CSE 431
Spring Quarter 2004
Assignment 4
Due Friday, April 30, 2004

All solutions should be neatly written or type set. All major steps in proofs and algorithms must be justified.

1. (10 points) In this problem you will give the details of the reduction of the acceptance problem for Turing machines to the non-emptiness problem for two-headed finite automata, thereby showing its undecidablilty. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ be a one-tape Turing machine and let $w \in \Sigma^*$ be an input. The goal is to construct a two-headed finite automaton $M' = (Q', \Sigma', \delta', F)$ such that M accepts w if and only if M' accepts some input. As described in class M' will accept only strings that represent accepting computation histories of M on input w. Such a string has the form $C_1\#C_2\#\cdots\#C_m\$$ where (i) $C_1 = q_0 w \sqcup^k$ for some k, (ii) C_{i+1} follows from C_i in one step of M for $1 \leq i < m$, (iii) C_m contains q_a, and (iv) $|C_1| = |C_i|$ for $1 \leq i \leq m$. Recall that M' runs in three phases. In the first phase, the first heads scans the input to the first $\#$ checking condition (i) above. In the second phase, the two heads scan the input in parallel checking condition (ii) above. At the end of the second phase, if all goes well, the first head is on the \$ and the second head is on the last $\#$. In the third phase, the second head scans the string between the last $\#$ and the \$ checking for condition (iii) above. If all goes well, condition (iv) holds.

   Recall that in the second phase, after a brief startup, M' uses states of the form $(a_1, a_2, a_3, b_1, b_2, b_3)$ which are six-tuples of symbols from $Q \cup \Gamma \cup \{\#, \$\}$. The meaning of this state is that $a_1a_2a_3$ are the three symbols just scanned by second head and $b_1b_2b_3$ are the three symbols just scanned by the first head. The relation $L(a_1, a_2, a_3, b_1, b_2, b_3)$ simply states that $a_1a_2a_3$ and $b_1b_2b_3$ are consistent with a move of M. For example, if $a_2 \in Q$, $a_3 \in \Gamma$, and $\delta(a_2, a_3) = (p, b, R)$ then $b_1 = a_1$, $b_2 = b$, and $b_3 = p$. There are several other conditions as well that define L. You can assume L is given for the rest of the problem.

   (a) Precisely define all of the states Q' of M'. Include the states for all three phases.

   (b) Precisely define all the input symbols Σ' of M'.

   (c) Precisely define the transition function δ' of M'. The function δ' maps $Q' \times \Sigma' \times \Sigma'$ to $Q' \times \{R, S\} \times \{R, S\}$. R means move the head right while S means keep it stationary.

   (d) Precisely define the set of final (accepting) states F.

2. (10 points) Consider the Nonempty Intersection Problem for Context-Free Grammars:

Input:	Two context-free grammars G_1 and G_2
Property:	$L(G_1) \cap L(G_2)$ is nonempty.

We can use a reduction using accepting computation histories to show that this problem is undecidable. In this case an accepting history $C_1, C_2, \ldots C_m$ of M on input w is represented by the string

$$C_1 \# C_2^R \# C_3 \# C_4^R \# \cdots \# C_m$$

assuming that m is an odd number. We can alway assume that if M accepts w then it does so in an odd number of steps. One can design a context-free grammar G_1 that generates the language

$$L(G_1) = \{C_1 \# C_2^R \# C_3 \# C_4^R \# \cdots \# C_m : C_i \text{ yields } C_{i+1} \text{ for } i \text{ even and } C_1 = q_0 w \sqcup^k \text{ for some } k\}.$$

The easiest way to think about this is to see how a PDA can accept $L(G_1)$. The PDA would first check that the string up to the first $\#$ is in $q_0 w \sqcup^*$. This is a regular language so the PDA does not even need its stack. At this point the PDA would push the string up to the second $\#$ on to the stack. It would then compare the string up to the third $\#$ to the string stored on the stack to see if the string stored on the stack yields the string on the input. This checking can be done because there is only a finite area on the two strings that are different. Where they are different (near the state symbol) the PDA can check that the rules of the Turing machine were followed. This process is continued for every pair of consecutive strings where the first appears just after an odd number of $\#$'s. The PDA can then be converted to the context-free grammar G_1 using the equivalence of PDA's and context-free grammars.

Define carefully the language $L(G_2)$ so that M accepts w if and only if $L(G_1) \cap L(G_2)$ is nonempty. Briefly describe how a PDA can accept the language $L(G_2)$.