

Name \_\_\_\_\_

**Exam Two**

**CS139 - Theory of Computing  
Drake University - Spring, 2004**

*Directions: Do all problems. They have the same weight. Show all work. Please work first on problems with which you are more comfortable.*

**Problem 1.** Produce a nondeterministic finite automaton (NFA) whose language is the same as the language described by the following regular expression.

(a)

$(0|1)0^*10^*(0|1)^*$

(b)

$(01(0(0|1)^*1)^*)^*$

**Problem 2.** For the regular expression

$$(a|b|c)(a|b|c)^*(b|c)(b|c)(a(bc^*)^*)^*$$

which of the following strings are *IN* the language of this expression and which are *OUT* (not IN)?

(a)

$\epsilon$

(b)

*ccbbbecc*

(c)

*abcabc*

(d)

*cbaabc*

(e)

*abbacccabba*

**Problem 3.** Use the state-elimination method to produce a regular expression whose language is the same as the language for the following nondeterministic finite automaton.

**Problem 4.** Use the Pumping Lemma (explaining your reasoning carefully) to demonstrate that the following language is *not* regular:

$$\{ 0^m 1^{m+1} 0^m \mid m \text{ is a nonnegative integer} \}$$

**Problem 5.** The set of terminal symbols in the following context-free grammar is  $\{ ( , ) , [ , ] \}$ . The set of variable symbols is  $\{ S, T, U \}$ , with  $S$  as the starting symbol. The substitution rules are as follows:

$$\begin{aligned} S &\rightarrow T \mid U \\ T &\rightarrow \epsilon \mid TT \mid (U) \\ U &\rightarrow \epsilon \mid UU \mid [T] \end{aligned}$$

(a) (6 pts) Show a *left*-derivation for the string  $[(())][([[]])]$ .

(b) (6 pts) Show the parse tree that corresponds to the derivation in part (a).

(c) (3 pts) Is this context-free grammar ambiguous or unambiguous? (No evidence required.)

**Problem 6.** Find a context-free grammar whose language is

$$\{ w \in \{0, 1\}^* \mid \text{either } w \text{ is a palindrome or } w \text{ has even length} \}$$

**Problem 7.**

(a) As clearly as you are able, describe the sort of “machine” modeled by a push-down automaton (PDA). Explain how it works.

(b) To prove that there exists a PDA equivalent to any given CFG  $G$ , we “built” one whose input alphabet was the set of terminal symbols of  $G$ , and whose stack alphabet was the set of all (variable and terminal) symbols of  $G$ . Briefly describe the behavior of this PDA. Why was *nondeterminism* an essential feature of the PDA we built?