CSL 705 Theory of Computation

Minor 1, Sem II 2009-10, Max 40, Time 1 hr

Name ______________________ **Entry No.** ______________________**Group**

Note (i) Write your answers neatly and precisely in the space provided with with each question including back of the sheet. You won't get a second chance to explain what you have written.

(ii) You can quote any result covered in the lectures without proof but any other claim should be formally justified.

1. Are the following statements true or false. Justify by a proof or a suitable counterexample. **(3 × 3 marks)**

   (a) If L_1 is finite and $L_1 + L_2$ is regular then L_2 is regular.
   Ans Let $L_3 = (L_1 \cup L_2) \cap (\bar{L_1})$ - this is regular. Therefore $L_2 = L_3 \cup (L_1 \cap L_2)$ is regular as the second component is finite (and regular).

   (b) If L_1 is regular (and infinite) and $L_1 \cdot L_2$ is regular then L_2 is regular.
   Ans False. Consider $L_1 = 0^*$ and $L_2 = 0^p$ where p is prime, then $L_1 \cdot L_2 = 000^*$ which is regular.

   (c) If L^* is regular, so is L.
   Ans False. Consider $L_2 = 0^p$ where p is prime, then $L_2^* = 000^*$ which is regular.

2. Use pumping lemma to show that $L = \{0^{i^2+2}, i \geq 0\}$ is not regular. **(5 marks)** **Ans** Assume L is regular and consider a string $s = 0^{n^2+2}$ which is long enough to apply pumping lemma. Then $s = uvw$ where $uv \leq n$ and $v > 1$, say $v = 0^k, 1 \leq k \leq n$. Then $uv^2w = 0^{n^2+2+k}$. Now note that $n^2 + 2 + k < (n+1)^2 + 2$ and hence does not belong to L.

3. Describe an algorithm to determine if a given regular language L equals Σ^* where Σ is the alphabet.**(4 marks)**
Ans Note that $L = \Sigma^*$ iff $\overline{L} = \phi$. The complement of L can be obtained by negating the output of the query for L.
Note that even if all strings of lengths n to 2n-1 are accepted it doesn't imply that $L = \Sigma^*$. All we know is that if L is infinite, then there is at least one string in L of this length.

4. Let L be all strings over $\{0, 1\}$ which when interpreted as a binary number is relatively prime to 12 (i.e., no common factors). Is L regular ? Justify. **(10 marks)**
Ans Note that the two prime factors of 12 are 2 and 3, so we will design languages L_{2b} and L_{3b} that consists of strings whose binary representations are divisible by 2 and 3 respectively. Then $L = \overline{(L_{2b} \cup L_{3b})}$ is regular if L_{2b} and L_{3b} are regular.

   Now construct a DFA for L_{3b} using the modulo classes as states and defining transitions in terms of $j * 2 mod 3$ and $j * 2 + 1 mod 3$ as the transitions from state j on inputs 0 and 1 respectively.

5. Describe a procedure to convert a regular expression r into an equivalent CFG G with some underlying justification.
Then use it to write a CFG for a language over $\{0, 1\}$ that does not contain 10 as a substring.**(6 + 6 marks)**
Ans Regular expressions are defined recursively in terms of

   - Base cases : $S \rightarrow a$ etc.
   - Union If S_1 and S_2 corresponds to two regular expression then $S \rightarrow S_1 | S_2$ corresponds to the union
   - Concatenation S_1 and S_2 corresponds to two regular expression then $S \rightarrow S_1 \cdot S_2$ corresponds to the concatenation.
   - Kleene * If S generates some regular expression R then $S \rightarrow S \cdot S | \epsilon$ corresponds to R^*.

   The proof follows by induction on the number of operators of the regular expression. The given

   strings can be represented by the r.e. $0^* \cdot 1^*$ and the above rules of concatenation and Kleene can be applied to obtain a grammar.
Note Some of the answers converted a r.e. to a DFA and constructed a grammar from the DFA. This requires writing out state equations and the proof about the correctness of the resultant grammar requires some albegraic arguments as it is not uniquely written. While this can be argued but in absence of that about half the marks were deducted.