Problem Set 8

Spring 10

Due: Tuesday, April 20, in class before the lecture.

Please follow the homework format guidelines posted on the class web page:

`http://www.cs.uiuc.edu/class/sp10/cs373/`

1. E.T. [**Category**: Puzzle, **Points**: 5]

   Show that the problem of deciding whether aliens exist in the universe is decidable. More precisely, show that there is a Turing machine that will print "YES" if there are aliens in the universe, and "NO" otherwise!

2. Reduction à la Rice's Theorem [**Category**: Proof, **Points**: 20]

   A language $L \subseteq \Sigma^*$ is *closed under reversal* if for every $w \in L$, $w^R \in L$.

   Show that $L_{rev} = \{\langle M \rangle \mid M \textit{ is a TM and } L(M) \textit{ is closed under reversal}\}$ is undecidable.

   You may not simply appeal to Rice's theorem (however, you can adapt the proof of Rice's theorem to solve this problem).

3. Queueueueueueueue [**Category**: Proof, **Points**: 20]

   A *queue automaton* is an automaton with finitely many states, that can manipulate an (unbounded) queue data-structure. Fix an input alphabet Σ and a queue alphabet Γ, where $\Sigma \subseteq \Gamma$. The input, a word in Σ^*, is given to the queue automaton in the queue, and in each step the automaton can *enqueue* a letter onto the queue, or dequeue a letter from the queue. A queue is simply a FIFO (first-in-first-out) data-structure, and can contain any number of letters. The queue automaton is non-deterministic, and accepts a word if there is some way to reach an accept state.

   Show that the membership problem for queue automata in *undecidable*.

   In other words, show that, given a queue automaton QA and a word $w \in \Sigma^*$, checking whether QA accepts w is undecidable.

   Your answer can be at a high-level description of a reduction.

   Below is a *formal* description of a queue automaton in case you want to understand the question better using a more precise description (you need not give the reduction in this kind of detail).

   Let $\Gamma_\epsilon = \Gamma \cup \{eps\}$.

   A queue automaton is a tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc})$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, $q_{acc} \in Q$ is the accepting state, and $\delta \subseteq Q \times \Gamma_\epsilon \times \Gamma_\epsilon \times Q$.

   Intuitively, if $(q, a, b, q') \in \delta$, then it means that the automaton can go from state q to state q' by dequeuing a from the queue and enqueing b to the queue.

   Formally, a *configuration* of a queue automaton is a pair (q, x) where $q \in Q$ and $x \in \Gamma^*$ (q is the state the queue automaton is in, and x is the content of the queue, with the head of the queue being the first letter in x and the tail of the queue being the last letter in x).

We define the transitions between configurations as follows: for any $x \in \Gamma^*$, $a, b \in \Gamma$, $(q, ax) \rightarrow (q', xb)$ iff $(q, a, b, q') \in \delta$. (This captures a move that dequeues a and enqueues b.)

A word w is *accepted* by the queue automaton if there is a sequence of configurations $C_1, C_2, \ldots, C_n$ such that $C_1 = (q_0, w)$, for each $1 \leq i < n$, $C_i \rightarrow C_{i+1}$, and $C_n = (q_{acc}, y)$ for some $y \in \Gamma^*$.

4. Nondeterminism [**Category**: Construction, **Points**: 20]

   For every natural number n, let n_b be the binary representation of n. For example, $5_b = 101$. Assume there is a TM, *Multiplier*, that when given inputs m_b and n_b on two tapes, outputs $(m * n)_b$ on the third tape. The TM *Multiplier* is provided for you as a black box that you can use.

   Construct a *nondeterministic Turing machine* M_{comp} to decide if a natural number x, represented as x_b, is a *composite number* (a composite number is a number that is not prime). Your NTM must be a decider (i.e. halt no matter what non-deterministic choices it makes) and furthermore halt within $O(poly(|x_b|))$ steps (i.e. work in polynomial time). To do the latter, you must exploit *non-determinism*.

   Describe your construction clearly (it need not be *formal*) and in sufficient detail so that is understandable, clear and easy to see it's correct.

5. Dovetailing [**Category**: Construction, **Points**: 20]

   Prove that the language $L_{two} = \{ \langle M \rangle \mid |L(M)| \geq 2\}$ is Turing-recognizable. Informally, L_{two} is the set of Turing machines that accept at least *two* strings.

6. (Extra Credit) Highly Non-recognizable (NOT COMPULSORY FOR HONORS) [**Category**: Proof, **Points**: 20]

   Let $L_{ALL} = \{\langle M \rangle | M \text{ is a TM with input alphabet } \Sigma \text{ and } L(M) = \Sigma^*\}$.

   Informally, L_{ALL} is the set of TMs that accept every input string.

   We want you to show that neither L_{ALL} nor its complement is TM-recognizable!

   You can assume that all strings encode some Turing machine, and hence $\overline{L_{ALL}} = \{\langle M \rangle | M \text{ is a TM with input alphabet } \Sigma \text{ and } L(M) \neq \Sigma^*\}$.

   (a) Prove that $\overline{L_{ALL}}$ is not TM-recognizable.

   Hint: Be careful when using reductions to prove non-recognizability. When you reduce A to B in order to show that if B was recognizable, then A is recognizable, you create a recognizer for A using a recognizer for B. However, you must be careful *not to flip the answer given by the oracle recognizing* B as recognizable languages are not closed under complement.

(b) Prove that L_{ALL} is non-recognizable.

Hint: This direction is trickier. Assume L_{ALL} is recognizable and that M_{ALL} is a TM recognizing it. Note that we cannot assume that M_{ALL} halts on all inputs. All we know is that it accepts x iff $x \in L_{ALL}$. You may use the existence of M_{ALL} to show that $\overline{A_{TM}}$ is recognizable, which we know is false. Also, when you construct the recognizer, when it is given input $\langle M, w \rangle$, you may want to consider simulating M on w for a *finite* number of steps.