Software Engineering 4I03 | Michael Soltys

DAY CLASS
DURATION OF EXAMINATION: 3 Hours
MCMASTER UNIVERSITY FINAL EXAMINATION | December 5, 2003

THIS EXAMINATION PAPER INCLUDES 2 PAGES AND 10 QUESTIONS. YOU ARE RESPONSIBLE FOR ENSURING THAT YOUR COPY OF THE PAPER IS COMPLETE. BRING ANY DISCREPANCY TO THE ATTENTION OF YOUR INVIGILATOR.

No aids allowed.

All questions worth 10 marks, for a total of 100.

1. The function **parity** : $\{0,1\}^* \longrightarrow \{0,1\}$ is 1 if the string has an odd number of 1s, and 0 if the string has an even number of 1s. For example, parity$(001011) = 1$, while parity$(11) = 0$ and parity$(0000) = 0$. Formally, parity$(w) = (\sum_{i=1}^{|w|} w_i) \pmod 2$.

   Let $L_{\text{par}} = \{w | \text{parity}(w) = 1\}$, and show that L_{par} is a regular language.

2. How would you restrict Turing machines so that they accept precisely the set of regular languages?

   (**Hint:** Turing machines are a model of computation with certain features. You can limit these features—for example you can restrict that the head move only in one direction. What limitations do you need to impose so that the limited model recognizes all, and only, regular languages?)

3. The function **majority** : $\{0,1\}^* \longrightarrow \{0,1\}$ is a function on binary strings that is 1 if the string has more 1s than 0s, and 0 otherwise. For example, majority$(01011) = 1$, but majority$(00011) = $ majority$(1100) = 0$.

   Let $L_{\text{maj}} = \{w | \text{majority}(w) = 1\}$. Show that L_{maj} is *not* a regular language.

4. Consider the context-free grammar given by the rules:

$$S \longrightarrow ASB|\varepsilon$$
$$A \longrightarrow aAS|a$$
$$B \longrightarrow SbS|A|bb$$

   (a) Are there any useless symbols? Eliminate them if so.
   (b) Eliminate ε-productions.
   (c) Eliminate unit productions.
   (d) Put the grammar into Chomsky normal form.

5. Are context-free languages recursive? Justify your answer.

Continued on Page 2

6. A **rewriting system** R is a generalized context-free grammar. The rules allow more than one symbol on the left; that is, we can have rules of the form $\alpha \to \beta$ where $\alpha, \beta \in (T \cup V)^*$. Then, from $\alpha_1 \alpha \alpha_2$ we can derive the string $\alpha_1 \beta \alpha_2$. Formally, $\alpha_1 \alpha \alpha_2 \Rightarrow \alpha_1 \beta \alpha_2$ by the rule $\alpha \to \beta$.

   There is a start variable S, and we say that a string $w \in T^*$ is in the language of a particular rewriting system R if there is a derivation of w from S using the rules of R, i.e., if $L(R) = \{w \in T^* | S \overset{*}{\Rightarrow} w\}$.

   Show that the set of languages defined by rewriting systems is precisely the set of recursively enumerable languages.

   (**Hint:** First show that if a language has a rewriting system, then it is RE. For the other direction, if L is RE then it has a Turing machine such that $L = L(M)$. Show that there exists a rewriting system that can simulate the computation of M on any w—similar idea to the Post's Correspondence Problem.)

7. Let $L_{\text{sub}} = \{\langle R, S\rangle | R$ and S are regular expressions s.t. $L(R) \subseteq L(S)\}$. Here $\langle R, S\rangle$ denotes an encoding of two regular expressions R and S. Show that L_{sub} is a recursive language.

8. Let $L_{\text{all}} = \{\langle M\rangle |\ M$ is a TM which on input 00 visits all of its non-halting states$\}$. That is, L_{all} consists of encodings of Turing machines which on input 00 at some point in the computation visit every non-halting state. The computation may never end, or it may end in a halting state.

   Show that L_{all} is undecidable.

   (**Hint:** There is nothing "magical" about the input 00. The same result would hold for any other input. This is not a problem for Rice's Theorem—use a direct reduction from L_U.)

9. If L is a recursive language, does it mean that any L' such that $L' \subseteq L$ is also recursive? Justify your answer.

10. Show that the class of polynomial time languages, P, is closed under the following operations:

    (a) Reversal (if $L \in P$, so is $L^R = \{w | w^R \in L\}$)
    (b) Union (if $L_1, L_2 \in P$, so is $L_1 \cup L_2$)
    (c) Concatenation (if $L_1, L_2 \in P$, so is $L_1 L_2$)
    (d) Star (if $L \in P$, so is L^*)
    (e) Complementation (if $L \in P$, so is $\overline{L}$)

The End