Spring, 2010 CIS 511

Introduction to the Theory of Computation
Jean Gallier

Final Exam

May 11, 2010

Note that this is a **closed-book exam**
Read all the questions **before** starting solving any of them!

Problem 1 (10 pts). Given an alphabet Σ, sketch an algorithm to decide whether

$$(R + S)^* \cong \Sigma^*,$$

for any two regular expressions R and S over Σ.

Problem 2 (15 pts). For any integer, $n \geq 0$, let L_n be the language over the alphabet $\{a, b\}$ consisting of all strings of length n,

$$L_n = \{w \in \{a, b\}^* \mid |w| = n\}.$$

Prove that the DFA shown in Figure 1 is a minimal DFA accepting L_n, where the set of states is $\{0, 1, \ldots, n, n + 1\}$, the start state is 0, the only accepting state is n and the transition function is given by

$$\begin{aligned} \delta_n(i, a) = \delta_n(i, b) &= i + 1, \qquad 0 \leq i \leq n, \\ \delta_n(n + 1, a) = \delta_n(n + 1, b) &= n + 1. \end{aligned}$$

Problem 3 (20 pts). Let Σ be an alphabet, and let L_1, L_2, L be languages over Σ. Prove or disprove the following statements (if false, then provide a counter example).

(i) If $L_1 \cup L_2$ is a regular language, then either L_1 or L_2 is regular.

(ii) If L_1L_2 is a regular language, then either L_1 or L_2 is regular.

(iii) If L^* is a regular language, then L is regular.

Problem 4 (10 pts). Give a context-free grammar for the following language:

$$L_2 = \{a^m b^{2m} c a^{2n} b^n \mid m, n \geq 1\},$$

where $\Sigma = \{a, b, c\}$.

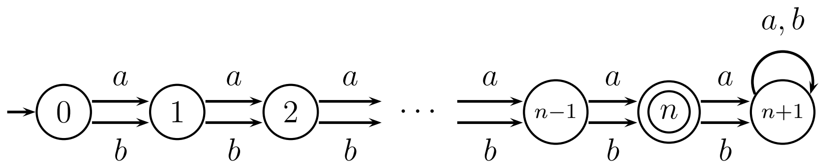

Figure 1: DFA for $L_n = \{w \in \{a, b\}^* \mid |w| = n\}$

Problem 5 (20 pts). Let $G = (V, \Sigma, P, S)$ be any context-free grammar. For any integer, $m \geq 0$, give an algorithm to decide whether $L(G)$ only generates strings of length $\geq m$.

Problem 6 (25 pts). (i) Prove that the following sets are not recursive ($\varphi_1, \varphi_2, \ldots, \varphi_i, \ldots$ is any acceptable indexing of the partial recursive functions):

$$\begin{array}{rcl} A & = & \{i \in \mathbb{N} \mid \varphi_i(0) \neq \varphi_i(1) \quad \text{and} \quad \varphi_i(0), \varphi_i(1) \text{ are both defined}\} \\ B & = & \{i \in \mathbb{N} \mid \varphi_i = \varphi_a * \varphi_b\} \\ C & = & \{\langle i, j, k\rangle \in \mathbb{N} \mid \varphi_i = \varphi_j * \varphi_k\} \\ D & = & \{i \in \mathbb{N} \mid \varphi_i(a) \text{ is undefined}\} \end{array}$$

where a and b are two fixed natural numbers.

(ii) Prove that D is not recursively enumerable.

Problem 7 (20 pts). Given any alphabet, $\Sigma = \{a_1, \ldots, a_m\}$, a regular expression, R, is said to be $*$-*free* iff it is built up from the atomic expressions, $a_1, \ldots, a_m$, $\emptyset$ and ϵ, using only $+$ and $\cdot$, that is, if S and T are any two $*$-free regular expressions, then $(S + T)$ and $(S \cdot T)$ are also $*$-free regular expressions (but S^* is *not* a $*$-free expression).

(i) Prove that for any $*$-free regular expression, R, for any string, w, if $w \in L_R$ (where L_R is the regular language denoted by R), then $|w| \leq |R|$.

Observe that the construction of an NFA, N_R, from a regular expression, R, (using the standard construction given in class) yields an NFA whose number of states is a most $2|R|$.

Prove that if R is $*$-free, then N_R is *acyclic*, which means that for every string, $w \neq \epsilon$, $q \notin \delta^*(q, w)$, for every state, q, of N_R.

Deduce from this that if N_R is constructed from a $*$-free regular expression, R, then for every string, w, if $|w| \leq |R|$, then we can decide in polynomial time (in $|R|$) whether $w \in L_R = L(N_R)$.

Use the above fact to prove that the problem of deciding whether $L_R \neq L_S$, for any two $*$-free regular expressions R and S is in $\mathcal{NP}$.

(ii) Reduce the satisfiability problem to the the problem of deciding whether $L_R \neq L_S$, for any two $*$-free regular expressions and thus, prove that this latter problem is $\mathcal{NP}$-complete.

Hint. For any Boolean proposition, $P = C_1 \wedge \cdots \wedge C_p$, if the propositional variables occurring in P are $x_1, \ldots, x_n$, produce two $*$-free regular expressions, R, S, over $\Sigma = \{0, 1\}$, such that P is satisfiable iff $L_R \neq L_S$. The expression S is actually

$$S = \underbrace{(0+1)(0+1)\cdots(0+1)}_{n}.$$

The expression R is of the form

$$R = R_1 + \cdots + R_p,$$

where R_i is constructed from the clause C_i in such a way that L_{R_i} corresponds precisely to the set of truth assignments that falsify C_i.