CS103 Handout 22
Spring 2012 June 4, 2012

Additional Practice Final #1

This additional practice final is an actual CS103 final exam from a previous quarter. The structure of this exam is not the same as the structure of the upcoming final exam, but the sorts of questions on it are similar to what you might expect to see on the actual exam.

This practice final is **not worth any extra credit points**, but should be a good resource to help you prepare for the exam.

Enjoy!

FINITE AUTOMATA

1. (5 points) Let L be the following language: $L = \{ awa \mid w \in \{a, b\}^* \}$. Show by drawing an NFA that L* is regular. Your NFA should have no more than three states.

2. (5 points) Draw the transition diagram for a two-state NFA that accepts (ab | a)*.

3. (5 points) In our definition of a DFA as $M = (Q, \Sigma, \delta, q_0, F)$, δ is the transition function. We could define an extended transition function as follows:

$\delta^* : Q \times \Sigma^* \rightarrow Q$ where $\delta^*(q, w)$ is the state that M is in after starting in state q and processing all the symbols in the string w.

Use δ^* to complete the following definitions of the strings that are accepted and not accepted by M:

$L(M) = \{ w \mid \qquad\qquad\qquad\qquad \}$

$\overline{L(M)} = \{ w \mid \qquad\qquad\qquad\qquad \}$

REGULAR EXPRESSIONS and REGULAR LANGUAGES

4. (12 points) There are three parts to this question. Answer each with T or F.

(a) If w is any string, then w^R denotes the reverse of w. If L is any language, then L^R denotes the language consisting of the reverses of all strings in L.

True or false: If L is any language, then the language $LL^R = \{ww^R \mid w \text{ belongs to } L\}$.

[There was some confusion on this question. What we are asking is whether the formula in brackets correctly describes the language LL^R. Note that if A and B are languages, $AB = \{xy \mid x \in A \text{ and } y \in B\}$.]

Answer: ________

(b) True or false: Every NFA can be converted to an equivalent one that has a single accepting state.

Answer: ________

(c) True or false: For every three regular expressions, R, S, and T, the languages denoted by (R | S)*T* and R*T* | S*T* are the same.

Answer: ________

5. (15 points) Write a regular expression for the set of strings that consist of alternating 0's and 1's. We consider the empty string and the string 0 and the string 1 to be members of this language. For example, the following are some of the strings in the language: ε, 0, 1, 01010101, 10, 101.

CONTEXT-FREE LANGUAGES

6. (13 points) Show that the following language is ambiguous, then construct an unambiguous grammar that yields the same language.

$S \rightarrow AB \mid aaB$
$A \rightarrow a \mid Aa$
$B \rightarrow b$

7. (15 points) Use the Pumping Lemma to show that $L = \{a^n b a^n \text{ for } n \geq 0\}$ is not regular. We have laid out the proof for you, so all you have to do is fill in the blank spots.

The proof is by contradiction. Assume that L is _________________. Let p be the pumping length for L as given by the pumping lemma.

Consider the string w = ________________.

Clearly, w belongs to L, because ____.

By the Pumping Lemma, we can write $w = xyz$ where $|xy| \leq p$, $|y| > 0$, and
(the rest of the proof goes here)

Thus, we have a contradiction, and L is not regular.

TURING MACHINES/UNDECIDABILITY

8. (15 points) Show that the language

$H_{ALL} = \{ \langle M \rangle \mid M \text{ is a Turing machine that halts on } \Sigma^* \}$

is not decidable, by reduction from $HALT_{TM}$.

NP-COMPLETENESS

9. (15 points) Assume that we have a polynomial-time algorithm for determining whether a formula is satisfiable, (By the Cook-Levin theorem on page 272, we are effectively assuming that P = NP). So given a formula, our algorithm will output in polynomial time "yes" if the given formula is satisfiable, and "no" otherwise. However, the algorithm does not generate a specific assignment for each variable in the formula that satisfies it (it only tells us whether the formula can be satisfied). Explain how you could use our algorithm to generate a satisfying assignment for a given formula in polynomial time. *(Hint: Can you think of a way to find the assignment bit-by-bit?)*