) Let  $r = a^*(ba^*)^*$ . Apply the algorithm given in class or in a textbook to convert  $r$  into an equivalent NFA, then into an equivalent DFA. By inspection of the DFA constructed, conclude that  $r$  denotes the set of all finite strings over  $\{a, b\}$ . Thus,  $(a + b)^*$  and  $a^*(ba^*)^*$  equivalent in denoting the same language.

We say that a regular expression is in  $(\cdot, *)$ -form if the expression does not contain any occurrence of  $+$  (union) symbol. Equivalently, we say that it is a  $(\cdot, *)$ -expression.

A regular expression is said to be in **disjunctive normal form** (DNF-form) if it is of the form  $(\alpha_1 + \alpha_2 + \dots + \alpha_n)$  where  $\alpha_i$  is in  $(\cdot, *)$ -form for  $1 \leq i \leq n$ . That is, we say that a regular expression is in DNF-form if it is a union of  $(\cdot, *)$ -expressions.

(c) Let  $r = (r_1 + r_2)^*$  where  $r_1$  and  $r_2$  are in  $(\cdot, *)$ -form. Give a regular expression in  $(\cdot, *)$ -form that is equivalent to  $r$ .

Hint: the resulting  $(\cdot, *)$ -expression may have nested  $*$  operators.

Suppose we want to write a program to manipulate regular expressions in DNF-form. Given a regular expression  $r$  where  $r = (r_1 + r_2 + \dots + r_n)$  and  $r_1, r_2, \dots, r_n$  are  $(\cdot, *)$ -expressions, we represent  $r$  in the computer program as a list of  $(\cdot, *)$ -expressions; that is,  $r = [r_1, r_2, \dots, r_n]$ .

(d) Given a regular expression  $r$  in DNF-form, where  $r$  is represented as  $[r_1, r_2, \dots, r_k]$  and  $r_1, r_2, \dots, r_k$  are in  $(\cdot, *)$ -form, provide a **recursive** algorithm for computing  $(r)^*$ . Your algorithm should return  $(r)^*$  in DNF-form represented as a list of  $(\cdot, *)$ -expressions.

(e) Show by a careful mathematical induction proof, *inducing on the structure of a regular expression according to the definition given in part (a)*, that every regular expression can be equivalently represented by a regular expression in DNF-form.

Question 2. (30%)

A context-free grammar (CFG) is called **linear** if the right hand sides of its productions contain **at most** one nonterminal.

An example of a linear context-free grammar is:  $S \rightarrow a \mid aAb, A \rightarrow b \mid bSa$

In general, the stack height of a pushdown automaton (PDA) can increase and decrease by push and pop operations. We say that a PDA is making a turn if the stack height is changed from increasing to decreasing, or the stack height is changed from decreasing to increasing. It is possible that a PDA may make many turns in processing an input.

A pushdown automaton PDA is said to be **single-turn** if the stack height cannot make a change from decreasing to increasing. That is, the stack height can only make one single change from increasing to decreasing.

Explain how one can convert a single-turn PDA to an equivalent linear CFG.

Remarks: In fact, we can also convert any linear CFG to an equivalent single-turn PDA. But you are not asked to give this conversion.

**Hints:**

You may assume that the PDA has been normalized (standardized) as follows:

- A symbol  $Z_0$  is initially placed in the bottom of the stack. The PDA does not have to push explicitly such an initial stack symbol as its first operation.
- The PDA accepts by both reaching a unique final state  $q_f$  and emptying the stack.
- The PDA pushes or pops at most one stack symbol in a transition.

Let the set of states of the single-turn PDA be  $Q$  and the stack alphabet set be  $\Gamma$ . We construct a linear CFG by defining the set of nondeterminals to be  $Q \times \Gamma \times Q$ .

Suppose the single-turn PDA is in state  $q$  with the top stack symbol  $A$ . Let  $L_{q,A,q'}$  be the set of input strings that causes the single-turn PDA to pop the top symbol  $A$  and enters state  $q'$ . Note that the PDA may have pushed and popped some stack symbols before removing the stack symbol  $A$ .

Let  $(q, A, q')$  be a nonterminal of the grammar. The purpose of  $(q, A, q')$  is to generate exactly the set of strings in  $L_{q,A,q'}$ .

Your job is to specify the starting symbol and the grammar rules.